Advances in Information Security 105

Sushmita Ruj
Salil S. Kanhere
Mauro Conti *Editors*

Blockchains

A Handbook on Fundamentals, Platforms and Applications

Springer

Advances in Information Security

Volume 105

Series Editors
Sushil Jajodia, George Mason University, Fairfax, VA, USA
Pierangela Samarati, Milano, Italy
Javier Lopez, Malaga, Spain
Jaideep Vaidya, East Brunswick, NJ, USA

The purpose of the *Advances in Information Security* book series is to establish the state of the art and set the course for future research in information security. The scope of this series includes not only all aspects of computer, network security, and cryptography, but related areas, such as fault tolerance and software assurance. The series serves as a central source of reference for information security research and developments. The series aims to publish thorough and cohesive overviews on specific topics in Information Security, as well as works that are larger in scope than survey articles and that will contain more detailed background information. The series also provides a single point of coverage of advanced and timely topics and a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook.

Sushmita Ruj • Salil S. Kanhere • Mauro Conti
Editors

Blockchains

A Handbook on Fundamentals, Platforms and Applications

Springer

Editors

Sushmita Ruj
School of Computer Science and Engineering
University of New South Wales
Sydney Kensington, NSW, Australia

Salil S. Kanhere
School of Computer Science and Engineering
University of New South Wales
Sydney Kensington, NSW, Australia

Mauro Conti
Department of Mathematics
University of Padova
Padova, Italy

ISSN 1568-2633 ISSN 2512-2193 (electronic)
Advances in Information Security
ISBN 978-3-031-32145-0 ISBN 978-3-031-32146-7 (eBook)
https://doi.org/10.1007/978-3-031-32146-7

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Contents

Part I Building Blocks

Ten Myths About Blockchain Consensus 3
David Hyland, João Sousa, Gauthier Voron, Alysson Bessani, and Vincent Gramoli

Cryptographic Primitives 25
Mayank Raikwar and Shuang Wu

Part II Popular Blockchains

Bitcoin Blockchain System: An Overview of Security and Privacy Aspects 75
Mauro Conti, Ankit Gangwal, Chhagan Lal, and Sushmita Ruj

The Ethereum Blockchain: Implementation and Security Aspects 109
Alessandro Brighente, Mauro Conti, and Andrea De Salve

The Future Ring Confidential Transaction Protocols for Privacy-Preserving Blockchain 147
Muhammed F. Esgin, Joseph K. Liu, Shi-Feng Sun, and Dimaz Ankaa Wijaya

Algorand Blockchain 173
Arash Mirzaei, Amin Sakzad, Ron Steinfeld, and Jiangshan Yu

Tendermint 195
Yackolley Amoussou-Guenou, Antonella Del Pozzo, and Sara Tucci-Piergiovanni

Part III Security and Scalability

The Security of Delegated Proof-of-Stake Wallet and Stake Pools 225
Mario Larangeira and Dimitris Karakostas

Layer 2 Scaling Solutions for Blockchains 261
Subhra Mazumdar and Sushmita Ruj

Illicit Blockchain Content: Its Different Shapes, Consequences, and Remedies 301
Roman Matzutt, Martin Henze, Dirk Müllmann, and Klaus Wehrle

Blockchain-Based Distributed and Secure Digital Forensic Investigation Systems 337
Mauro Conti, Gulshan Kumar, Chhagan Lal, and Rahul Saha

Part IV Applications

Supply Chain Management Using Blockchain 365
Christopher Klinkmueller, H. M. N. Dilum Bandara, Xiwei Xu, and Qinghua Lu

Blockchain Technology for E-Governance Applications 399
Ras Dwivedi, Nilesh Vasita, Mukul Verma, Tanmay Yadav, and Sandeep Shukla

When Blockchain meets Smart Cities: Opportunities, Security and Future Research 423
Roben C. Lunardi, Regio A. Michelin, Maher Alharby, Volkan Dedeoglu, Henry C. Nunes, Eduardo Arruda, Avelino F. Zorzo, and Aad van Moorsel

Decentralized Identity Management and Blockchains: Design Patterns and Architectures 465
Hye-young Paik, Yue Liu, Qinghua Lu, and Salil S. Kanhere

From Centralized to Decentralized Remote Electronic Voting 493
Christian Killer, Bruno Rodrigues, Eder John Scheid, Muriel Figueredo Franco, and Burkhard Stiller

Blockchain Technology Accelerating Industry 4.0 531
Jan Pennekamp, Lennart Bader, Eric Wagner, Jens Hiller, Roman Matzutt, and Klaus Wehrle

Blockchain for Health Data Management 565
Dinh C. Nguyen, Pubudu N. Pathirana, Ming Ding, and Aruna Seneviratne

Supporting Secure Trusted Manufacturing via Blockchain 587
Ali Dorri, Sabah Suhail, Zahra Jadidi, Rasheed Hussain, Colin Fidge, and Raja Jurdak

Blockchain for Data Sharing 601
Gowri Sankar Ramachandran and Bhaskar Krishnamachari

About the Editors

Sushmita Ruj is Faculty of Engineering Lead of UNSW Institute for Cybersecurity, IFCYBER and Senior Lecturer in the School of Computer Science and Engineering at UNSW, Sydney. Her research interests are in applied cryptography, post-quantum cryptography, blockchains and privacy enhancing technologies. She designs practical, efficient, and provably secure protocols that can be deployed in real-world applications. She has published over 110 peer-reviewed articles and delivered over 100 technical lectures around the world. She has won several competitive grants like Samsung GRO Award, NetApp Faculty Fellowship and Cisco Academic Grant. She is an Associate Editor of the *Transactions on Information Forensics and Security*. Before joining UNSW, she was a Senior Research Scientist at CSIRO's Data61, an Associate Professor at Indian Statistical Institute and an Assistant Professor at Indian Institute of Technology, IIT, Indore. She was a visiting researcher/faculty at NTU, Singapore; KDDI R&D Labs, Japan; INRIA, France; Kyushu University, Japan; and Microsoft Research Labs, India. She was a member of the "Blockchain for Cybersecurity" working group of the National Blockchain Roadmap of Australia and the Blockchain working group founded by the Reserve Bank of India. Sushmita is a senior member of both ACM and IEEE.

Salil S. Kanhere is a Professor of Computer Science and Engineering at UNSW Sydney. He is also affiliated with the Cybersecurity Cooperative Research Centre (CSCRC) and the UNSW Institute for Cyber Security (IFCYBER). His research interests span the Internet of Things, cyberphysical systems, cybersecurity, blockchain and applied machine learning. He has published over 350 peer-reviewed articles and is leading several government and industry-funded research projects on these topics. He received the Friedrich Wilhelm Bessel Research Award (2020) and the Humboldt Research Fellowship (2014), from the Alexander von Humboldt Foundation in Germany. He is an ACM Distinguished Member and served as an ACM Distinguished Speaker from 2019 to 2021. He is an IEEE Computer Society Distinguished Visitor and a Senior Member of the IEEE. He has held visiting positions at I2R Singapore, Technical University Darmstadt, University of Zurich, RWTH Aachen and Graz University of Technology. He serves as the

Editor in Chief of the *Ad Hoc Networks* journal and as an Associate Editor of *IEEE Transactions on Network and Service Management, Computer Communications* and *Pervasive and Mobile Computing*. He regularly serves on the organising committee of IEEE/ACM international conferences. Salil co-authored a book titled *Blockchain for Cyberphysical Systems* published by Artech House in 2020.

Mauro Conti is Full Professor at the University of Padova, Italy. He is also affiliated with TU Delft and University of Washington, Seattle. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, the Netherlands. In 2011 he joined as Assistant Professor at the University of Padova, where he became Associate Professor in 2015 and Full Professor in 2018. He has been Visiting Researcher at GMU, UCLA, UCI, TU Darmstadt, UF and FIU. He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His research is also funded by companies including Cisco, Intel and Huawei. His main research interest is in the area of security and privacy. In this area, he has published more than 550 papers in topmost international peer-reviewed journals and conferences. He is Editor-in-Chief for *IEEE Transactions on Information Forensics and Security*, Area Editor-in-Chief for *IEEE Communications Surveys and Tutorials* and has been Associate Editor for several journals, including *IEEE Communications Surveys and Tutorials, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security* and *IEEE Transactions on Network and Service Management*. He was Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, CSS 2021, WiMob 2023 and ESORICS 2023, and General Chair for SecureComm 2012, SACMAT 2013, NSS 2021 and ACNS 2022. He is Fellow of the IEEE, Fellow of the AAIA, Senior Member of the ACM and Fellow of the Young Academy of Europe.

Part I
Building Blocks

Ten Myths About Blockchain Consensus

David Hyland, João Sousa, Gauthier Voron, Alysson Bessani, and Vincent Gramoli

1 Introduction

Over the last decade, blockchain experienced an important momentum leading to a plethora of new consensus protocol proposals (e.g., Ripple's consensus algorithm [55], Democratic Byzantine Fault Tolerance (DBFT) [24], Aura [41], Casper [16], HotStuff [65], IBFT [46], Tendermint [43]). A consensus protocol is a key element of the blockchain system as it helps a distributed set of machines agree on a unique block at each given index of a chain. In contrast with the problem of consensus that has been known by the distributed computing community for the past four decades [52], a significant part of these proposals is often unclear. Most of these new consensus protocols are described in white papers, wikis, and online documentations, rather than in more traditional academic publications, and it is unclear whether they satisfy the application requirements [19].

As a result, there are several misconceptions about what is blockchain consensus, the guarantees it should offer, and how it could be made secure and fast. For example, efforts to standardize blockchain technologies have revealed ambiguous terminologies [36], and, when consensus protocols are stated informally, one can

D. Hyland · G. Voron
University of Sydney, Camperdown, NSW, Australia
e-mail: dhyl2836@uni.sydney.edu.au; gauthier.voron@sydney.edu.au

J. Sousa · A. Bessani
Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal
e-mail: jcsousa@fc.ul.pt; anbessani@fc.ul.pt

V. Gramoli (✉)
University of Sydney, Camperdown, NSW, Australia

EPFL, Lausanne, Switzerland
e-mail: vincent.gramoli@sydney.edu.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_1

easily find scenarios in which these protocols fail [6, 19, 33, 60]. As blockchains are becoming important components to guarantee the security of critical applications, these myths can have devastating effects, whose latest example is probably the vulnerability of some of the mostly deployed blockchain software (e.g., `parity` and `geth`) [31], used to handle high-value digital assets. Other examples include the recent efforts devoted to develop quantum-resilient cryptographic software on top of blockchain systems that are already vulnerable to the misbehavior of a single participant [20]. While cryptography and fault tolerance address problems that may seem orthogonal, a blockchain cannot provide the security level required by such applications without both cryptography and fault tolerance (cf. Myth #7).

As blockchains are now being offered as a service by most cloud providers and have become the cornerstone of various applications, it is crucial to clarify some of the misconceptions that will have, sooner or later, dramatic consequences. Some of these misconceptions could be attributed to the lack of knowledge of the distributed computing and database literatures. First, there are various failure models in which an algorithm can solve consensus. In fact, a consensus algorithm typically allows n nodes (or processes) reach an agreement despite f of them failing. These failures are generally classified in two types: *crash*, where a node simply stops, and *Byzantine*, where a node behaves arbitrarily, misbehaving either accidentally or with malicious intent. This is why we distinguish crash fault-tolerant (CFT) from Byzantine (or arbitrary) fault-tolerant (BFT) protocols (cf. Myths #6 and #8). Second, the three properties of the classic consensus problem [44] are often misinterpreted: (i) validity requires that the value decided by a non-faulty (or correct) node has to be valid, (ii) agreement requires that two non-faulty nodes cannot decide differently, and (iii) termination requires that eventually the non-faulty nodes decide. Also, the factors affecting the performance of a blockchain are often misunderstood (Myths #4 and #5) as they typically embed, in addition to the consensus protocol, other components like cryptography (Myth #11). This may distract engineers from other challenging problems (Myths #9 and #10). One of the most common misinterpretations is illustrated by the large family [49] of so-called proof-of-$*$ consensus (*sic*) that cannot ensure agreement upon a unique block and may lead the blockchain to counterintuitively fork into multiple branches (cf. Myth #3).

The aim of this chapter is to bust ten myths about blockchain consensus, summarized in Table 1. These myths correspond typically to misinformation that professionals and students commonly learn by reading posts and blogs online before attending a blockchain course. We proceed by listing each myth and explaining why it is incorrect by offering a counterexample, sometimes using a blockchain (Hyperledger Fabric, Red Belly Blockchain, or R3 Corda) or a consensus algorithm (BFT-SMaRt, DBFT, or HotStuff). Our aim is not to survey existing approaches for blockchain consensus, as there is an extensive literature on the subject (e.g., [19, 49]), neither to provide a formal treatment of these myths but rather to state them in a pedagogical language to reach the blockchain community at large. As some of these clarifications already helped improving the scalability of blockchains (Myth #12), we hope that they will help build secure and efficient blockchain applications in the future.

Table 1 A summary of common blockchain consensus myths

# Myth	Clarification	At risks
1. Po* solves consensus	Po* selects blocks without guaranteeing uniqueness	Users
2. Consensus bottlenecks blockchain in LANs	BFT-SMaRt runs faster outside Hyperledger	Designers
3. Consensus bottlenecks distributed ledgers in WANs	DBFT and BFT-SMaRt scale better outside Corda	Designers
4. CFT consensus tolerates Byzantine faults	BFT quorums are larger than CFT ones	Users
5. Signatures and hashes make blockchain secure	They do not cope with disagreement and double spending	Users
6. A CFT blockchain with a BFT consensus becomes BFT	The whole communication pattern needs to be changed	Users + Designers
7. BFT consensus needs linear message complexity	Some quadratic complexity algorithms perform better	Designers
8. Reconfiguring consensus participants is easy	It is difficult to make it non-disuptive	Designers
9. Blockchain performance is not limited by cryptography	CPU demand can surpass the network demand of consensus	Designers
10. Blockchain needs to solve the classic consensus problem	Deciding $\Omega(n)$ proposals help scale	Designers

2 Background on Consensus and Proof-of-$*$

Before listing each myth, we present various interpretation of the term "consensus" in the blockchain context. A blockchain is easily understood as a chain of blocks abstraction where new blocks get regularly appended; however, the system that replicates this abstraction on multiple machines or *nodes* of the network does not always maintain the sequence structure of the chained blocks.

Instead a more precise description of this abstraction is a directed acyclic graph or—to put it simply—a tree structure whose nodes are blocks and whose edges are directed upward in the tree, from children blocks to their parent block [35]. Each of these edges is implemented as a hash of the content of the parent block stored as a field of the child block. A parent block has multiple children in the tree as soon as the blockchain *forks*, which means that two nodes disagree about the block that should be inserted at the next available index of the chain. The *consensus* abstraction is employed to guarantee that there is no such disagreement among all nodes about the unique block to be appended next.

The consensus problem was defined more than half a century ago [52] and requires to guarantee that if replicas propose their block, then no two replicas should decide differently (agreement), the block that is decided is one of the proposed block (validity), and the non-faulty replicas should eventually decide (termination). While this definition presents some inherent limitations for scalability that will be discussed in Myth #11, it paved the way for research on consensus

algorithms in small settings and influenced newer definitions of the consensus problem considering cryptography [18] and scalability [24, 25].

Interestingly, the term "consensus" has been extensively used on blog posts and websites about blockchain to denote proof-of-∗ methods of selecting a subset of blocks proposed to a particular index of the blockchain. As surveyed in [49], these proof-of-∗ (Po*) methods include proof-of-work, proof-of-reputation, proof-of-lock, proof-of-activity, proof-of-stake, proof-of-burn, proof-of-authority, and proof-of-location, each choosing a different local attribute in order to decide whether a block is selectable to a particular index. These attributes include the computational power of the node, stake owned by the node in the considered blockchain system, its activity, location, etc. The diversity of existing proof-of-∗ makes it difficult to list them all here and is not part of the scope of this paper.

3 Myth #1: Proof-of-∗ Solves Consensus

A proof-of-work [29] is a mechanism to limit the power of the adversary that can control the Byzantine participants. By requesting each participant to solve a moderately complex crypto-puzzle and produce a proof of this effort, called proof-of-work, before producing a block [48], the ability for the adversary to overwhelm the system with new valid blocks becomes limited by its computational power. Many alternatives to this proof-of-work mechanism have been proposed, like proof-of-stake that limits the power of the adversary to its stake, hence leading to the large family of proof-of-∗ mechanisms. For a list of more than a dozen of these proof-of-∗ proposals, we refer the interested reader to a survey on the topic [49].

As it is widely believed that proof-of-∗ protocols solve consensus,[1] many protocols have been called "proof-of-∗ consensus (sic)." As an example, Cointelegraph explains how proof-of-work can be seen as the "original consensus algorithm in a blockchain network" by using a "complicated mathematical puzzle and a possibility to easily prove the solution."[2] It may seem as if "proof-of-work consensus" was a metonymy aimed at being pedagogical and referring to proof-of-work as the component of a more complex consensus procedure. However, the frequent forks [64] in proof-of-work blockchains indicate the possibility of disagreements, which contradict the scientific notion of consensus [52]. This metonymy, albeit simple, hides more complex intricacies. Such simplifications put the users at risk, because, strictly speaking, consensus prevents double spending that could be induced by forks whereas proof-of-∗ actually tolerates forks and may lead to double spending.

[1] Wikipedia makes use of the term "proof-of-work consensus (*sic*)" at https://en.wikipedia.org/wiki/Proof_of_work.

[2] https://cointelegraph.com/explained/proof-of-work-explained.

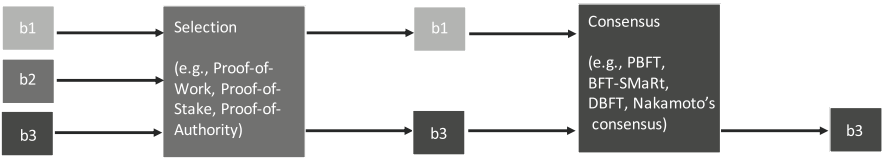


Fig. 1 Proof-of-$*$ mechanisms do not solve the consensus problem. Instead, they select a subset, say, $\{b1, b3\}$, of proposed blocks, say, $\{b1, b2, b3\}$, as legitimate based on some attribute (computational power, coins owned, etc.) of the nodes that generated them, but a separate consensus algorithm is necessary to guarantee that the block, like $b3$, decided for a given index of the chain is unique

In fact, the proof-of-$*$ mechanism cannot solve consensus but is instead a mechanism to enforce some of the consensus prerequisites so that a proper consensus algorithm can be executed. Typically, a proof-of-$*$ mechanism selects a small subset of nodes to participate in the consensus by proposing a block for the same index of the chain as depicted in Fig. 1, where $\{b1, b3\}$ are proposed. As this mechanism cannot guarantee that a unique block is proposed, it is insufficient to decide a single block and does not solve the consensus problem. Recently, researchers have tried to clarify the terminology to address this confusion [36], but it remains well spread on the Internet.

3.1 Example: Bitcoin

Bitcoin [48] uses proof-of-work to limit the number of nodes that can create a new block for a given index of the chain. Once these blocks are created, Bitcoin solves consensus by deciding among multiple candidate blocks (that all contain a valid proof-of-work) the unique block that is part of the longest branch. This is the reason why Bitcoin [48] uses an extra mechanism, Nakamoto's consensus, to try to solve the problem: it selects the blocks of the longest branch by assuming that the delay to create and propagate every block to all miners in the network is upper-bounded such that every miner can safely choose the candidate block at index i once they have waited long enough [34]. The proof-of-work mechanism helps limit the block creation speed by requiring that a node first solves a computationally hard crypto-puzzle before propagating a new block. This proof-of-work mechanism alone does not ensure the uniqueness of the block at each index—this is precisely why a separate consensus protocol based on identifying the block of the longest branch is required in Bitcoin.

4 Myth #2: Consensus Is the Bottleneck of Blockchains in LANs

Distributed consensus performance (expressed, e.g., in decisions per unit of time) generally degrades with the number of participants. This led to the folklore belief that the performance of consensus is the key limiting factor of the performance of the blockchain in normal executions as was described by NEC.[3] Typically, BFT consensus algorithms involve a quadratic number of messages [13, 21, 24] that are believed to rapidly consume the limited bandwidth resource. For example, the classic PBFT consensus algorithm [21] employs a distinguished participant, the leader, proposing a value and having participants exchanging sufficiently many prevotes to make sure that no other value will be accepted and then having participants exchanging sufficiently many votes to make sure the value is accepted by other participants before deciding this value.

For this reason, it has long been thought that a blockchain performance is limited by the performance of its consensus, and recent solutions, like HotStuff [65], attempted to boost blockchain performance by reducing the message complexity of the consensus algorithm using threshold signatures (as discussed in Myth #9). HotStuff follows the classic leader-based pattern but reduces the quadratic message complexity, induced by the participants sending prevotes and votes to each other, to the linear complexity, induced by the participants sending their prevotes and votes only to the leader that retransmits them.

It turns out that there are other scalability limitations in blockchain systems, especially in local area networks (LANs)—one example is the verification of transaction signatures [12, 62]. Traditional blockchain would require every node to verify the signatures of every transaction. Although there are some blockchains that employ verification sharding [25] for each signature to be verified by only $f + 1$ nodes, in general, even this optimization is not sufficient to make the verification finish significantly earlier. We present the cost of verification sharding in Myth #11. Additionally, previous works [66] point the execution of smart contracts as a significant scalability limitation, specifically the frequent disk operations they cause as well as their computational cost.

4.1 Example: Hyperledger Fabric

In order to test our hypothesis on the existence of other performance limitations, we selected the Hyperledger Fabric blockchain [3] that aims at being modular and offering privacy. Even though Hyperledger Fabric v1.x is crash fault tolerant and does not aim at tolerating malicious behaviors, there exists a prototype version [57] that makes use of a BFT orderer (cf. Myth #8 for more details) based on BFT-

[3] https://www.nec.com/en/global/insights/article/2020022520/index.html.

SMaRt. BFT-SMaRt [13] is a modular BFT consensus protocol in which a stable leader sends a proposal to all nodes that confirm its consistency in two all-to-all communication steps, similar to PBFT [21]. The goal of the test is to observe whether Hyperledger Fabric could offer similar performance as its consensus or orderer component alone, when run in the same environmental settings.

We ran the Hyperledger Fabric blockchain system and a BFT-SMaRt baseline on four machines, each with Ubuntu Linux 14.04.1 LTS, hosted in Dell PowerEdge R410 servers. Each machine has 32 GB of memory and two quad-core 2.27 GHz Intel Xeon E5520 processors with hyper-threading. These machines communicate through an isolated gigabit Ethernet LAN. The results were obtained with Hyperledger Caliper [17] and are averaged over five runs with error bars showing the standard deviation. Figure 2 depicts throughput for HLF-BFT for blocks of 10 (HLF-BFT10) and 100 (HLF-BFT100) transactions of payload 4 KiB and envelopes of 900 bytes, HLF-Solo for blocks of 100 transactions (Caliper benchmarks could not complete with blocks of 10) and a BFT-Only benchmark with blocks of 10 transactions. We can clearly observe that the Byzantine fault-tolerant orderer based on BFT-SMaRt [57] performs 2× better than Hyperledger Fabric with the same version of this orderer. Furthermore, the BFT-SMaRt replication library alone, without the required Fabric machinery, can order 5× more 4 KiB transactions per second than what was obtained in the orderer [13].

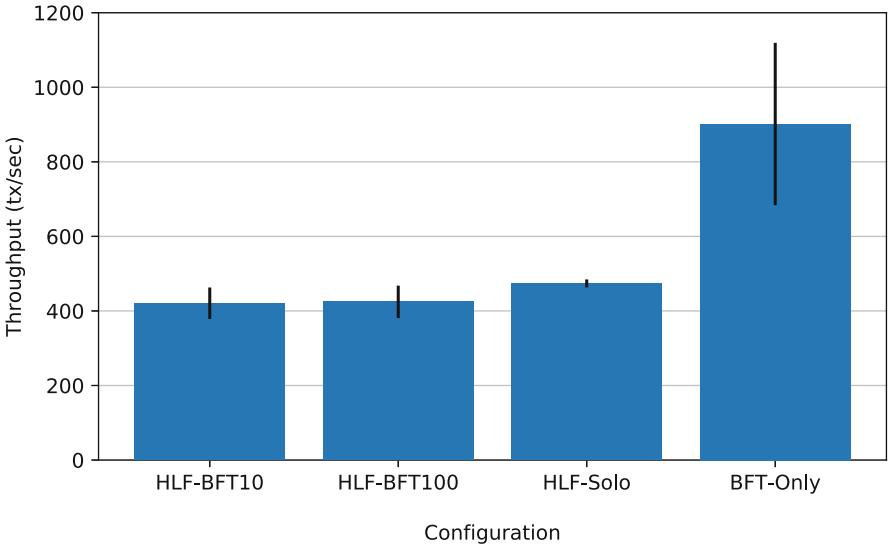


Fig. 2 In a LAN, Hyperledger Fabric running with an experimental BFT orderer based on BFT-SMaRt is slightly slower than without consensus (HLF-Solo) but significantly slower than the BFT orderer alone (BFT-Only). This indicates that the bottleneck of blockchains is not necessarily their consensus component

5 Myth #3: Consensus Is the Bottleneck of Distributed Ledgers in WANs

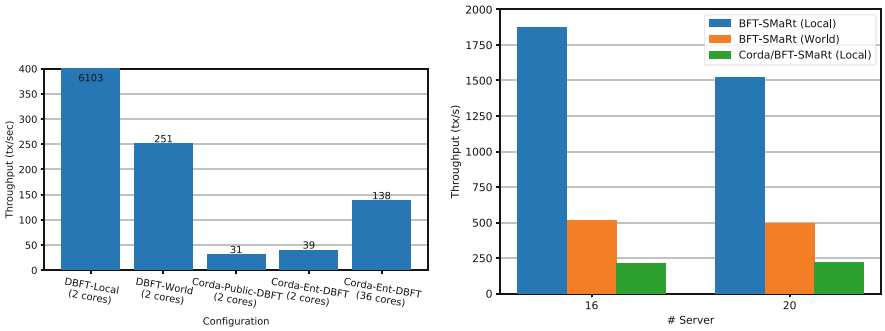
Myth #4 showed that consensus is not necessarily the bottleneck of a blockchain in a LAN. This result could be attributed to the blockchain prototype in use, and it is interesting to explore whether the same observation would apply to a distributed ledger like Corda [15] that aims at enabling businesses to transact in privacy using smart contracts without necessarily chaining blocks. Additionally, Myth #4 did not contradict the belief that the consensus protocol could be the bottleneck of a distributed ledger deployed on the Internet. A LAN typically offers bandwidth resources that are not comparable to the resources one could obtain in a wide area network (WAN) subject to congestions, long distances, etc.

Here we push the study one step further by selecting a recent BFT consensus protocol designed especially for distributed ledgers, DBFT [24], and two versions of Corda, its public version and its enterprise edition, and running experiments across four regions, in Australia, Brazil, Europe, and the USA. The goal of these new sets of experiments is to pinpoint what overhead comes from the consensus itself and what overhead comes from other parts of the distributed ledger. If consensus was truly the bottleneck of the distributed ledger, the performance of the distributed ledger would be comparable to the performance of its bottleneck when running in isolation.

5.1 Example: Corda

Corda is crash fault tolerant as it uses the Raft consensus algorithm and cannot tolerate Byzantine failures. In fact, some of us tried deploying a BFT prototype of Corda running on BFT-SMaRt in the past, however, we were not able to make it run, and the Corda developers recommended that we use the crash fault-tolerant version instead [38]. We thus modified Corda to use BFT-SMaRt and DBFT as consensus algorithms. DBFT is a formally verified consensus protocol [24, 60] that differs from the previously mentioned leader-based protocols as each of its n participants reliably broadcasts its proposal to other nodes and spawns n binary consensus instances that define a bitmask in order to output a decision. We measure both the throughput of the Corda distributed ledger, its public and enterprise version Corda 4.0-SNAPSHOT, and the throughput of the BFT-SMaRt and DBFT consensus algorithms, as the average observed over four runs on Amazon Web Services (AWS) EC2 instances. As in Corda, each transaction has a serialized size of 8 KiB bytes (that is much larger than the original size); we use the same transaction size for running the consensus algorithms alone.

Figure 3a compares the throughput of DBFT and Corda with DBFT on four c5.large instances, each with one hyperthreaded Intel Xeon 8124M core. We also add the throughput of Corda enterprise with DBFT on 4 c5.9xlarge instances, each with 18 hyperthreaded Intel Xeon 8124M cores. We execute DBFT in two setups. In



(a) (b)

Fig. 3 The bottleneck of distributed ledger is not necessarily the consensus protocol. (**a**) When running with the DBFT consensus, the public and the enterprise versions of the Corda distributed ledger are significantly slower than DBFT alone. (**b**) When running with the BFT-SMaRt consensus, the enterprise version of Corda runs slower than BFT-SMaRt alone. (**a**) Corda w/ DBFT vs. DBFT alone. (**b**) Corda w/ BFT-SMaRt vs. BFT-SMaRt alone

the "DBFT-Local" setup, the four instances are in a single datacenter. In the "DBFT-World" setup, each instance is in a separate datacenter located in Ohio, Frankfurt, Sydney, and São Paulo. We execute Corda in a local setup only as we did not manage to deploy it on a geo-distributed setup. The result indicates that with comparable setups the throughput of DBFT is 156× higher than the throughput we could obtain from any version of Corda running DBFT with the same number of cores. Moreover, DBFT executed on a georeplicated setup on a 2-core machine still has a throughput 1.8× higher than Corda on a local setup with a 36-core machine and 6.4× higher than Corda on a local setup with a 2-core machine. We were not able to locate precisely the cause of the Corda performance drop; however, we observe that Corda enterprise performance increases with the number of cores. This seems to indicate that the throughput of Corda is limited by computation.

In Fig. 3b we compare the throughput of BFT-SMaRt and Corda enterprise with BFT-SMaRt on 16 and 20 c5.4xlarge instances, each with 8 hyperthreaded Intel Xeon Platinum 8275CL cores. Similar to DBFT, deploy BFT-SMaRt in two setups. In the "Local" setup, all instances are in the same datacenter. In the "World" setup, instances are spread among four datacenters in Ohio, Frankfurt, Sydney, and São Paulo. We execute Corda in a local setup only as we did not manage to deploy it on a geo-distributed setup. We observe that, on a local setup, the throughput of BFT-SMaRt is consistently more than 6× larger than the throughput of Corda. Moreover, even when executed in a geo-distributed setup, the throughput of BFT-SMaRt is still more than 2× larger than Corda executing in a single datacenter. Since the only difference between BFT-SMaRt and Corda is the set of operations outside the consensus, this performance gap indicates that these additional operations are the bottleneck of the Corda distributed ledger.

6 Myth #4: CFT Consensus Algorithms Are Safe Under Byzantine Faults

There is a misconception that CFT protocols are enough for implementing a safe blockchain consensus. In particular, the possibility that Byzantine nodes can affect the safety of these protocols is often overlooked: two recent consensus algorithms, called Clique and Aura [41], have been integrated in two of the mostly deployed blockchain software, called `geth` and `parity`, in order to cope with $\lceil n/2 \rceil - 1$ Byzantine failures, a problem that is known to be unsolvable [14]. As a result, the Attack of the Clones led to double spending in two testnets running these software [31], hence demonstrating, in practice, this safety violation. More generally, consider a system with n nodes in which up to f can fail. In a practical system, where there are no strict time bounds for communication, a node can wait for responses from at most $n - f$ nodes (a quorum) since up to f nodes can be faulty and never answer. In such systems, quorums must intersect in at least one correct process to ensure actions observed from a quorum (i.e., a subset of $n - f$ nodes) are observable from any quorum. In the crash failure model, all $n - f$ responding nodes are correct, and thus $n > 2f$ (*the number of nodes in any two quorums must be greater than the total number of nodes in the system*, i.e., $(n - f) + (n - f) > n$) is enough to ensure intersection. In the Byzantine model, there might be faulty nodes in an accessed quorum sending wrong values; therefore $n > 3f$ (*the number of nodes in any two quorums must be greater than the total number of nodes in the system plus the maximum number of "liars,"* i.e., $(n - f) + (n - f) > n + f$) and quorums intersecting in at least $f + 1$ nodes.

6.1 Example: Raft

In Fig. 4, one can see that two quorums of three nodes intersecting at one node could be sufficient to ensure safety if nodes can only crash but would violate safety if the intersecting node misbehaves by responding that no new transaction *tx* has been written. Raft [51] is a CFT protocol currently employed in popular blockchains such as Corda [15] and Hyperledger Fabric [3]. In Raft, a leader coordinates the replicated execution by writing the next transaction to be processed onto a write quorum and then informs the client that its transaction is committed as depicted in the distributed execution of Fig. 4 where time increases from left to right. If, after this, the leader fails before committing this transaction in all servers and a new leader takes over, this new leader will read the pending transactions to ensure the same safety requirement of the algorithm: *that if some server committed this transaction, all servers will commit it in the same position on the transaction log*. If this new leader accesses a read quorum that intersects with the write quorum in a single server (as in the figure), and if this server is Byzantine and misbehaves, then it can lie about what happened (ignoring or replacing the transaction) and make the

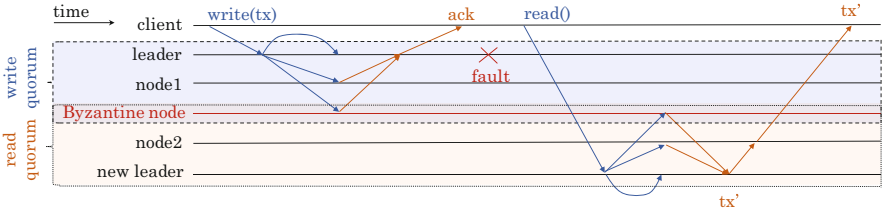


Fig. 4 A possible execution of a replicated write-then-read with a quorum size of $\lfloor n/2 \rfloor + 1$. We assume the quorum leaders are correct for simplicity. The execution would be correct in a CFT model but is incorrect in a BFT model. The intersection of BFT quorums should be larger than the intersection of CFT quorums for the consensus protocol to guarantee safety. If the intersection contains only Byzantine nodes, then the response obtained after reading from a quorum can violate safety, which can be exploited to double-spend in cryptocurrency applications

new leader commit a corrupted state, hence *violating safety*. This is precisely why, in contrast with Raft, BFT systems typically use quorums whose intersections contain at least $f + 1$ nodes.

7 Myth #5: A Blockchain with Signatures and Hashes Is Secure

An important misconception is that cryptography could solve the same problem as fault tolerance as illustrated by quantum-resilient efforts devoted on top of fundamentally insecure blockchains [20]. In particular, public-key cryptosystems are typically used in blockchains to ensure authentication: if a transaction is not signed by the owner of the account from which it tries to withdraw, then it is rejected—meaning that the signature is incorrect. Combined with hashes these signatures guarantee tamper evidence in that a non-authorized change will be easily detected as the hash of the signed transaction will not correspond to the observed signatures. These guarantees are not sufficient for security to ensure, for example, that a client is observing the right blockchain version. A malicious node could convince a client who acts as a merchant that a transaction buying its product is committed while it is not. To do so, the malicious node presents a fake version of the blockchain where it appends correctly signed transactions that it created without ever sending them to the actual blockchain.

This misconception is illustrated by some recent efforts devoted to develop quantum-resilient cryptographic software [20] on top of blockchain systems because these blockchains cannot tolerate the misbehavior of a single participant. In fact, quantum resilience aims at coping with the misbehavior of a quantum computer node to avoid asset theft, yet the misbehavior of a single node participating in the underlying CFT blockchain is sufficient to produce this theft, hence defeating the purpose of the quantum resilience.

7.1 Example: Raft

To guarantee that a client can truly identify whether its transaction is committed, it must make sure that $f + 1$ nodes say so. If not, it risks to be a victim of a double spending where the coins used in the supposedly committed transaction of the fake version are actually reused in a truly committed transaction of the blockchain. Notice that the use of signatures cannot help in the Raft scenario described in Myth #6. If the write operation of the leader is signed in Fig. 4, the Byzantine node on the intersection cannot create a different proposal (with tx') for the leader, but it can still hide it, stating it has not received any transaction at all. In this way, the transaction, despite being confirmed to the client, will be "undone" from the system history, opening up the possibility of double spending attacks that may be devastating for financial applications. More generally, there exist various attacks against blockchains that rely on delaying messages (e.g., selfish mining [32], eclipse attacks [39], BGP hijacking [5], and more generally man-in-the-middle attack [30]) that an attacker can exploit to double spend without violating authentication or forging private keys.

8 Myth #6: Adding a BFT Consensus to a CFT Blockchain Makes It BFT

Modularity is important for enabling project collaboration at the heart of open-source communities. Due to this, several blockchain platforms have been built with modular or replaceable components [3, 8]. Hyperledger Fabric separates the orderer service in charge of ordering transactions from the endorser service in charge of validating transactions to offer modularity [3]. In particular, Fabric offers several interchangeable orderers, a centralized service, a CFT one, and an experimental BFT one [57], hence illustrating this modularity.

A common mistake stems from extrapolating the guarantees offered by a module of a blockchain system to the whole blockchain system, for example, when using Corda in applications where a consortium of competitors do not trust each other[4] or when using a BFT orderer within Hyperledger Fabric, which tolerates only crashes. To illustrate this issue, let us consider a distributed ledger that tolerates only crash failures, like Corda or Hyperledger Fabric—this type of distributed ledgers can thus only run in a trustworthy environment as its guarantees would be violated if a participant misbehaves. A developer who want to make the blockchain service more robust may identify the consensus module as a module that is limited by its crash fault tolerance and be tempted to replace it by a more robust module offering Byzantine fault tolerance. If the blockchain system is modular enough, then the

[4] https://www.corda.net/samples/.

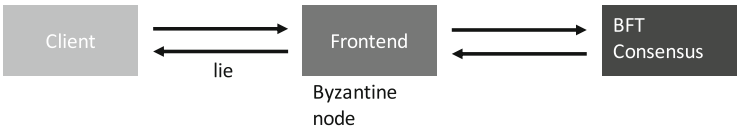


Fig. 5 A client contacting a single Byzantine front-end server makes the BFT consensus useless. This is why replacing one component of the software architecture is rarely sufficient to make a blockchain originally tolerant to crash failures, a blockchain that also tolerates Byzantine failures

developer expects the BFT consensus module to have the same interface as the CFT module. However, typical BFT services do not have the same interface as CFT services, precisely because the acknowledgment of a Byzantine server does not convey the same guarantee as the acknowledgment of a correct server. Figure 5 depicts the problem that might happen when the interface remains unchanged while the model changes from CFT to BFT: the client might be fooled by a flawed acknowledgment coming from a Byzantine front end while it could only receive correct servers in a CFT model.

8.1 Example: Hyperledger Fabric and BFT-SMaRt

Hyperledger Fabric [3] defines a modular architecture for supporting permissioned blockchains, which is used in production since version 1.0.[5] An important component of this architecture is the orderer service, a group of nodes responsible for ordering and packing transactions into blocks. At the time of writing, Fabric supports only CFT orderers: one based on Apache Kafka and another based on CoreOS' Raft implementation. In both cases, a single front end/server receives the transaction to be ordered and produces and signs the block (with ordered transactions) in the end. This design still tolerates crashes because there are redundant front ends in the system, and in case of failures, a backup front end takes over the role. However, in this model, even if we replace Raft by PBFT [21], the front end of the ordering service will still be receiving and signing the block. An experimental BFT orderer available for Fabric [57] changes this design to make the server create a block and sign it; however, it impacts modularity. In the end, the final generated block is signed by at least $f + 1$ servers. Yet, this change requires additional changes in other modules, say, to verify these $f + 1$ signatures. This shows that replacing a module was insufficient.

[5] https://www.fintechfutures.com/2017/03/ibm-cloud-launches-enterprise-ready-blockchain-for-hyperledger-fabric/.

9 Myth #7: BFT Consensus Needs a Linear Message Complexity

It is often the case that bad consensus performance at large scale is attributed to the all-to-all communication pattern of state-of-the-art consensus protocols [23, 63, 65]. In fact n servers sending messages to all other servers necessarily lead to a quadratic message complexity [13, 21, 22]. As a result, lots of efforts have been devoted to replacing all-to-all message exchanges by one-to-all exchanges [1, 7, 8, 42, 65]. The consensus protocol at the heart of Facebook's blockchain [65] is a typical case which uses threshold signatures to replace all-to-all by one-to-all communications. This result in a linear message complexity for the steady-state case. It appears that improving this message complexity does not necessarily lead to better performance. Several factors may limit the throughput of a replicated state machine before the message complexity becomes an issue. Some of these factors are the computational cost of cryptography (cf. Myth #11), the latency of writing new blocks to stable storage [12], the unique value decided per consensus instance (cf. Myth #12), and the use of a slow leader [4, 10].

9.1 Example: HotStuff vs. Democratic BFT (DBFT)

In Fig. 6, we compare the throughput of two systems, HotStuff [65] and DBFT [25], when the number of server increases on two different setups. On the "Europe" setup, a total of 64 or 128 servers run in 4 different AWS datacenters in Ireland, London, Paris, and Frankfurt. On the "World" setup, a total of 80 servers run in 4 different AWS datacenters in Ohio, Frankfurt, Sydney, and São Paulo. Each server is an AWS EC2 c5.xlarge instance with two hyperthreaded Intel Xeon 8124M cores for a total of four hardware threads. HotStuff has a linear message complexity, while DBFT has a quadratic message complexity. The DBFT replicated state machine does not use any specific optimization and is written in Java. HotStuff replicated state machine uses pipelining to reduce communication delays and is written in C++. We use transactions of 400 bytes, which is a typical Bitcoin transaction size. We observe that DBFT has a throughput up to 11× larger than HotStuff in the "Europe" setup and 16× larger in the "World" setup despite the fact that HotStuff has a better message complexity. We explain this difference by the fact that the HotStuff consensus is leader based and uses expensive threshold signatures while DBFT consensus is leaderless and its most expensive operation is a cryptographic hash. While we do not deny the impact of message complexity, we stress that other bottlenecks limit a consensus performance before message complexity becomes an issue.

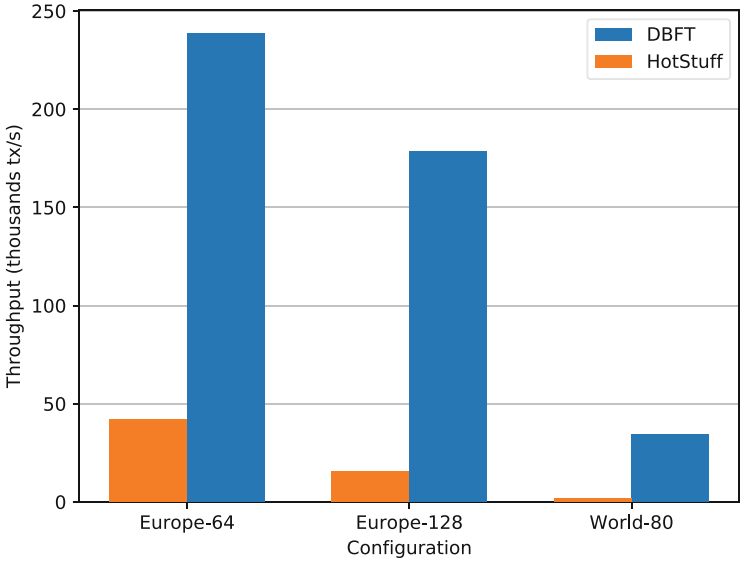


Fig. 6 Linear message complexity does not always bring good performance. The throughput of DBFT, which has a quadratic message complexity, is always higher than HotStuff, which has a linear message complexity in the steady state. This is because DBFT balances the network load on more links

10 Myth #8: Reconfiguring Consensus Participants Is Easy

Consortium blockchains rely on a globally known set of participants. This set of participants does not change while the consensus decides a new block. However, it is necessary or even desirable to change this set regularly between the consensus executions (e.g., to prevent bribery attacks). This operation, called reconfiguration or membership change, seems to be a built-in feature of some blockchains. For example, Tendermint design mentions validator set changes [58], but unfortunately this requires to shut down the blockchain. One of the challenges is purely related to software design as it is generally hard for a complex system to handle component hotplug [40, 47]. A more subtle issue is the possibility of blockchain forks when reconfiguration is employed, since participants already removed from the group can be compromised and cope with the ledger [12]. More generally, the challenge is for the system client to know where to send requests as the participant set changes over time. This last challenge is especially hard since a malicious front end could always present fake participant set in a scenario similar to Myth #8.

10.1 Examples: Hyperledger Fabric and Tendermint

Hyperledger Fabric aims at supporting membership changes without compromising the network; however, it still *"requires that the peer or orderer process is restarted."*[6] Tendermint mentions validator set changes [58]; however, this requires an external application that handles those reconfigurations.[7] BFT replication systems like BFT-SMaRt also require a trusted external application to manage such reconfigurations [13]. New implementations to reconfigure the Byzantine consensus participants rely on consensus protocol themselves and need a trusted domain name system to catch up with the latest configuration [61] and a forgetting protocol to ensure removed participants are unable to validate blocks from an invalid fork [12]. These new implementations make it possible for alive participants (i.e., participants being notified of configuration changes when they happen) to keep track of the system configuration. However, participants joining the system for the first time or catching up after being offline still have to rely on a permissioned trusted infrastructure that cannot be reconfigured, hence falling back on the initial challenge.

11 Myth #9: Blockchain Performance Is Not Limited by the Cryptography

Blockchain is believed to be bottlenecked by the large bandwidth consumed by its Byzantine consensus component. This is the reason why substantial efforts were devoted to reduce communication complexity at the price of increasing CPU overhead through the use of threshold signatures [65] (cf. Myth #9). There is, however, a CPU bottleneck induced by the cryptographic verification of transaction signatures inherent to blockchain systems [25, 59] that is often neglected. In particular, this CPU-intensive verification is necessary to ensure that a Byzantine node does not withdraw an amount of assets from the account of someone else. Each transaction has thus an associated signature that needs to be cryptographically verified. This verification limits the speed at which the blockchain can commit transactions.

11.1 Example: Red Belly Blockchain

Red Belly Blockchain [25] relies on verification sharding to minimize the number of verifications needed per transaction and to maximize its performance. The idea

[6] https://hyperledger-fabric.readthedocs.io/en/latest/msp.html.

[7] See Section 7.1, second paragraph, of https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf?sequence=7&isAllowed=y.

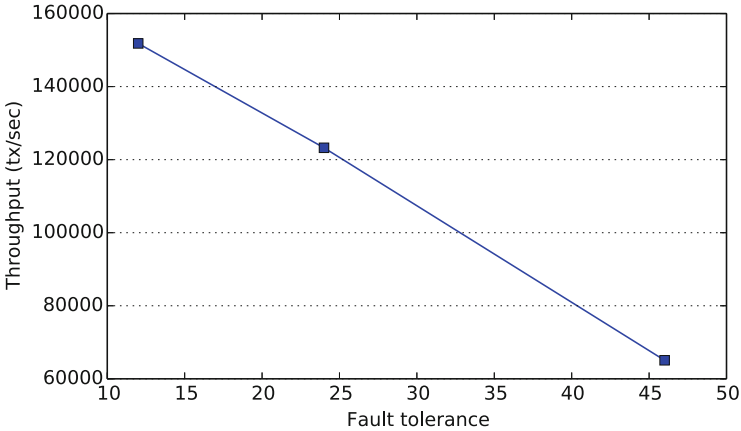


Fig. 7 The performance of Red Belly Blockchain decreases with the number of verifications of transaction signatures that is linear in the fault tolerance

is to have each transaction being verified by between $f + 1$ and $2f + 1$ nodes instead of all nodes in order to minimize CPU consumption. Figure 7 depicts the throughput (expressed in 400-byte transactions per second) of Red Belly when increasing the number of failures f on 140 c4.8xlarge AWS VMs geo-distributed on 4 continents (see details in [25]). The performance decreases linearly with the system fault tolerance f precisely because the number of verifications per node increases, illustrating the negative impact of verifications on the throughput of Red Belly Blockchain. Note that similar observations have been made in the context of Hyperledger Fabric [59] where the caching of verified endorsement certificates led to significant performance improvements.

12 Myth #10: Blockchain Needs to Solve the Classic Consensus Problem

A blockchain system needs a consensus protocol to totally order a series of blocks in order to guarantee that the blockchain it implements remains a chain. If disagreement happens, then the blockchain forks, making the system vulnerable to double spending. As participants can be incentivized to steal assets from others, most systems resort to traditional Byzantine consensus protocols [52], which ensure (1) agreement in that correct nodes cannot decide different values, (2) termination in that a correct node eventually decides, and (3) validity in that the decided value is one of the proposals. This is what is done in most blockchain systems that rely on a BFT consensus: Tendermint consensus [43] uses a variant of DSL [28], Facebook's Libra [8] uses HotStuff [65], and an orderer of Hyperledger Fabric uses BFT-SMaRt [13].

These protocols share the same goal of guaranteeing that only one proposal among all the proposals of the system gets decided. This means that if each server proposes $O(1)$ transactions to the consensus service, then the number of transactions decided by the consensus is $O(1)$, regardless of the number of servers participating in the protocol. This happens because they ensure this classic notion of consensus validity where the value decided is one of the proposed values. There is a variant of consensus for the Byzantine model that replaces this property. For instance, in interactive consistency [44], which is well suited for synchronous system but cannot be solved in a partially synchronous environment, the proposed values of all correct nodes are obtained in the end. Another definition is vector consensus [27, 50] that requires servers to decide the same vector containing at least $f + 1$ values proposed by different servers, where f is the maximum number of Byzantine failures. Unfortunately, in blockchain systems one cannot guarantee that $f+1$ values are valid, and thus, we would need to select the union of the valid transactions in each vector position. In particular, proposers (or miners) typically group transaction requests from client and cannot guarantee that all the received transactions are valid. An alternative to this problem is thus to relax the validity further to only accept values as long as they are deemed valid by the application (e.g., are correctly signed [13, 18]) but without any restrictions on their numbers [24, 25].

12.1 Example: Validity Predicate-Based Byzantine Consensus Problem

With this new consensus definition called validity predicate-based consensus [24], one can thus have distinct servers collecting transaction requests from the clients and then propose a batch of these transactions to a common consensus instance. If each of n servers proposes disjoint batches of $\Omega(1)$ transactions, and provided that the transactions are correctly signed, then the number of transactions that can be decided at the end of the consensus is $\Omega(n)$. This approach is currently implemented in the pipelined Byzantine state machine replication, called Dispel [62], and in Red Belly Blockchain [25] in the form of a Set Byzantine Consensus solution. As depicted in Fig. 8, the number of committed transactions per consensus instance is linear in the number of servers participating, and the throughput of the blockchain system grows with the number of servers (until exhaustion of some resources). This Byzantine consensus variant is especially designed for the geo-distributed environment of blockchains.

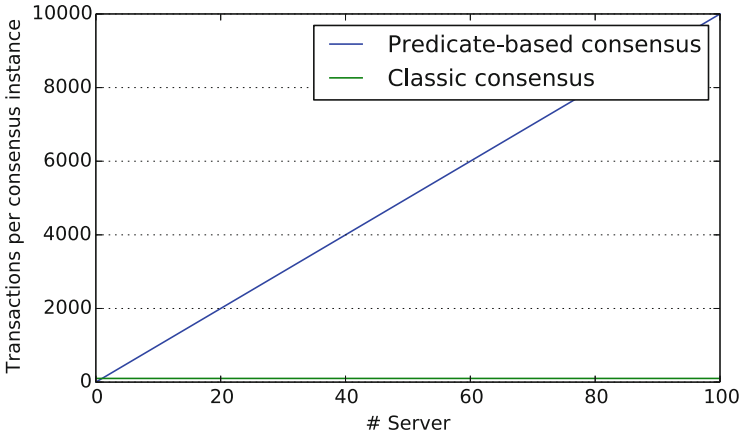


Fig. 8 A protocol solving the classic consensus definition would commit less transactions than one solving predicate-based validity consensus

13 Final Remarks

Related Work Although we are not aware of any work focusing on debunking blockchain myths, there are several research papers presenting evidence contradicting previous claims about blockchain consensus. Armknecht et al. [6] show that some assumptions listed in the Ripple consensus protocol white paper are not sufficient to ensure its safety. Cachin and Vukolić [19] explain that proof-of-work blockchains do not offer finality. Amoussou-Guenou et al. give counterexamples where Tendermint core does not solve consensus [2]. Saltini and Hyland-Wood [54] explain that the consensus at the heart of the Quorum blockchain does not terminate. Tholoniat and Gramoli [60] explain that the consensus at the heart of HoneyBadgerBFT does not terminate and argue in favor of formal verification. We ourselves rely on the correctness of consensus protocols to debunk these myths, but we have chosen the BFT-SMaRt library that has been publicly available for more than a decade [13] and Democratic BFT [24] that has recently been formally verified using parameterized model checking [11].

Prior to this work, various efforts were dedicated to evaluating the performance of distributed ledgers. BFT-Bench [37] predates blockchains and evaluates multiple BFT consensus algorithms. Blockbench [26] and [53] compare the performance of proof-of-work blockchains against private blockchains. Han et al. measures the number of participants at which the performance of blockchain drops [38]. Some evaluate proof-of-authority blockchains [45] that rely on a set of n authoritative validators among which at most f can be Byzantine to run the consensus. An evaluation focuses on comparing the performance of Byzantine fault-tolerant blockchains [56], whereas the recent DIABLO benchmark allows to measure blockchain performance under realistic workloads [9].

Conclusion We presented ten myths on blockchain consensus that represent misunderstandings about the performance and trust model of existing applications. Since consensus is a key element of blockchain and distributed ledger systems and is instrumental in the design of fault-tolerant replicated state machines, these misconceptions can lead to dramatic consequences, including asset losses in financial applications. In fact, several blockchains have been built to offer crash fault-tolerant consensus; however, adapting them to work in an adversarial context is not trivial. It is thus crucial to debunk these myths to strengthen the security of future blockchain-based applications to be put in production. We argue that it is possible for application designers to rethink blockchains with built-in Byzantine fault tolerance to offer reasonable performance after identifying the right bottlenecks.

References

1. M. Abd-El-Malek, G.R. Ganger, G.R. Goodson, M.K. Reiter, J.J. Wylie, Fault-scalable Byzantine fault-tolerant services, in *SOSP* (2005), pp. 59–74
2. Y. Amoussou-Guenou, A.D. Pozzo, M. Potop-Butucaru, S. Tucci Piergiovanni, Correctness of tendermint-core blockchains, in *OPODIS*, vol. 125 (2018), pp. 16:1–16:16
3. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Cocco, J. Yellick, Hyperledger fabric: a distributed operating system for permissioned blockchains, in *EuroSys* (2018)
4. K. Antoniadis, A. Desjardins, V. Gramoli, R. Guerraoui, I. Zablotchi, Leaderless consensus, in *ICDCS* (2021)
5. M. Apostolaki, A. Zohar, L. Vanbever, Hijacking bitcoin: routing attacks on cryptocurrencies, in *S&P* (2017), pp. 375–392
6. F. Armknecht, G.O. Karame, A. Mandal, F. Youssef, E. Zenner, Ripple: overview and outlook, in *Trust and Trustworthy Computing* (2015), pp. 163–180
7. P.L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, M. Vukolić, The next 700 BFT protocols. ACM Trans. Comput. Syst. **32**(4), 12:1–12:45 (2015)
8. S. Bano, M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, A. Sonnino, State machine replication in the Libra blockchain (2019). https://developers.libra.org/docs/assets/papers/libra-consensus-state-machine-replication-in-the-libra-blockchain.pdf. Accessed 10 2019
9. H. Benoit, V. Gramoli, R. Guerraoui, C. Natoli, Diablo: a distributed analytical blockchain benchmark framework focusing on real-world workloads. Tech. Rep. 285731, EPFL (2021)
10. C. Berger, H.P. Reiser, J. Sousa, A. Bessani, AWARE: adaptive wide-area replication for fast and resilient byzantine consensus. IEEE Trans. Depend. Secure Comput. **19**(3), 1605–1620 (2020). Early access
11. N. Bertrand, V. Gramoli, I. Konnov, M. Lazic, P. Tholoniat, J. Widder, Compositional verification of byzantine consensus. Tech. Rep. 03158911, HAL (2021)
12. A. Bessani, E. Alchieri, J.A. Sousa, A. Oliveira, F. Pedone, From byzantine replication to blockchain: consensus is only the beginning, in *DSN* (2020)
13. A. Bessani, J. Sousa, E.E.P. Alchieri, State machine replication for the masses with BFT-SMART, in *DSN* (2014), pp. 355–362
14. G. Bracha, An asynchronous $[(n-1)/3]$-resilient consensus protocol, in *PODC* (1984), pp. 154–162
15. R.G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: an introduction. R3 CEV (2016)

16. V. Buterin, V. Griffith, Casper the friendly finality gadget. Tech. Rep. 1710.09437v4, arXiv (2019)
17. Hyperledger caliper benchmarks (2020). https://www.hyperledger.org/use/caliper
18. C. Cachin, K. Kursawe, F. Petzold, V. Shoup, Secure and efficient asynchronous broadcast protocols, in *CRYPTO* (2001), pp. 524–541
19. C. Cachin, M. Vukolić, Blockchain consensus protocols in the wild (keynote talk), in *DISC* (2017), pp. 1:1–1:16
20. R. Campbell, Transitioning to a Hyperledger Fabric quantum-resistant classical hybrid public key infrastructure. The Journal of The British Blockchain Association (2019)
21. M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002)
22. A. Clement, E. Wong, L. Alvisi, M. Dahlin, M. Marchetti, Making Byzantine fault tolerant systems tolerate Byzantine faults, in *NSDI* (2009), pp. 153–168
23. P.R. Coelho, T.C. Junior, A. Bessani, F.L. Dotti, F. Pedone, Byzantine fault-tolerant atomic multicast, in *DSN* (2018), pp. 39–50
24. T. Crain, V. Gramoli, M. Larrea, M. Raynal, DBFT: efficient leaderless Byzantine consensus and its applications to blockchains, in *NCA* (2018), pp. 1–8
25. T. Crain, C. Natoli, V. Gramoli, Red belly: a secure, fair and scalable open blockchain, in *S&P* (2021)
26. T.T.A. Dinh, J. Wang, G. Chen, R. Liu, B.C. Ooi, K.L. Tan, Blockbench: a framework for analyzing private blockchains, in *SIGMOD* (2017), pp. 1085–1100
27. A. Doudou, A. Schiper, Muteness detectors for consensus with byzantine processes, in *PODC* (1998), p. 315
28. C. Dwork, N. Lynch, L. Stockmeyer, Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988)
29. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail, in *CRYPTO*, vol. 740 (1993)
30. P. Ekparinya, V. Gramoli, G. Jourjon, Impact of man-in-the-middle attacks on Ethereum, in *SRDS* (2018), pp. 11–20
31. P. Ekparinya, V. Gramoli, G. Jourjon, The attack of the clones against proof-of-authority, in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'20)* (2020)
32. I. Eyal, E.G. Sirer, Majority is not enough: bitcoin mining is vulnerable, in *International Conference on Financial Cryptography and Data Security*. Springer, Berlin (2014), pp. 436–454
33. V. Gramoli, On the danger of private blockchains, in *DCCL* (2016)
34. V. Gramoli, From blockchain consensus back to byzantine consensus. Fut. Gener. Comput. Syst. **107**, 760–769 (2020)
35. V. Gramoli, Blockchain scalability and its foundations in distributed systems (2021). Coursera MOOC
36. V. Gramoli, M. Staples, Blockchain standard: can we reach consensus? IEEE Commun. Stand. Mag. **2**(3), 16–21 (2018)
37. D. Gupta, L. Perronne, S. Bouchenak, BFT-Bench: a framework to evaluate BFT protocols, in *ACM/SPEC ICPE* (2016), pp. 109–112
38. R. Han, G. Shapiro, V. Gramoli, X. Xu, On the performance of distributed ledgers for internet of things. Internet of Things **10** (2020)
39. E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on bitcoin's peer-to-peer network, in *USENIX Security* (2015), pp. 129–144
40. B. Kemme, A. Bartoli, O. Babaoglu, Online reconfiguration in replicated databases based on group communication, in *DSN*. IEEE (2001), pp. 117–126
41. P. Khahulin, I. Barinov, V. Baranov, POA network white paper (2018). https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper
42. R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva: speculative Byzantine fault tolerance, in *SOSP* (2007), pp. 45–58
43. J. Kwon, Tendermint: Consensus without mining (2014)

44. L. Lamport, R. Shostak, M. Pease, The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
45. F. Leal, A.E. Chis, H. González-Vélez, Performance evaluation of private ethereum networks. SN Comput. Sci. **1**(285) (2020)
46. H. Moniz, The Istanbul BFT consensus algorithm. Tech. Rep. 2002.03613, arXiv (2020)
47. Z. Mwaikambo, A. Raj, R. Russell, J. Schopp, S. Vaddagiri, Linux kernel hotplug CPU support, in *Linux Symposium*, vol. 2 (2004)
48. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system (2008)
49. C. Natoli, J. Yu, V. Gramoli, P. Esteves-Verissimo, Deconstructing blockchains: a comprehensive survey on consensus, membership and structure. Tech. Rep. 1908.08316, arXiv (2019)
50. N.F. Neves, M. Correia, P. Verissimo, Solving vector consensus with a wormhole. IEEE Trans. Parallel Distrib. Syst. **16**(12), 1120–1131 (2005)
51. D. Ongaro, J.K. Ousterhout, In search of an understandable consensus algorithm, in *USENIX Annual Technical Conference* (2014), pp. 305–319
52. M.C. Pease, R.E. Shostak, L. Lamport, Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
53. S. Pongnumkul, C. Siripanpornchana, S. Thajchayapong, Performance analysis of private blockchain platforms in varying workloads, in *ICCCN* (2017), pp. 1–6
54. R. Saltini, D. Hyland-Wood, Correctness analysis of IBFT. Tech. Rep. 1901.07160, arXiv (2019)
55. D. Schwartz, N. Youngs, A. Britto, The Ripple consensus protocol (2014). https://ripple.com/files/ripple_consensus_whitepaper.pdf
56. G. Shapiro, C. Natoli, V. Gramoli, The performance of Byzantine fault tolerant blockchains, in *NCA* (2020)
57. J.A. Sousa, A. Bessani, M. Vukolić, A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform, in *DSN* (2018)
58. Tendermint 0.10.2 (2017). White paper. https://jepsen.io/analyses/tendermint-0-10-2.pdf
59. P. Thakkar, S. Nathan, B. Vishwanathan, Performance benchmarking and optimizing Hyperledger Fabric blockchain platform, in *MASCOTS* (2018)
60. P. Tholoniat, V. Gramoli, Formal verification of blockchain byzantine fault tolerance, in *FRIDA* (2019)
61. G. Vizier, V. Gramoli, ComChain: a blockchain with Byzantine fault tolerant reconfiguration. Concurrency Comput. Pract. Experience **32**(12), e5494 (2019)
62. G. Voron, V. Gramoli, DISPEL: byzantine SMR with distributed pipelining. Tech. Rep. 1912.10367, arXiv (2019)
63. M. Vukolic, The quest for scalable blockchain fabric: proof-of-work vs. BFT replication, in *IFIP WG 11.4 International Workshop on Open Problems in Network Security* (2015), pp. 112–125
64. I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A.B. Tran, P. Rimba, On availability for blockchain-based systems, in *SRDS* (2017), pp. 64–73
65. M. Yin, D. Malkhi, M.K. Reiter, G.G. Gueta, I. Abraham, HotStuff: BFT consensus with linearity and responsiveness, in *PODC* (2019), pp. 347–356
66. P. Zheng, Z. Zheng, X. Luo, X. Chen, X. Liu, A detailed and real-time performance monitoring framework for blockchain systems, in *ICSE-SEIP* (2018), pp. 134–143

Cryptographic Primitives

Mayank Raikwar and Shuang Wu

The underlying design fundamental of blockchain is cryptography. As blockchain operates in a decentralized peer-to-peer network, therefore to safeguard the transaction information along with maintaining transaction integrity, cryptography plays a vital role to fulfill all these requirements. The primary focus of cryptography in the blockchain is to make blockchain secure, reliable, and efficient by ensuring the security of user transactions, participants and safeguard against different attacks such as double-spending. Accordingly, many cryptographic primitives have been applied in blockchain to ensure these properties.

Apart from primarily focusing on these above properties, cryptography provides more application in blockchain, e.g., in the verification of asset transfer, achieving anonymity of participants. Thus, there has been a tremendous amount of work on the applicability of different cryptographic primitives in the blockchain. Since the advent of blockchain, many studies scrutinize the use of cryptographic concepts in the blockchain. Many cryptographic primitives have already been well-studied, exhibited, and employed in blockchain; nevertheless, a few primitives are still left to be investigated more.

Blockchain is promising, powerful, and still a growing technology. However, it still encounters many research challenges. Some of the important challenges are scalability, key management, protection against different attacks, smart contract management, and further improvements on security and privacy in blockchain designs. These challenges emerge due to the underlying consensus mechanism,

This work was done in part while the authors were at NTNU, Norway.

M. Raikwar (✉) · S. Wu
University of Oslo, Oslo, Norway

DNV, Trondheim, Norway
e-mail: mayankr@ifi.uio.no; Shuang.wu@dnv.com

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_2

network infrastructure, and participants' behavior. Therefore, to overcome these challenges and to achieve the proper functioning of blockchain, many cryptographic primitives can be investigated, scrutinized, and applied in the blockchain.

In this chapter, we present a brief description of cryptographic primitives employed in blockchain. For each cryptographic primitive, we provide the definition, properties, and their use in blockchain domain. Furthermore, we also postulate research problems which can be of independent interest.

1 Core Cryptographic Primitives in Blockchain

1.1 Hash Function

Hash functions are extremely used in most cryptographic protocols and security applications. It is used as a mathematical compression function. The output of the mathematical function is called a message digest or hash value. Hash functions are vital to ensure the immutability and integrity of data stored in blockchain. Each block of the blockchain contains a hash of the block header of the previous block, ensuring the immutability of the blockchain. Hash functions are also widely used in constructing cryptographic puzzles and signature message digest in the blockchain.

Definition 1 A hash function H is an algorithm which maps an arbitrary size input to a fixed size output (Fig. 1).

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

A hash function should satisfy few security criteria:

3. *Preimage resistance* implies that hash function is not invertible, meaning given an output y, it is hard to find an input x such that $H(x) = y$.
3. *Second preimage resistance* infers that given an input x, it is hard to find x' where $x \neq x'$, such that $H(x) = H(x')$.

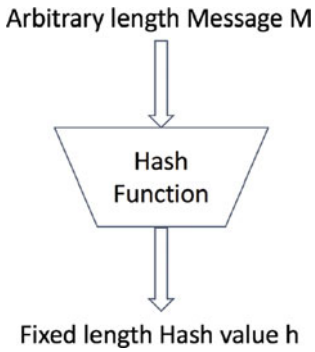


Fig. 1 Block diagram of hash function

3. *Collision resistance* implies that it is hard to find a pair of input x, x' mapping to the same hash value, which means $H(x) = H(x')$.

Many standard hash functions are mathematically defined and analyzed and fulfill the security criteria. Moreover, the theory of the hash function has been well-studied in the past. Different hash functions fulfill different purposes such as transaction verification, address generation, finding cryptographic puzzle solutions (mining), or message digest in signatures. According to Narayanan et al. [1], assuming a random value r is sampled from high minimum entropy distribution, a hash function H used in blockchain should have two properties:

2. *Hiding:* Given hash value $H(x||r)$, if it is computationally hard to retrieve the value x, then the hash function H is hiding.
2. *Puzzle-friendliness:* Given the n bit output y as $H(x||r) = y$, if it is not feasible to find such x in significantly less than 2^n time, then the hash function H is puzzle-friendly.

A family of cryptographic hash functions named SHA (Secure Hash Algorithms) gives the two most popular hash algorithms SHA-2 and SHA-3 used in the blockchain. SHA-2 hash functions are built using Merkle-Damgard construction. A variant of SHA-2 named SHA256 is the most popular hash function used in many blockchains including Bitcoin [2]. In many of the blockchain designs, the hash functions are used in the form of a combination of several invocations of the same or different hash functions. In the case of Bitcoin, SHA256 is combined twice and called a double SHA256 or SHA256d. To implement SHA256d and to speed up the mining process, many application-specific integrated circuit hardware (ASIC miner) were introduced which were faster in huge order of magnitude than ordinary CPUs and GPUs. With the invention of ASIC miners and further, the centralization of mining became a crucial problem in the decentralized blockchain field. Therefore, in 2011, Litecoin [3] (an earlier fork of Bitcoin) introduced Scrypt, which asymptotically requires more memory than the one used in Bitcoin, making it memory-intensive and impractical to implement in ASIC, hence an ASIC-resistant hash function. Now, Scrypt is used in many proof-of-work-based cryptocurrencies.

SHA family member, SHA-3, which is a subset of the broader cryptographic primitive family Keccak [4] designed using sponge construction is also used in blockchain. Keccak256, an alias to SHA-3, is used in Ethereum hash function Ethash [5]. Many memory-hard and ASIC-resistant hash functions such as Ethash, Equihash [6], X11 [7], and CryptoNight [8] are designed to fight the development and centralization of mining techniques.

- Ethash, a hash function in Ethereum blockchain, uses Keccak256 and Keccak512 which are variants of SHA-3. In Ethash, due to the large size of DAG, it requires external memory.
- Equihash, hash function in Zcash [9] and Bitcoin Gold blockchain, uses Blake2b hash function for the construction. The underlying problem is to solve the "generalized birthday problem" over the generated hash dataset. The best-known

algorithm to solve the Equihash puzzle is Wagner's algorithm which runs in exponential time and is memory-intensive.
- X11, hash function in Darkcoin [10], uses 11 hash functions from SHA-3 candidates in its construction. These 11 hash functions are pipelined in a fixed sequence. X11 is an extension of the hash function of QuarkCoin which was the cascade of six hash functions.
- CryptoNight, hash function in Monero [8], is based on AES encryption and five hash functions (finalists in NIST SHA-3 competition: Keccak, BLAKE, JH, Grøestl, and Skein). The hash output of CryptoNight is expensive to verify.

Although the concept of ASIC-resistant is a prominent thing in hash functions and the above hash functions claim to be ASIC-resistant, there are commercially available ASIC miners for each of the hash functions. Therefore, it leaves friction between the cryptocurrency community and ASIC miners. The concern about the insecurity of hash functions has already been raised among researchers. With this concern, a study about time-memory trade-off attacks on Argon2 hash function was presented by Dinur and Nadler [11]; Heilman et al. also reported practical attacks on IOTA blockchain hash function Curl-P-27 [12].

Research Problem Study and assess the available lightweight hash functions with respect to security, privacy, and feasibility. Find a suitable candidate to be employed in IoT blockchain. □

Research Problem Construct a new lightweight hash function specifically tailored for blockchain IoT nodes. □

1.2 Signature Scheme

A signature scheme (digital signature) is an inevitable cryptographic primitive in the blockchain. The concept of digital signature came forward from public key cryptography. Digital signatures are used to verify the authenticity of digital messages and documents. Signatures are widely used in different applications such as contract signings, software distribution, and financial transactions.

Signatures are a fundamental building block in blockchains. Signatures are primarily used to verify the authenticity of blockchain transactions. By providing a signature over a transaction, a user proves that it is authorized to spend the fund of the transaction while preventing other users to spend those funds.

Definition 2 A signature scheme consists of three algorithms:

3. KeyGen(1^n). A key generation algorithm generates private key and corresponding public key pair (sk, pk) by using the input 1^n.
3. Sign(m, sk). A signing algorithm takes a message m and the secret key sk and produces a signature σ as output.

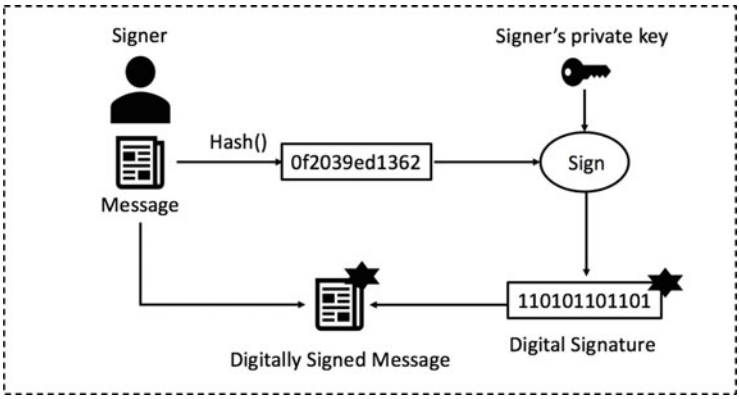


Fig. 2 Signing process

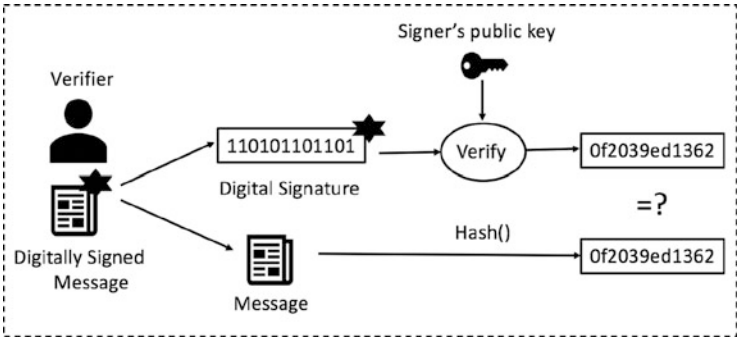


Fig. 3 Verification process

3. Verify(m, σ, pk). Given a message m, signature σ, and a public key pk, a verification algorithm outputs accept or reject.

In short, a signature scheme produces short codes using the private key of a user; these codes are called signatures and can be verified by using the public key of the user. The signing and verification processes are depicted in Figs. 2 and 3, respectively. The primary motive of digital signature in the blockchain is to sign a transaction, hence ensuring the source authentication and source non-repudiation along with establishing the integrity of the transaction. The security of digital signature is given as existential unforgeability against adaptively chosen messages attacks (EUF-CMA), which states that an adversary having access to signing oracle cannot forge a valid signature on a new message of his choice.

With the rapid development of blockchain technology, signature schemes have gained immense attention in real-world applications and academic research. Many new signature scheme constructions were also proposed in the past which affects the efficiency and characteristics of blockchain. In the general context, various signature

schemes are used to provide integrity, privacy, and unlinkability in the blockchain network. The most used signature schemes in blockchain are ECDSA (Elliptic Curve Digital Signature Algorithm [13]) and EdDSA (Edwards-curve Digital Signature Algorithm [14]). The security of both signatures lies under the hardness assumption of the elliptic curve version of the discreet logarithm problem. ECDSA is built on general elliptic curve cryptography and is used in many blockchains including Bitcoin and Ethereum. However, EdDSA works on a variant of Schnorr signature [15] based on twisted Edwards curves and used in many blockchains such as Monero [8] and Libra [16]. EdDSA provides better performance and security than ECDSA in many ways: (1) EdDSA has smaller public keys and signature size, (2) EdDSA does not use the unique random number for each signature compared to ECDSA, and (3) EdDSA is more resilient to side-channel attacks than ECDSA.

There are other signature schemes that are also used in different blockchains to increase the privacy and anonymity in the blockchain. Some of these schemes are as follows:

- *Ring Signature* The notion of ring signature was formalized in [17]. The idea of a ring signature is to create a signature on a message on behalf of a spontaneous group of signers while preserving the identity of the signer. Additionally, the creation of the signature does not require cooperation among the group members. Besides the existential unforgeability of the signature scheme, the ring signature also requires an extra security requirement and unconditional anonymity. Unconditional anonymity states that the identity of the signer is preserved and two signatures signed by the same signer cannot be linked. Later, a property called linkability was proposed in the ring signature scheme, and the overall scheme was referred to as *linkable spontaneous anonymous group* which is essentially *linkable ring signature*. With the linkable property, one can detect whether the two signatures are created by the same signer while still preserving anonymity. In line with modification to the ring signature scheme, a scheme of *traceable ring signature* was proposed which provides the possibility to trace the signer of the two messages. A modified version of traceable ring signature was first used in electronic cash system "CryptoNote" [18]. Later some attacks were found on CryptoNote based on the analysis of transaction amounts. Hence, to hide the transaction amount, Maxwell proposed the idea of confidential transactions [19]. Afterward, the idea of confidential transaction coupled with multilayered linkable ring signature was introduced in Monero blockchain which was already using the ring signature scheme for the protection against the privacy of the signing party in the input transaction. The new scheme was called Ring Confidential Transactions for Monero (RingCT) [20].
- *Multi-signature* The idea of multi-signature was first introduced in [21]. It allows a group of n signers (each having its private/public key pair) to jointly produce a single short signature σ on a message m. This short signature σ convinces a verifier that all n signers signed the message m. Given the message m and set of public keys of all the n signers, the verification of σ can be publicly performed. For practical purposes, the size of the produced signature in a multi-signature

scheme should be close to the ordinary signature size and independent of the number of signers in the scheme. While dealing with multi-signature, the rogue-key attack must be taken into account. In a rogue-key attack, first, an adversary crafts public keys which are correlated with the public keys of honest signing parties, and then using these crafted public keys, the adversary forges a multi-signature on a message of his choice. The security against a rogue-key attack is achieved by requiring proof of possession (knowledge) of the corresponding secret key from each signer of the scheme.

Maxwell et al. constructed a Schnorr-based multi-signature scheme [22] called MuSig which was built on the work of Bellare and Neven [23] and provably secure in the plain public key model (where users do not need to prove the knowledge of their secret key). The MuSig supports public key aggregation where the verifier only requires an aggregated public key from all the public keys, instead of an explicit list of all public keys. They also analyzed and simulated the impact of MuSig in the Bitcoin blockchain to save storage space. However, the aggregation in the MuSig is a multi-round process and can only take place at the time of signing. Hence, Boneh et al. constructed a new multi-signature scheme [24] which is useful not only for reducing Bitcoin blockchain size but also for use in other settings where multi-signatures are needed. Their scheme is pairing-based multi-signatures derived from BLS signature scheme [25]. It supports public key aggregation and security in a plain public key model.

- *Blind Signature* The phenomenon of blind signature was introduced by David Chaum [26]. In a blind signature scheme, the message is disguised (blinded) before signing. The message is blinded by combining it with some blinding factor. These signatures are employed in privacy-related protocols, especially where the sender privacy is prominent. Blind signatures are mostly used in the electronic voting system and digital cash systems. Blind signatures can be used in different applications to provide anonymity and unlinkability. There have been quite many works done with the use of blind signatures in the blockchain. A distributed mixing network, BlindCoin [27], uses the blind signature to provide the unlinkability of transactions. Another blind mixing scheme to strengthen the anonymity in the Bitcoin system has been proposed and demonstrated in [28]. A work [29] using a blind signature for the anonymity of Bitcoin provides a solution for the anonymity not only of on-chain transactions but also of off-chain transactions (micropayment channel network transactions). A privacy-preserving fair contract signing scheme [30] has been proposed to protect the privacy content related to the contract. The construction was built using the blind signature scheme combined with verifiable signatures.

  Blind signature together with the blockchain has been quite applied with the electronic voting (E-voting) system. E-voting has been an interesting research topic in cryptography for the last few decades. In an E-voting system, an electronic system is used to aid casting and vote counting. There is much existing work now on E-voting protocol implementation using blockchain and blind signature. Some of the interesting works are [31–33].

- *Threshold Signature* It is a distributed multiparty signature protocol for distributed key generation, signing, and verification algorithms. In this scheme, there are n participants, and each participant gets a share of a secret key, and t out of n parties create a signature from the available shares. The full share of the key never gets revealed in the entire scheme, so the secret key remains secure throughout the execution of the scheme. In the scheme, the information of the signers remains secret as the constructed threshold signature is verified by the same public key. Hence, threshold signatures can be used to provide anonymity in the blockchain. Threshold signature scheme has a flexible access structure of keys, and it is a more efficient scheme as it does not require an extra fee in the verification process and it does not require any additional support for the on-chain smart contracts. The use of threshold signature in blockchain brings a new paradigm to enhance the resilience, robustness, and security of the system while preserving the distributed nature of blockchain.
  A study on the use of threshold signatures on the blockchain has been performed by Stathakopoulou and Cachin [34] which shows the integration of threshold RSA and threshold BLS signature with Hyperledger Fabric [35] blockchain. Threshold signatures are also used in multiparty mixing of Bitcoins [36] and on-chain mixing in Ethereum [37]. Threshold signature is also a good candidate for the key management solutions. It is being used in cryptocurrency wallets [38] and other key management systems. The cryptocurrency wallets based on threshold signatures are more secure as the private key does not need to be rebuilt.

Research Problem Study the (im)possibility of constant-size aggregate signature scheme from general groups to reduce the blockchain storage. □

Research Problem Construct an efficient post-quantum hash-based signature scheme for the blockchain use. □

1.3 Encryption Scheme

Encryption is a mechanism to provide confidentiality of data. It is a process of converting a representation of information (plaintext) into another representation (ciphertext) so that only authorized parties can access the original information from the ciphertext representation. In an encryption scheme, a sender S encrypts a plaintext using a key and sends the output ciphertext to receiver R; then the receiver decrypts the ciphertext using a key.

In the blockchain, encryption plays an important building block to achieve privacy of data. Encryption schemes are used in blockchain to hide transaction data such as transaction value or asset information. Encryption is also used to provide privacy of users' account balances.

Definition 3 An encryption scheme involves three algorithms:

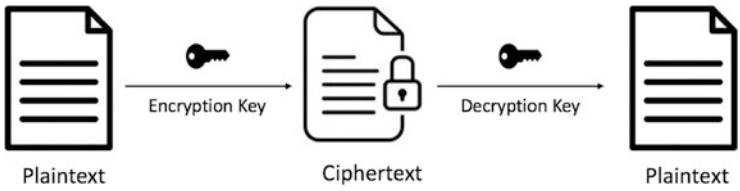


Fig. 4 Encryption scheme

- KeyGen(κ): On input a security parameter κ, the algorithm generates an encryption key ek and a decryption dk.
- Enc(M, ek): On input a plaintext M and an encryption key ek, encryption algorithm produces a ciphertext C.
- Dec(C, dk): On input a ciphertext C and a decryption key dk, decryption algorithm outputs a message M.

An encryption scheme has correctness if for all M, $\text{Dec}(dk, \text{Enc}(M, ek)) = M$. The correctness does not ensure security of the encryption scheme.

Figure 4 depicts a general encryption scheme. However based on the encryption and decryption keys, encryption schemes are categorized in two:

1. Symmetric key encryption, where encryption and decryption keys are same and secret to the parties.
2. Asymmetric/public key encryption, where encryption key ek is a public key pk and decryption key dk is a secret key sk. Keys (pk, sk) belong to the receiver R.

Symmetric key encryption is primarily used for the privacy and confidentiality of data. Within this encryption scheme, the key (in fact secret) must be known to both the sender S and receiver R. However, the main difficulty with this scheme is the key distribution. Symmetric key encryption schemes are generally categorized as block or stream ciphers. Asymmetric/public key encryption scheme is mostly used in practice. It is used for non-repudiation, authentication, and key exchange.

Privacy-preserving smart contract model, Hawk [39], achieves privacy of transaction data using encryption and stores encrypted transactions on the blockchain along with proof of correctness. Another privacy-preserving smart contract model, Zether [40], accomplishes transaction confidentiality by encrypting the user account balances using the ElGamal encryption scheme. On-ledger encryption is also an application of encryption scheme in blockchain [41]. In on-ledger encryption, a user's public key is stored in the metadata of one of her addresses, and later other users can use this public key to encrypt information created for the public key user and storing the ciphertext in the transaction. Encryption is also employed in constructing random beacon protocols [42, 43] which are substantially used to get randomness in blockchain consensus protocols. Proof of delay [44] uses symmetric key encryption algorithm to iterate a delay function on the top of block hash and subsequently produces the randomness.

In the majority of applications, encryption has been applied to achieve the confidentiality of blockchain data. Hyperledger fabric applies symmetric key encryption to achieve confidentiality of smart contract [35]. Ethereum [5] also uses Elliptic Curve Integrated Encryption Scheme (ECIES) in its transport protocol for inter-node communication.

There has been emerging interest in the techniques to compute or search over encrypted data. These techniques are homomorphic encryption, searchable encryption, and functional encryption. Monero [8] uses half additive homomorphic encryption and supports value transactions. Searchable encryption and functional encryption have also been coupled with blockchain in different applications [45–47]. Other forms of encryption providing extra functionalities such as broadcast encryption and attribute-based encryption have also been investigated and applied in the blockchain context [48, 49].

Research Problem Find the applicability of symmetric key encryption in different context of blockchain ecosystem. □

Research Problem Study the harness of implementing homomorphic encryption for privacy-preserving blockchain-based outsourced storage system. □

1.4 Accumulator

A cryptographic accumulator [50] commits to a collection of values as a succinct digest. These values can be a vector, a set, or a multiset. The digest is computationally infeasible to equivocate about the collection represented by the digest. Nevertheless, an accumulator can be defined as a one-way function that gives a membership proof without revealing individual identity in the underlying set. An accumulator can be used to build other cryptographic primitives such as commitment, ring signatures, and zero-knowledge proofs.

The primary use of an accumulator in the blockchain is to prove the membership of transactions within a block by an element which is obtained by committing to a set of transactions using the accumulator. The other important usage of accumulators in blockchain is accumulating state updates or batching balance/coin commitments.

Definition 4 (Accumulator) Let $\mathcal{M}$, $\mathcal{D}$, and $\mathcal{P}$ be the message set, the digest set, and the proof set respectively. A (strong) accumulator consists of four PPT algorithms, $\mathsf{Acc} = (\mathsf{Gen}, \mathsf{Proof}, \mathsf{Dig}, \mathsf{Ver})$, that are defined as follows:

- $\mathsf{Gen}(1^k)$. Take input as a security parameter k and output a public key pk.
- $\mathsf{Proof}_{\mathsf{pk}}(m, S)$. Take input as a message m and a message set S; if $m \in S$ and $S \subseteq \mathcal{M}$, then it outputs a proof of membership $p \in \mathcal{P}$. Otherwise output fail.
- $\mathsf{Dig}_{\mathsf{pk}}(S)$. Take input as a message set S; if $S \subseteq \mathcal{M}$, then it outputs a digest $d \in \mathcal{D}$; otherwise output fail.

- $\mathsf{Ver}_{\mathsf{pk}}(m, d, p)$. Take input as a message m, a digest d, and a member proof p, if $m \in \mathbb{M}$, $d \in \mathcal{D}$ and $p \in \mathcal{P}$, output either Member or NotMember, otherwise output fail.

- **Correctness:** For a valid public key $pk \leftarrow \mathsf{Gen}(1^{\mathsf{k}})$, $m \in \mathcal{M}$, and $S \subseteq \mathcal{M}$, $\mathsf{Ver}(m, \mathsf{Dig}_{\mathsf{pk}}(\mathsf{S}), \mathsf{Proof}_{\mathsf{pk}}(m, S))$ should output Member if $m \in S$.
- **Security in Trusted Setup:** Let $\mathsf{Acc} = (\mathsf{Gen}, \mathsf{Proof}, \mathsf{Dig}, \mathsf{Ver})$ be an accumulator. In trusted setup model, Acc is collision-resistant if for any PPT adversary $\mathcal{A}$

$$\Pr\left[\begin{array}{c}\mathsf{pk} \leftarrow \mathsf{Gen}(1^{\mathsf{k}}), (m, S, p) \leftarrow \mathcal{A}(\mathsf{pk}) : \\ m \notin S \wedge \mathsf{Ver}_{\mathsf{pk}}(m, \mathsf{Dig}_{\mathsf{pk}}(S), p) = \mathsf{Member}\end{array}\right] = \mathsf{negl}(k)$$

Acc is undeniable if

$$\Pr\left[\begin{array}{c}\mathsf{pk} \leftarrow \mathsf{Gen}(1^{\mathsf{k}}), (m, S, p, \bar{p}) \leftarrow \mathcal{A}(\mathsf{pk}) : \\ \mathsf{Ver}_{\mathsf{pk}}(m, d, p) = \mathsf{Member} \wedge \mathsf{Ver}_{\mathsf{pk}}(m, d, \bar{p}) = \mathsf{NotMember}\end{array}\right] = \mathsf{negl}(k)$$

If a party wants to prove that an element v is in a given set S or not while given the constant-size accumulator A for the values of set S, the accumulator works by proving that a membership witness w when combined with value v results in accumulator A. An accumulator should fulfill the security requirements which are as follows:

- *One-wayness* guarantees that the witness w for the value v cannot reveal any information about the members in the set.
- *Indistinguishability* ensures that, given a witness w for value $v \in S_0 \cup S_1$, no one can tell whether v is in S_0 or S_1.
- *Collision resistance* assures that one cannot generate a membership witness for a nonmember of a set S.
- *Undeniability* ensures that one cannot generate both membership and nonmembership proof for the same element v.

An accumulator can be dynamic if elements can be added and removed efficiently from the set; otherwise, the accumulator is static. A universal accumulator is dynamic and it supports both membership and nonmembership proofs. Accumulators can be categorized into the following three categories where each one offers trade-offs between accumulator size, setup parameters, witness, and proof performance.

1. *Unknown-order-based accumulators* These accumulators are subdivided into RSA-based and hyperelliptic curve-based accumulators. RSA-based accumulators afford the most efficient trade-off between the number of setup parameters, updating witness requirements, proof computation, and succinctness. RSA accumulators are constant size (a single group element) and witness updates are cleanly segregated from the other set elements. Li et al. [51] proposed the

first universal accumulator, but their scheme requires a trusted setup. Later on, different constructions [52, 53] are proposed to remove the trusted setup and improve efficiency.

2. *Pairing-based accumulators* These accumulators are based on pairings. Nguyen [54] proposed the first pairing-based accumulator scheme which can be used to build an identity-based ring signature scheme, but this accumulator does not provide the nonmembership proof. Therefore, a series of new constructions [55, 56] are introduced to make it dynamic, universal, or both. Later, another interesting scheme [57] was introduced which delegates the nonmembership proofs without the limitation on the number of accumulated elements.
3. *Hash-based accumulators* The hash-based accumulator was proposed in 1996 by Nyberg [58]. These accumulators are trapdoor-free and more efficient than RSA-based accumulators. Several undeniable universal dynamic accumulators were proposed [59, 60] which make the use of hash function and hash trees but with a trusted setup. Thus, later to solve this problem, many strong constructions were proposed [61, 62].

Merkle tree (depicted in Fig. 5) is the most commonly used accumulator in cryptocurrencies due to its space- and time-efficient data structure for set membership. The most common usability of Merkle tree accumulator is to commit to a collection of transactions or data to only one single Merkle tree root. Bitcoin and Ethereum both use this technique so that the blocks only contain the Merkle tree root of the transactions/states. Zerocoin uses the RSA accumulator to batch the coins; later a user gives proof to spend a coin that is accumulated in the accumulator. Accumulators are used in [63] to design a stateless blockchain where to participate in consensus, the node only needs a constant amount of storage. Another interesting work [64] shows verifiable outsourcing of state updates in the blockchain using a set accumulator.

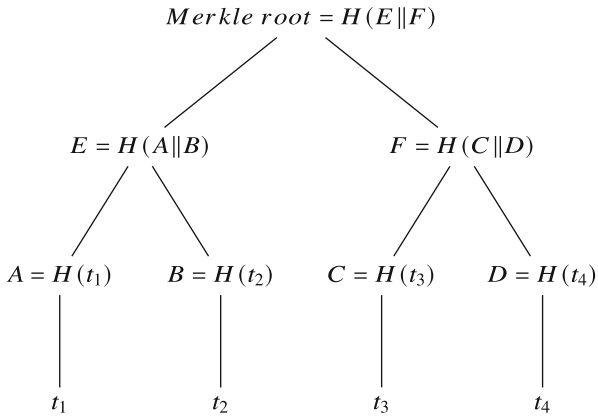


Fig. 5 Merkle tree of transactions

Research Problem Construct an accumulator based on class group that supports aggregation of nonmembership proofs and does not require trusted setup. □

Research Problem Build a practically efficient quantum-resistant accumulator with constant-size witness. □

1.5 Commitment Scheme

A commitment scheme can be defined as a digital-analog of a sealed envelope. The concept of a commitment scheme was formalized in 1988 by Gilles Brassard, David Chaum, and Claude Crépeau [65]. A commitment scheme provides an excellent way to hide a value while maintaining its proof of commitment.

A commitment scheme provides an excellent way of hiding transaction data in blockchain while maintaining its proof of commitment. Commitment schemes are used in blockchain for coin value commitment or smart contract state commitment. An example of commitment scheme is a blind auction in blockchain where participant commits to their bid and gives a proof of their commitments.

Definition 5 Commitment scheme is a two-phase process between two parties, sender S and receiver R. The two phases are called as *commit* and *reveal*.

1. In commit phase, the sender S chooses a value x; then a commitment $Com(x, r)$ of the value x is sent to receiver R.
2. In reveal phase, the sender S opens the committed value $Com(x, r)$ and the receiver R checks that the sender did not cheat.

Reveal phase is also referred as open phase in some context. These two phases possess two additional properties:

- *Hiding* The value chosen in the first phase cannot be known to the receiver until the reveal phase.
- *Binding* Sender cannot alter the value after the commit phase.

To understand the commitment scheme as depicted in Fig. 6, the following example illustrates the binding commitment:

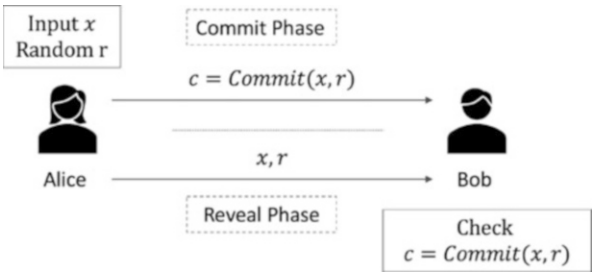


Fig. 6 An illustration of commitment scheme

- Sender S chooses a secret value x to commit from 0 to $q-1$ where q is a large prime.
- Sender S computes a commitment C as $C = g^x mod q$.
- Sender S gives the value C to receiver R.

Here, the binding properties follow as any other party can verify the committed value C if x is given and it is infeasible to compute another value y generating the same C. Moreover, given the values C, g, and q, finding the value x is a computationally hard problem of the discrete logarithm.

The commitment scheme is an important building block for other cryptographic primitive such as zero-knowledge proofs and verifiable secret sharing. It is also used as a building block in different blockchain applications. The following are a few commitment schemes which are used in blockchain:

- *Pedersen Commitment* The commonly used commitment scheme in the blockchain is Pedersen commitment [66]. Pedersen commitment provides computational binding and unconditional hiding properties based on the discrete logarithm problem. Given a group $\mathbb{G}$ of order q, two generators $g, h \xleftarrow{\$} \mathbb{G}$, a Pedersen commitment to a value $x \in \mathbb{Z}_q$ is computed as

$$C_x = g^x h^r, \text{ where } r \xleftarrow{\$} \mathbb{Z}_q$$

  Pedersen commitment is used in Zerocoin [67] to bind a serial number s to a Zerocoin z by constructing the commitment as $C_s = g^s h^z$. Pedersen commitment is also used to commit the stake values in the private version of proof-of-stake blockchains [68]. Pedersen commitment scheme has been applied in different applications such as verifiable sealed-bid auction in Ethereum blockchain [69] and in the construction of RingCT 2.0 [70]. The variants of Pedersen commitments such as Pedersen vector commitment and elliptic curve Pedersen commitment are also used in blockchain.
- *Polynomial Commitment* In a polynomial commitment scheme, the committer commits to a polynomial with a short string, and the verifier confirms the claimed evaluation of the committed polynomial using the short string. A polynomial commitment can be considered as a sort of hash of some polynomial $P(x)$ having the functionality of performing arithmetic checks on hashes, for example, given three polynomials $P(x), Q(x), R(x)$ and their respective polynomial commitment $h_P = Commit(P(x)), h_Q = Commit(Q(x)), h_R = Commit(R(x))$. With the property of the polynomial commitment scheme, it is possible to generate a proof against the hashes h_P, h_Q, h_R for the arithmetic operations on given polynomials $P(x), Q(x), R(x)$. The arithmetic operations can be as follows:
  - $P(x) + Q(x) = R(x)$
  - $P(x) * Q(x) = R(x)$

Polynomial commitments can be used as vector commitments similar to the Merkle tree. Most of the blockchains use the Merkle tree to accumulate the transaction data of a block. These Merkle roots of block data can be replaced with polynomial commitments, and the tree branches can be replaced with openings which makes it easier to perform data availability checks. Recently, Tomescu et al. [71] formalize a new aggregatable subvector commitment which aggregates multiple proofs into a single, small subvector proof. Their construction is based on Kate et al. polynomial commitment [72] and has linear-size public parameters. They proposed a payments-only efficient stateless cryptocurrency based on their updatable and aggregatable commitment scheme. Another concurrent work by Gorbunov et al. [73] also constructs a new commitment scheme called Pointproofs for noninteractive aggregation of proofs across multiple commitments. In Pointproofs, they aggregate the opening proofs for different commitments and used them in blockchain to reduce the bandwidth overhead for the block propagation. They also build their construction using polynomial commitment. Moreover, they demonstrated how to apply the Pointproofs on blockchain and smart contracts.

- *Functional Commitment* In a noninteractive functional commitment scheme, the committer commits to a message M in such a way that in the opening phase, committer reveals a function of the message $F(M)$. The scheme was first proposed by Libert et al. [74]. They constructed a functional commitment scheme for linear functions based on the well-researched assumptions on composite-order bilinear groups. The construction is defined as follows: a message is a vector of n elements and defined as $\mathbf{m} = (m_1, \ldots, m_n) \in \mathcal{D}^n$, where each element is from a domain $\mathcal{D}$. Then later the commitments can be opened to a specific linear function defined as $F_{\mathbf{x}}(\mathbf{m}) = \langle \mathbf{x}, \mathbf{m} \rangle = \sum_{i=1}^{n} x_i m_i$. Fixing a specific $\mathbf{x} \in \mathcal{D}^n$, such that $F_{\mathbf{x}}(\mathbf{m}) = y \in \mathcal{D}$, then an opening for the function indeed evaluates to y.
  Matteo et al. [75] proposed a scheme for vector commitments with subvector openings that are based on a variant of the functional commitment scheme. They also gave a construction of verifiable decentralized storage where a file is stored efficiently in a decentralized network and the integrity of the file is preserved.

Research Problem Construct a vector commitment scheme with short parameters that has incrementally aggregatable subvector openings and updatability notion and can be employed to construct verifiable decentralized storage. □

Research Problem Find suitable aggregatable vector commitment schemes for stateless cryptocurrencies, and subsequently evaluate the performance enhancement in these cryptocurrencies when applying these commitment schemes. □

1.6 Secret Sharing

A secret sharing scheme provides a method to split a secret s among a group of participants P and to reconstruct the secret from the sufficient number of

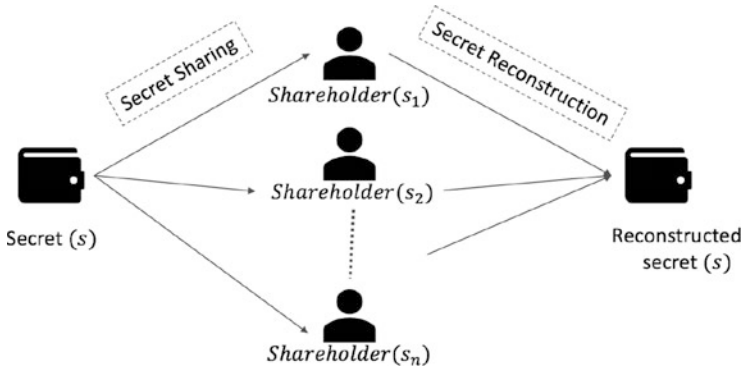


Fig. 7 t-out-of-n secret sharing scheme

participants S of the group, where $S \subseteq P$. In one type of secret sharing scheme, a *dealer* distributes the shares to the participants. A secret sharing scheme provides an efficient and secure way to store a secret across multiple servers (participants) and remain recoverable despite multiple server failures. Secret sharing schemes are the ideal mechanism to store highly sensitive and highly important information. It is an interesting and useful tool in many cryptographic systems (notable in secure computation).

In the blockchain, a secret sharing scheme is used to store sensitive information. Distributed storage makes use of a secret sharing scheme to store secret information in the blockchain. A publicly verifiable variant of a secret sharing scheme is a useful component in building randomness beacon protocols [42, 76], and the random output from these protocols is being used in blockchain consensus.

Definition 6 (t-Out-of-n Secret Sharing) A t-out-of-n secret sharing scheme (Fig. 7) can be defined as a pair of algorithms (Share, Reconstruct) designed over message space $\mathcal{M}$:

- *Share* It is a randomized algorithm that takes an input message (secret) $m \in \mathcal{M}$ and generates $n-$tuple of shares $(s_1, \ldots, s_n)$.
- *Reconstruct* It is a deterministic algorithm that takes an input as $t-$tuple of shares $(s_{i_1}, \ldots, s_{i_t})$ and outputs a message in $\mathcal{M}$.

And it satisfies the following requirement for the correctness of the scheme:
$\forall m \in \mathcal{M}, \forall S = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ of size t,

$$\underset{Share(m) \rightarrow (s_1, \ldots, s_n)}{Pr} [Reconstruct(s_{i_1}, \ldots, s_{i_t}) = m] = 1$$

Security of a secret sharing scheme is based on the statement that anyone having fewer than t shares has no more information about the secret than someone having no share at all.

The secret sharing scheme is invented independently by Adi Shamir and George Blakley in 1979. Shamir's secret sharing is a t-out-of-n secret sharing scheme, so any t out of n shares can be used to reconstruct the secret. The main idea of Shamir's secret sharing scheme is to create a degree $t-1$ polynomial with the first coefficient as secret and the remaining coefficients as randomly picked values. Next, the n points out of the polynomial are constructed and given as shares to n participants. The idea behind the method is that a unique $t-1$ degree polynomial can be fit to any set of t points on the polynomial. However, Blakley's scheme is based on hyperplanes. A secret is specified as a point in n-dimensional space and the share of the secret corresponds to hyperplanes that intersect the secret point. The idea behind the method is any n nonparallel $(n-1)$-dimensional hyperplanes intersect at a specific point. Shares in Blakley's scheme are t times larger than Shamir's schemes; hence it is less space-efficient.

There are a few other efficient secret sharing schemes. As the shares are stored on players' computers, there is always a threat for the compromise. Therefore, the shares should be updated periodically such that an attacker has less time to compromise the shares. This method is called a proactive secret sharing (PSS) scheme where the shares are renewed using a new random polynomial with the constant term as zero. Another issue in the secret sharing scheme is players can lie about their shares to gain access to other shares. Hence, there should be a method to check the consistency of the shares. The scheme to perform the check on consistency is called a verifiable secret sharing (VSS) scheme which ensures that the players can always reconstruct the secret even if the dealer is malicious. Another variant of VSS is PVSS which is publicly verifiable secret sharing scheme where an outside party can also verify the outcome of the scheme. PVSS is quite used in constructing random beacon protocol that provides randomness for blockchain [77, 78]. Another interesting construction, high-threshold asynchronous verifiable secret sharing (HAVSS), is also used to produce randomness [79].

In the blockchain, a secret sharing scheme can be used in different ways. A secret sharing scheme can be applied for private data protection. The private data can be a secret, personal data, or wallet private key. Therefore, by using secret sharing, private data can be stored as different shares with different participants. A secret sharing scheme has been used to protect personal data [80] and to store a secret in blockchain [81]. It can also be applied in decentralized autonomous organizations (DAO) which can be practiced in consensus to store the full system state as a set of shares of the system state among participating nodes. As the nature of the secret sharing scheme is to distribute the shares, so it can also be used in distributed storage [82]. The scheme can also be used in different applications such as E-voting [83] or E-health [84] for storing and sharing sensitive data in the blockchain.

Research Problem Design a secure transaction data dissemination protocol among blockchain peers using secret sharing scheme. □

1.7 Multiparty Computation

The multiparty computation (MPC) scheme is introduced by Andrew Yao in the 1980s. It is an important tool for privacy-preserving distributed applications. A secure multiparty computation (SMPC) scheme allows the parties to jointly compute a function over their inputs while keeping their inputs private at the same time. Therefore, a multiparty protocol has a few players, each having certain input. These inputs are either the player's share of a common secret value or the player's input to the computation. Each party produces certain outputs, either the shares of a common value to be used in further computation or results from the computation.

MPC is a promising primitive to enhance the security and privacy of blockchain. MPC provides fairness, correctness, enhanced privacy, and guaranteed output delivery in blockchain solutions. MPC can be used to protect the private key of blockchain wallets by sharding the key and reconstructing it dynamically to sign blockchain transactions. In some scenarios, a high-value transaction involving multiple parties or confidential transactions can be executed using MPC; one way can be to process the transaction off-chain using MPC and upload the transaction receipt on-chain as proof.

The theory of MPC flourished in the 1980s, but the implementation of MPC for real-world application is developed in the last decade. Moreover, there are many unsolved challenges that exist for the optimized SMPC implementation in terms of computation, scalability, bandwidth, and latency.

Definition 7 In an MPC, there are n parties $P_1, P_2, \ldots, P_n$ each holding a private input value $x_1, x_2, \ldots, x_n$, respectively, and there is a n-ary function F. A secure multiparty computation for F is a joint protocol between parties for computing output $y = F(x_1, x_2, \ldots, x_n)$ while keeping their own input private.

The multiparty computation protocol ensures two basic properties:

1. *Input privacy*: The information/share about the private data held by the parties cannot be inferred from the communication during the execution of the protocol. That means party P_i cannot get an other input $x_j (j \neq i)$.
2. *Correctness*: Honest parties should either compute the correct output y or abort if they find an error, after the execution of the protocol.

Information-theoretic secure MPC protocol is built on to the secret sharing problem and specifically verifiable secret sharing (VSS). The security of the MPC protocol depends on different assumptions. These assumptions can be any computationally hard problem or synchronous network among parties or the existence of a secure channel, etc.

A noninteractive multiparty computation (NI-MPC) model deals with the group of asynchronous parties which evaluate a function over their input by sending a single message to an evaluator [85]. The evaluator further computes the output. Nevertheless many of the existing NI-MPC models cannot provide full privacy for inputs, because of the inherent leakage of residual function. A recent NI-MPC

protocol, Gage MPC [86], bypasses the residual function leakage and also provides a proof of concept for auction functionality over Ethereum blockchain.

In the context of blockchain, SMPC has been used in a few systems to ensure strong privacy. Andrychowicz et al. [87] designed SMPC protocols for multiparty fair lotteries on the Bitcoin system in 2014. In their protocol, if a user interferes with the protocol, the user is forced to pay a monetary penalty. A decentralized platform, Enigma, was proposed by Zyskind et al. in 2015. [80] leverage the SMPC concept. Enigma encodes shared secret data using a modified distributed hash table. In the Enigma platform, a blockchain network containing the hashes is combined with an SMPC network containing the data corresponding to those hashes. Enigma employs a verifiable secret sharing scheme to achieve the strong privacy of its computational model.

A secure smart contract system is built on SMPC over the blockchain [88]. The system achieves privacy protection for the parties' inputs and smart contract data. The proposed system also guarantees the correct execution of the contract even if the adversary corrupts several participants. SMPC can also be exercised in blockchain model Hawk [39] to minimize the trust in the generation of common reference string in SNARK proof used in the model or to design private data storage systems such as Keep [89] or to do the cross-chain network transfer in Wanchain blockchain.

MPC can be applied in a variety of applications for enhancing privacy. It can be used in cryptocurrency exchanges to safely store users' assets and at the same time providing an improved user experience. MPC can also be leveraged in services such as cloud service providers or banks to provide security of users' data and end-to-end visibility.

Research Problem Find the areas in blockchain ecosystem where MPC (especially NI-MPC) can improve the privacy in a cost-effective manner. □

1.8 Zero-Knowledge Proofs

Zero-knowledge proof [90] is a technique where a prover asserts some statement and proves the validity of the statement to a verifier without revealing the actual content of the statement except the statement.

Zero-knowledge proofs are applied in blockchain to strengthen privacy and security and also to achieve scalability. From the early introduction of zero-knowledge proofs in Zerocash [91] blockchain for untraceable and unlinkable confidential transactions, there has been a tremendous effort in the applicability of zero-knowledge proofs in the blockchain. Zero-knowledge proofs have been efficiently applied in many different scenarios in blockchain such as transmitting private or off-chain transactions, storage protection, and consensus mechanism.

Let R be an efficient computable binary relation that has a pair of variables (x, w) where x is a statement and w is a witness. Let $\mathcal{L}$ be the language defined by R, i.e., $\mathcal{L} = \{x | \exists w s.t. R(x, w) = 1\}$. A zero-knowledge proof for language $\mathcal{L}$ is a pair of

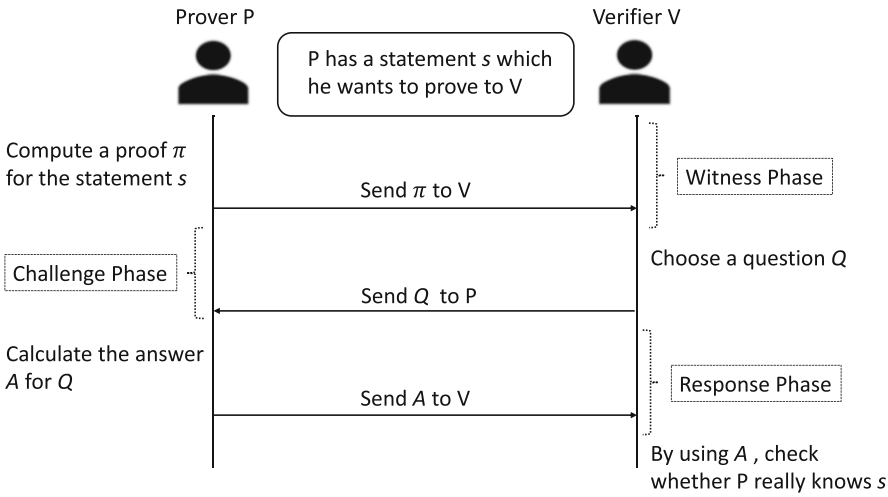


Fig. 8 An interactive zero-knowledge protocol

prover P and verifier V where prover wants to convince the verifier that $x \in \mathcal{L}$ without revealing w. A valid proof demonstrates not only $x \in \mathcal{L}$ but also prover holds the corresponding witness w. The validation of the proof is determined based on some public parameter, the message, and the verifier's input. Zero-knowledge proof operates on the following three basic tenets:

- *Completeness* Prover P with witness w for statement $x \in \mathcal{L}$ can convince the verifier V that P possesses the correct input.
- *Soundness* Prover P cannot convince the verifier V when statement $x \notin \mathcal{L}$ which means a false statement cannot be proved.
- *Zero Knowledge* Verifier V cannot learn anything (particularly witness w), except the statement x, which means the statement does not leak any information apart from the truth of the statement.

Zero-knowledge proofs are mainly of two types: (1) *interactive* where the prover performs a series of actions to convince the verifier about the validity of the statement as depicted in Fig. 8 and (2) *non-interactive* where the prover performs all the actions (challenges) at once and sends the proof to the verifier and later the verifier checks for the validity of the statement through the proof. Apart from these two types of proofs, the following definitions provide a broad spectrum of zero-knowledge proofs.

Definition 8 (Noninteractive Zero-Knowledge Argument [92]) A *NIZK* argument for a NP relation R includes three polynomial-time algorithms (Setup, Prove, Verify) defined as follows:

- $\mathsf{Setup}(1^\kappa)$ *takes as input the security parameter* κ *and outputs the common reference* σ.

- $\mathsf{Prove}(\sigma, x, w)$ *takes as input the common reference* σ*, a statement x, and a witness w for the statement and outputs an argument* π.
- $\mathsf{Verify}(\sigma, \pi, x)$ *takes as input the common reference* σ*, a statement x, and an argument* π *for the statement and outputs 1 if the verifier accepts the argument; otherwise it outputs* 0 *rejecting the argument.*

Definition 9 (Noninteractive Zero-Knowledge Argument of Knowledge) A *NIZK* argument of knowledge for an NP relation R is a NIZK argument for R with an additional knowledge extractor that can compute a witness whenever the adversary produces a valid argument. The extractor gets full access to the adversary's state, including any random coins. Formally, we require that for all nonuniform polynomial-time adversaries $\mathcal{A}$, there exists a nonuniform polynomial-time extractor Ext such that

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\Sigma, \hat{s}, \hat{\pi}) = 1 \\ \wedge R(\hat{s}, w') = 0\end{array}\ \begin{array}{c}\Sigma \leftarrow \mathsf{Setup}(1^k) \\ : (\hat{s}, \hat{\pi}) = \mathcal{A}(1^k, \Sigma; r) \\ w' = \mathsf{Ext}(\Sigma, \hat{s}, \hat{\pi}; r)\end{array}\right] = \mathsf{negl}(k)$$

Definition 10 (Zero-Knowledge Succinct Noninteractive Argument of Knowledge (zk-SNARK)) A zk-SNARK for a relation R is a noninteractive zero-knowledge argument of knowledge with additional requirement that the proof size is succinct to the size of statement and witness.

One common approach to build zero-knowledge proof is using a common reference model where some special structured parameters are generated during the trust setup phase. The protocols based on trusted setup have the drawback that they cannot achieve a complete trust-less system. We describe the existing zero-knowledge protocols in the literature into three categories based on the requirement for the trusted setup and how the trusted setup is performed.

- *Trusted setup per computation:* The zero-knowledge protocols under this category require a trusted setup every time the circuit is changed. For arithmetic circuit satisfaction, the most efficient construction is by Groth [93], which is implemented in *Zcash* project. The algorithm is based on *Quadratic Arithmetic Programs* [94] that represents an arithmetic circuit in a polynomial format. It achieves a constant size of proof that only contains three group elements, with the prover's proving time $O(N \log N)$ to the circuit size N. An adopted approach to do the trusted setup is by multiparty computation where the security of the system can hold as long as one participant destroys its secret. However, the multiparty computation on large scale alone has many unsolved challenges, and it can never achieve a completely trust-less thread model. Along with Groth's algorithm, these zk-SNARK [94–96] constructions also have the trusted setup downside.
- *Universal zk-SNARKs:* To reach the completely trust-less (or transparent) thread model, the most tricky part is that removing the trust setup will decrease the efficiency of the protocol. These trade-offs between trust level and efficiency lead to another line of work on universal zk-SNARK system [96–100], which aims to

reach a middle ground to eliminate the need for trust setup phase per computation challenge. For universal zk-SNARKs, a trusted setup is done once for a particular computation class that has a certain bound on the number of its operations. On the other hand, universal zk-SNARKs still keep maintaining the succinctness and efficient verification guarantees provided by efficient zk-SNARK constructions.
There are two variants of universal zk-SNARK systems presented in the literature. The first universality is respect to the common reference, which means a common reference can be adapted by multiple circuits. The second universality is relevant to the circuit itself, in which a circuit receives the proof that is going to be verified as part of its input, and processes its verification. The second category of universality is also known as recursive zk-SNARKs.
Under the first category, Groth et al. introduced a universal zk-SNARK with updatable common reference strings [97], while the common reference in the construction is quadratic to the size of the circuit making it hard to be practical. Another zk-SNARK called LegoUAC [98] has linear universal CRS but the proof size is polylogarithmic. On the other hand, Sonic provides an updatable zk-SNARK with constant-size proofs [99].

- *Transparent setup:* There are some constructions proposed to remove the need for a trusted setup. This work line includes (1) Bulletproofs, which rely on discrete log-based techniques and in which the trusted setup gets removed but the proof size is logarithmic to the circuit size, and (2) zk-STARKs [101], Ligero [102], and Aurora [103] which rely on symmetric cryptography and are supposed to have post-quantum security.

Research Problem Construct an efficient zk-SNARK proof composition technique that can achieve privacy and scalability in blockchain solutions with reduced cost. □

Research Problem Construct a post-quantum zero-knowledge range proof system that optimizes the proof size or produces a constant-size proof irrespective of the secret size. □

2 Allied Cryptographic Primitives in Blockchain

This section describes those cryptographic primitives which open a broad spectrum of research direction in the field of blockchain. Some of these primitives have gained a lot of attention due to their usability and applicability for improving the security and privacy of blockchain. A few of the following primitives have not been much explored for their usability and applications in the blockchain domain. Nevertheless, we also demonstrate those primitives and their usefulness in the blockchain.

2.1 Verifiable Random Function

A verifiable random function (VRF) is a pseudorandom function that produces pseudorandom output along with proof about the correctness of the output. The output of a VRF is resistant to preimage attacks and can be published publicly.

The primary application of VRF in the blockchain is to produce a decentralized random beacon [104] whose output is unpredictable until it becomes available to everyone. This output is used in leader election (lottery protocol) [105] or to solve the problem with true randomness in blockchain space.

Definition 11 A VRF consists of three algorithms:

- KeyGen(r): On input value r, the algorithm generates a secret key sk and a verification key vk.
- Eval(sk, M): Evaluation algorithm produces pseudorandom output O and the corresponding proof π on input sk and a message M.
- Verify(vk, M, O, π): Verify algorithm outputs 1 if and only if the output produced by evaluation algorithm is O and it is verified by the proof π given the verification key vk and the message M.

Loosely speaking, a VRF is an asymmetric keyed hash function, where, given a message and secret key, it produces an output along with a proof, which is verified using the corresponding public key. More precisely, the output of the VRF is calculated by hashing the proof, making the output indistinguishable from a random value.

Security Properties of VRF A VRF ensures the following security properties:

- *Uniqueness*: It ensures that for any input M and for a fixed public key of VRF vk, there is a unique VRF output O that can be proved valid using vk.
- *Collision resistance*: VRF should be collision resistant which means that an output of the VRF O should not be generated from different inputs M_1, M_2.
- *Pseudorandomness*: It ensures that VRF output O should be indistinguishable from a random value in a view of adversarial verifier without seeing the proof π.
- *Unpredictability*: Pseudorandomness does not hold if the VRF keys are generated adversarially. Therefore, a different type of unpredictability property is desirable in certain VRF applications. This property states that if the input has enough entropy (unpredictable), then the correct output is indistinguishable from uniform, similar to an ordinary hash function.

There are RSA and elliptic curve VRF variants and some other lesser-known VRF approaches. Another interesting construction of VRF [106], DDH-VRF, is based on DDH assumption, where, along with output of VRF, a zero-knowledge proof of correctness is also provided. Recently, a distributed version of DDH-VRF, DVRF [104], is presented that provides strong pseudorandomness but non-compact proof of correctness. Therefore, another distributed variant of VRF, GLOW-VRF [104], is constructed that achieves strong pseudorandomness with compact proof size. These distributed variants of VRF [104, 107] are used to construct

random beacon protocols. There is also recent advancement to construct post-quantum VRFs [108, 109] that can be very useful in post-quantum blockchain constructions.

VRF has been introduced over 20 years ago [110], but, recently, VRF became quite popular due to its usefulness in blockchain applications. Since the introduction of VRF, it has been employed in many useful applications in protocol design. For example, VRFs have been used to construct one-time sound resettable zero-knowledge protocol [111], to construct noninteractive lottery in micropayments [112]. VRF has many use cases such as blind auctions, DNS denial of existence, and cryptographic sortition. Many blockchain platforms such as Algorand [105], Ouroboros [113], and Coda [114] started using cryptographic sortition in their consensus mechanism which brought VRF into the limelight. Chainlink [115] employs VRF to generate on-chain randomness using a smart contract. Moreover, VRF has also been written in Solidity to be used in the Ethereum blockchain.

Cryptographic sortition offers a very attractive alternative approach to resource waste in consensus mechanisms. In the sortition method, each participant can self-identify himself for participation in consensus and block finalization. Each participant privately calculates their unique output and proof on the existing information for the block (such as round, seed, etc.) using their secret keys. Then based on the consensus-specific rule on the output value (e.g., has specific trailing digits, perhaps less than a threshold value), leader candidates are selected to propose the new blocks in the blockchain. The further consensus generally involves counting votes on blocks from consensus participants and reaching an agreement based on the number of votes. Further, the use of VRF in sortition requires no communication overhead and consumes significantly less energy and computational power from the participants.

Research Problem Find the applicability of (noninteractive) fully distributed VRF (DVRF) to construct an efficient leader election protocol in the blockchain consensus. □

2.2 Verifiable Delay Function

A verifiable delay function (VDF) is a recent cryptographic primitive and an important tool to add a delay in decentralized applications. It is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ defined formally by Boneh et al. [116] that takes a prescribed minimum number of steps to compute and is exponentially easy to verify. That means the VDF cannot be parallelized. The output of the VDF is unique and sound; hence an adversary has a negligible chance of randomly guessing the correct output.

VDF is also used in blockchain to generate on-chain randomness or to elect a leader in blockchain consensus. It is also used as a proxy for the passage of time in decentralized applications. Additionally, VDF can also be incorporated as client

puzzles [117] to solve the problem of denial of service attacks in the blockchain ecosystem.

Definition 12 A VDF is defined as a tuple of following algorithms:

- Setup(λ, T): It is a randomized algorithm that takes security parameter λ and time parameter T and outputs public parameter pp.
- Eval(pp, x, T): The evaluation algorithm takes public parameter pp, input value $x \in \mathcal{X}$, and time parameter T and returns an output value $y \in \mathcal{Y}$ together with a proof π. The algorithm may use random coins to generate the proof π but not for the computation of output y.
- Verify(pp, x, y, π, T): The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

A VDF has the following three properties:

- ϵ-evaluation time: The evaluation algorithm of VDF runs in at most $(1 + \epsilon)T$ time.
- Sequentiality: An algorithm using at most $poly(\lambda)$ parallel processors cannot compute the function output in less than time T.
- Uniqueness: Evaluation algorithm should produce exactly one output value y for an input value x which can be verified using the verification algorithm.

After the introduction of VDF by Boneh et al. [116], VDF has gained attention from the academic and industrial community, resulting in follow-up constructions proposed recently. An active work regarding the implementation of fast VDFs is taking good shape as well. VDF brings a large range of applications; especially VDF is a good candidate for a random beacon that can be used in blockchain to generate randomness in the consensus mechanism.

The two proposals on VDF from Wesolowski [118] and Pietrzak [119] use a group of unknown order to construct their VDF. A group of unknown orders has a property where exponentiation cannot be sped up by parallelism. There are recent constructions defining variants of VDFs:

- Continuous Verifiable Delay Function (cVDF): A function f is called continuous verifiable delay function if (1) $f^{(t)}$ can be evaluated in roughly t times to the time to evaluate on some random input (iterative sequential) f and (2) the output of $f^{(t)}$ is efficiently verifiable.
- Tight Verifiable Delay Function: A tight verifiable delay function is a verifiable delay function where the evaluation takes not much more time than the sequential bound T.

Apart from random beacon as an application, VDF can be used in different promising applications such as proof of replication, resource-efficient blockchains, and computational time-stamping in proof-of-stake consensus systems. Research toward constructing efficient random beacon protocols using VDF has been rising up, and a number of constructions have been proposed [120, 121]. VDF is also

quite explored to construct Nakamoto-style consensus mechanisms [122, 123]. An interesting direction of research is exploring the VDF to design client puzzles for DoS prevention [124].

Research Problem Find a post-quantum secure simple VDF for its use in blockchain. □

Research Problem Construct a cost-effective noninteractive client puzzle using VDF. □

2.3 Pseudorandom Function

Pseudorandom functions (PRFs) are efficiently computable functions (computable in polynomial time) that output pseudorandom values indistinguishable from random sequences. Thus, pseudorandom functions emulate a random oracle.

Definition 13 A pseudorandom function (PRF) $X \times K \rightarrow Y$ defined over (K, X, Y) is a PRF if:

- Given a key $K \in \{0, 1\}^l$ and an input $X \in \{0, 1\}^n$, there is an efficient algorithm to compute $f_K(X) = f(X, K)$ in poly(n) time
- Querying function f on many different inputs looks like a uniformly random output process

There are different constructions of PRF. It can be constructed using a pseudorandom number generator. PRF can also be constructed based on the DDH assumption. PRF is also an important tool for cryptographic primitive constructions such as encryption schemes and message authentication codes. PRFs can be used for constructing a memoryless authentication scheme, constructing a radar-based authentication system, and resolving collisions in a hash table data structure. PRF also helps to solve the secure login problem. A popular example is the Google Authenticator app which uses a pseudorandom function to generate a one-time password. Another example of a secure login is the SecurID token of RSA.

PRFs have also been used in blockchain. Recently, PRF has been used to implement the commitment in an anonymous lottery protocol for proof-of-stake consensus [125]. Ouroboros Crypsinous [126], a proof-of-stake-based blockchain, also defines a PRF with the additional property of unpredictability under malicious keys. This PRF is called MUPRF and is used in the leader election process by applying it on the input of a slot and checking it with a certain threshold value. There are other variants of PRFs such as oblivious PRF or threshold PRF. In an oblivious PRF, information between two parties, involved in PRF, is hidden from each other. Oblivious PRF is an interesting variant of PRF which has been used in different applications, e.g., key management of storage systems [127]. A threshold PRF can be applied to implement the efficient byzantine agreement by replacing the random oracle in the protocol. The first practical threshold implementation was

given by Neilsen in 2002 [128]. After that, it has been used in different applications, also in the blockchain. A privacy-preserving trustworthy smart contract platform *Ekiden* [129] uses threshold PRF to generate periodic smart contract-specific symmetric keys to encrypt the smart contract.

Another exciting PRF is the Legendre PRF which is a 1-bit PRF; $\mathbb{F}_p \rightarrow \{0, 1\}$ defined using the Legendre symbol:

$$L_{K,p}(x) = \left\lceil \frac{1}{2}\left(\left(\frac{K+x}{p}\right)+1\right)\right\rceil$$

where x is the PRF input and K is the secret. Legendre PRF was proposed by Damgård [130]. It performs very efficiently in arithmetic MPC [131]. Due to its exceptional performance and suitability in MPC, it is being applied in the construction of Ethereum 2.0 [132], where it is used in proof of custody by replacing mix function with Legendre PRF. It can be further applied in other efficient blockchain construction involving MPC.

Research Problem Find the use cases of MPC-friendly Legendre PRF in blockchain ecosystem. □

2.4 Oblivious RAM

Oblivious random access machine (ORAM) was initially introduced by Goldreich [133] and Ostrovsky [134]. Oblivious RAM was introduced for secure data outsourcing. Oblivious RAM is a protocol between client and server to store sensitive or encrypted data on the server without any privacy leakage from the user's access pattern. Oblivious RAM has been considered theoretical for most of its lifetime; however, recently, due to the rapid growth of cloud platforms/computing, there has been tremendous interest from industry and academia. Many cloud storage applications have started applying ORAM in their cloud system to achieve provable security.

Application of ORAM in blockchain can be found in a blockchain-based cloud server, in order to capture the malicious behavior of the user or the cloud server.

Definition 14 Let an input of length l from the client be defined as

$$\mathbf{y} = \{(op_l, addr_l, block_l) \ldots, (op_1, addr_1, block_1)\}$$

where op_i defines either read or write operation on the address $addr_i$ and $block_i$ denotes the data at address $addr_i$.

Define y as the operation sequence performed by a client and $A(y)$ as the access pattern returned by the ORAM protocol, which is a sequence of accesses to the server storage. An oblivious RAM protocol guarantees that for any two input

sequences y_1, y_2 of the same length from the same client, the access patterns $A(y_1)$ and $A(y_2)$ are computationally indistinguishable.

In ORAM protocol, client has limited memory and server has large storage. In this client-server model, client stores data at the server and the client can:

- Apply MAC on data such that server cannot change it
- Apply encryption to hide the actual data
- Perform a sequence y of operations where operations can be read or write

ORAM was originally designed for software piracy solutions, but later ORAM has appeared as a good candidate for multiparty computation, privacy in social networks, etc. Numerous constructions of ORAM have been proposed such as basic square-root ORAM [133], basic hierarchical ORAM [135], basic binary tree ORAM [136], etc. However, to adopt one of the many available ORAM constructions, Chang et al. provided a thorough study of several ORAM constructions and their implementation [137]. Recently, Asharov et al. presented a secure optimal oblivious RAM with optimal overhead using one-way function [138].

ORAM has been found very useful in different blockchain applications and use cases. A protocol named Solidus [139] gives a new construction of ORAM called publicly verifiable ORAM for the confidential transactions of its users. Another work on the usage of ORAM penalizes the server in Ethereum blockchain for losing the data [140]. Recent work proposes a Blockchain-based Traceable Oblivious RAM to detect the malicious behaviors of client and server [141]. Functionality of ORAM along with trusted execution environment (TEE) has been leveraged to protect the privacy of Bitcoin simplified verification payment (SVP) clients [142].

Research Problem Construct an oblivious RAM with restricted access patterns tailored for the blockchain use. □

2.5 Oblivious Transfer

The oblivious transfer was introduced in 1985 by Rabin. An oblivious transfer (OT) is a protocol between two parties, a sender S and a receiver R with the following definition:

Definition 15 In oblivious transfer protocol, the sender S holds n pieces of information and transfers potentially many pieces from n to the receiver R but remains oblivious to which pieces have been transferred.

There are many variants of oblivious transfer; some of these are equivalent:

- Rabin oblivious transfer: Sender S chooses a bit b. Then receiver R gets the bit b with probability $1/2$, or nothing otherwise.
- 1-out-of-2 oblivious transfer: Sender S chooses two input bits b_0, b_1. Then receiver R chooses a selection bit s and gets the output bit b_s.

The above two variants can be reduced to each other. Moreover, similar functionalities can be constructed for the strings instead of bits. OT can be constructed from DDH, RSA, enhanced trapdoor permutations, and lattices. Recently, highly efficient OT has been constructed in a random oracle model under a wide range of security assumptions such as DDH, CDH, and coding-based assumptions [143].

A protocol realizing oblivious transfer is said to be secure if the parties learn nothing more about what they intended to learn as per the definition of OT. Many generic secure computation protocols use OT as a basis such as Yao's garbled circuits [144] and Goldreich-Micali-Wigderson multiparty protocol construction for playing any mental game [145]. Hence, OT is a sufficiently useful and powerful building block for constructing arbitrary multiparty computation protocol, e.g., bit commitment. OT can also be directly applied in special-purpose protocols such as set intersection [146] and biometric identification [147]. Apart from these protocols, OT can also be used in other protocols such as private information retrieval, exchange of secrets, and contract signing.

In the context of blockchain, the oblivious transfer has been used to build private and verifiable smart contract [148] and privacy-preserving smart grid tariff decision [149], for an autonomous and fair privacy-preserving scheme for big data exchange [150]. It has also been used to construct scriptless lotteries in blockchain [151]. Some extensions of oblivious transfer are also designed to fulfill specific goals in blockchain, e.g., for penalizing multimedia breaches automatically [152].

Research Problem Design a privacy-preserving data transmission scheme using oblivious transfer for efficient blockchain synchronization. □

2.6 Access Control Encryption

Access control encryption (ACE) provides a way to selectively restrict the access of data from unauthorized parties using some policies and criteria. In this context, data refers to any resource or information.

Definition 16 (Access Control Encryption (ACE)[153]) An access control encryption (ACE) scheme is defined by the following algorithms:

- **Setup:** The Setup algorithm on input the security parameter κ and a policy $P : [n] \times [n] \leftarrow \{0, 1\}$ outputs a master secret key msk and public parameters pp, which include the message space $\mathcal{M}$ and ciphertext spaces C, C'.
- **Key Generation:** The Gen algorithm on input the master secret key msk, an identity $i \in \{0, \ldots, n+1\}$, and a type $t \in \{\mathsf{sen}, \mathsf{rec}, \mathsf{san}\}$ outputs a key k. We use the following notations for the three kinds of keys in the system:
  - $ek_i \leftarrow \mathsf{Gen}(msk, i, \mathsf{sen})$ (we call it an encryption key for $i \in [n]$)
  - $dk_j \leftarrow \mathsf{Gen}(msk, j, \mathsf{rec})$ (we call it a decryption key for $j \in [n]$)
  - $ek_0 = dk_0 = pp$

– $rk \leftarrow \mathsf{Gen}(msk, n+1, \mathsf{san})$ (we call it the sanitizer key)

- **Encrypt:** The Enc algorithm on input an encryption key ek_i and a message m outputs a ciphertext c.
- **Sanitizer:** San transforms an incoming ciphertext $c \in \mathcal{C}$ into a sanitized ciphertext $c' \in \mathcal{C}'$ using the sanitizer key rk.
- **Decryption:** Dec recovers a message $m' \in \mathcal{M} \cup \{\perp\}$ from a ciphertext $c' \in \mathcal{C}$ using a decryption key dk_j.

An ACE scheme has the following requirements:

- Correctness: When an honest sender S_i sends a message m, all receivers R_j with $P(i, j) = 1$ learn m.
- No-Read Rule: At the same time, all receivers R_j with $P(i, j) = 0$ learn no information about m.
- No-Write Rule: No (corrupt) sender S_i should be able to communicate any information to any (possibly corrupt) receiver R_j if $P(i, j) = 0$.

The access control systems on blockchains are based on different types of access control mechanisms.

- *Attribute-based encryption (ABE) mechanism* [154] is public key encryption sharing data from one to many entities. Only the user whose attributes satisfy the access policy set by the encryptor can decrypt the ciphertext. These attributes can be user-specific, environment-specific, or object-specific. Wang et al. [155] introduce a framework for data sharing access control, which is based on IPFS decentralized storage system, Ethereum blockchain, and ABE.
- *Oblivious Commitment-Based Envelope (OCBE)* [156] enables attribute-based access control without revealing any information about the attributes, i.e., the user obtains a service if and only if the attribute values satisfy the policy of the service provider, yet the service provider learns nothing about these attribute values. Alansari et al. [157] present an identity and access control management framework for cloud federation while keeping the attributes related to the users' identity private by using the OCBE and Intel SGX (trusted execution environment).

2.7 Identity-Based Encryption

Identity-based encryption (IBE) was first proposed in 1984 by Adi Shamir [158]. However, Shamir only gave the instantiation of identity-based signatures. Identity-based encryption was an open problem for many years. Later, it was realized as a complete cryptographic primitive by Boneh and Franklin in 2001 [159] using pairings. In identity-based encryption, any party can generate its public key from any known identity value. An encrypting party uses any known identity of the receiving party as its public key to encrypt a message. Upon receiving the encrypted message, the receiving party communicates with a trusted third party called private

key generator (PKG), which generates the corresponding private key to decrypt the encrypted message. There are a few extensions of IBE such as hierarchical IBE [160], anonymous IBE [161], and lattice-based IBE [162].

Definition 17 An IBE scheme consists of four algorithms:

- Setup(k): This algorithm is run by PKG to generate public parameters $params$ and a master secret key msk, given an input k. Parameters include message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.
- Extract($params, msk, ID \in \{0, 1\}^*$): This algorithm is run by PKG whenever a user requests his private key corresponding to the public identifier ID. PKG returns private key s for the identifier ID to the user.
- Encrypt($params, ID, m$): On input $params$, public key ID of receiver, and a message $m \in \mathcal{M}$, it outputs a ciphertext $c \in \mathcal{C}$.
- Decrypt($params, s, c$): On input ciphertext c, it outputs message m using s and parameters $params$.

Identity-based encryption can be helpful to generate public keys using social identities as LinkedIn, Twitter, etc. IBE can bring a major advantage in the case of finite users where after generating all users' private keys, the master secret key of PKG can be destroyed. IBE also removes the need for public key infrastructure by introducing PKG. The major drawback of IBE is PKG to be compromised. As PKG is a single entity, it does not suit well to be used directly in a permissionless blockchain system. However, some blockchain applications can still benefit from IBE to use it in their specific domain. IBE can be a good candidate for distributed ledger and permissioned blockchains. The trusted third party responsible to generate users' private keys can be replaced by PKG of IBE schemes in permissioned blockchains.

There have been some instantiations of IBE in the blockchain. An improved key distribution solution is designed using the integration of IBE with blockchain [163]. Another usage of IBE in blockchain context can be found in blockchain-based access verification protocol [164] and blockchain-based IC supply chain management [165].

2.8 Incremental Cryptography

Incremental cryptography is a method to speed the computation of cryptographic functions in certain settings. These cryptographic functions can be signing, encryption, hash functions, MAC, etc. The idea behind incremental cryptography is to design cryptographic algorithms such that if the algorithm is already applied to a document and this document is modified further, then the result of the algorithm for the modified document can be quickly computed from the old value rather than recomputing it from scratch. Incremental cryptography was introduced in 1994 by

Bellare et al. [166]. The other follow-up works [167, 168] make the foundation of incremental cryptography strong.

Definition 18 In incremental cryptography, once a transformation is applied to a document M, and if it is modified to M', then the time to update the result for M' should be directly proportional to the amount of modification done to M.

The challenge in incremental cryptography is not only to allow updates but also to ensure that updates are secure. Incremental cryptography can be very relevant and improve the efficiency in the settings where cryptographic algorithms (signatures, hash function, etc.) are frequently applied to changing documents. Now in this world of big data, incremental cryptography can be very important where recomputation of cryptographic functions from scratch is quite expensive.

Initial works on incremental cryptography focus on an incremental hash function, incremental signature scheme, and incremental message authentication schemes. However, in the past decade, many works proposed the incremental scheme of other cryptographic functions such as incremental public key encryption [169] and incremental pseudorandom function [170].

An efficient public verifiable two-party secure computation protocol over blockchain uses an incremental hash function for its construction [171]. An incremental hash scheme is also applied in the design of blockchain-enabled privacy audit logs [172]. Kadena blockchain [173] also proposes the use of incremental hashing for its transaction verification and updating the distributed log among blockchain nodes. A decentralized cryptocurrency, Coda, uses incrementally computable SNARKs to construct succinct blockchain [114].

Research Problem Construct a new blockchain mechanism that uses an incremental hash function for the updates of the distributed ledger. □

2.9 Lightweight Cryptography

Lightweight cryptography is evolved to overcome the difficulties that arise with the applicability of conventional cryptography on resource-constrained devices (limited processing and storage capabilities). Lightweight cryptography has been an emerging and hot topic in the field of cryptography. The driving force behind lightweight cryptography is the lack of cryptographic primitives that can be run efficiently in resource-constrained environments such as RFID, IoT devices, and sensor networks.

Definition 19 Lightweight cryptography is a subfield of cryptography aiming toward providing crypto solutions tailored to resource-constrained environments.

In lightweight cryptography, there is a trade-off between lightweightness and security of the lightweight crypto primitive which states how to achieve a high level of security having only limited (small) computational power. The lightweightness of

a crypto primitive is defined as a weight of an algorithm for the primitive. Naively, the weight of an algorithm can be treated as the amount of resources required for the primitive in terms of time and space. Hence, depending upon the implementation design, the weight property can be different in distinct contexts: mainly in software and in hardware. In the case of software, the weight of an algorithm is measured in terms of throughput, code size, and RAM. However, in the context of hardware, the weight of an algorithm is determined by the hardware area, throughput, latency, and power consumption. This lightweightness in software and hardware does not imply each other which means there exists a trade-off between these two.

In the past decade, there has been tremendous growth in the implementation of lightweight cryptography algorithms and primitives, and many of these have been standardized. Many designs for lightweight ciphers, hash functions, and one-pass authentication encryption have been proposed. The works given in [174–176] describe lightweight cryptography methods, designs, and their implementation.

Lightweight cryptography has been actively investigated for use in the blockchain. In the blockchain domain, it has been mostly applied for IoT blockchains due to its nature of applicability in constrained environments. IOTA distributed ledger [177, 178] is designed for IoT nodes to solve the scalability problem of blockchain. Another work [179] for scalable lightweight blockchain, LSB, provides privacy and security for IoT devices. Other interesting works focusing on the amalgam of lightweight cryptographies are blockchain-based authentication for IoT [180] and blockchain client for lightweight users [181].

Research Problem Construct a privacy-preserving scalable lightweight blockchain whose consensus relies on lightweight cryptographic primitives. □

2.10 White-Box Cryptography

White-box cryptography aims to protect the confidentiality of secret data (usually secret key) of a cryptographic algorithm in a white-box environment. A white-box environment can be defined as where an adversary not only sees the input and output of a cryptographic algorithm but also has access to see and amend the intermediate steps and data. White-box cryptography was originally designed as an obfuscation technique to construct software implementation of cryptographic algorithms in such a way that an adversary having access to the implementation and execution environment is unable to extract the secret key.

White-box cryptography [182, 183] was introduced in 2002 as a new model for evaluating cryptographic software, called a white-box attack context. In 2002, the first white-box attack-resistant commercial implementation of AES was produced. Later, white-box implementation of elliptic curve algorithms, RSA, SHA, etc., was introduced. From 2003 to 2011, white-box implementations of well-known ciphers were produced [184]. White-box implementations achieve better security but suffer in terms of efficiency.

White-box cryptography applies cryptographic techniques such as obfuscation, encryption, and mathematical transformations to protect the secret keys and data inside the software running in untrusted environments. This mathematical transformation to blend the software code and secret keys prevents these keys from being extracted from the software. In original white-box cryptography works, a fixed key was baked into the software. Nevertheless, many applications need variable keys in their designs which means the software protects an input key and operates with different keys at different times. Variable-key versions of white-box implementation were also produced. Every white-box implementation of an algorithm is unique and usually confidential to the creator. These white-box implementations can protect against timing attacks, runtime code modification, static analysis, and fault injection among others.

In a blockchain, white-box cryptography can be applied to protect the secret key inside the smart contract code which can unlock different functions (such as signing, encryption) during the execution of a smart contract on the blockchain. Hence, it can be helpful to establish the trust and privacy of assets in the blockchain. In the blockchain, especially in crypto-wallets, keys and their seeds are a single point of compromise; these can be safely stored using white-box cryptography. An instantiation of white-box cryptography is given to design a runtime self-protection environment in a trusted ledger [185]. Therefore, white-box cryptography brings an opportunity to do more investigation and its applicability to be applied in the blockchain.

2.11 Threshold Cryptography

Threshold cryptography was introduced by Desmedt [186, 187]. However, the first threshold system with complete threshold and security properties for trapdoor function in 1994 was introduced by Santis et al. [188]. In a way, threshold cryptography can be considered as cryptographic key management using multiparty computation.

In a traditional public key cryptosystem, the power to decrypt ciphertext or to sign a message is centralized to the party holding the secret key. However, the power given to a single party can cause several damages to the system. Threshold cryptography allows splitting a secret key into multiple shares among multiple parties so that the cryptographic operations as decryption or signing can be done with the distributed secret key shares without the construction of a single secret key. The number of required shares to perform decryption or signing is determined by a set threshold value t, which means any group having at least t parties can collaborate. If the total number of parties is n, then the system is called as $(t, n)-$threshold system.

Definition 20 (Threshold Cryptosystem) It consists of three protocols:

- KeyGen: Key is generated jointly by the participants.

- Enc: Ideally, the sender does not know the cryptosystem is threshold.
- Dec: The ciphertext can be decrypted by an authorized set without explicitly reconstructing the secret key.

Security of threshold cryptosystem follows that no fewer number of participants than the threshold should be able to reconstruct the secret.

Threshold versions of encryption or signing can be built for many public key cryptographic schemes. These threshold versions have been defined by the cryptosystems such as RSA, ElGamal, ECDSA, etc. Threshold cryptography is mostly used for storing secret information with multiple parties as a form of shares. This secret is mostly the secret key of a signature scheme or public key encryption scheme. Threshold cryptography has been applied in schemes as voting, auctions, etc.

The application of threshold cryptography has been found in many contexts of blockchain. A threshold signature scheme has been applied in different blockchain platforms. While preserving the distributed nature of blockchain, threshold signature in blockchain improves the robustness and resilience of the system. Threshold signature has been implemented in permission blockchain platform Hyperledger [34]. Their constructions were based on threshold versions of RSA and BLS signature schemes. A leader-based byzantine fault-tolerant consensus protocol, HotStuff [189], also makes use of a threshold signature scheme to combine the signed votes over a leader proposal. Threshold signature schemes are also used in cryptocurrency wallets. Threshold scheme-based wallet enables more security of the private key. It also provides the private key rotation without making any change with the corresponding public key and blockchain address. Goldfeder et al. [38] gave a design and implementation of ECDSA threshold signature scheme for securing the Bitcoin wallets. Later, another design based on a weighted ECDSA threshold signature scheme has also been proposed for Bitcoin wallet [190].

Threshold signature scheme can also replace different functionalities in blockchains such as on-chain mixing solutions [191] or scaling solutions. Not only the threshold signature scheme but also the threshold encryption scheme is an important candidate to be applied in the blockchain. Honey Badger BFT consensus protocol uses a threshold encryption scheme to prevent the targeted censorship attacks in a BFT network. Regarding the applications using threshold cryptography, many instantiations can be found, e.g., voting [192] and smart vehicles [193].

2.12 Post-quantum Cryptography

Post-quantum cryptography is about devising cryptographic algorithms (usually public key algorithms) that are thought to be secure against the attack by quantum computers.

With the advent of quantum computers, most of the public key algorithm including RSA and ECDSA can be broken in polynomial time using Shor's

algorithm [194]. Moreover, Grover's algorithm [195] can also cause sufficient breakdown by accelerating the generation of hashes or detecting the hash collisions. Both algorithms are groundbreaking which have built a strong foundation toward breaking today's number theoretic-based public key cryptosystems. Shor's algorithm is a polynomial-time algorithm for solving integer factorization and discrete logarithm problems. However, Grover's algorithm can be used to break the symmetric key cryptosystem. After the invention of these algorithms, many follow-up works were proposed and can be called a variant of these algorithms. As most of the blockchain systems and cryptocurrencies are based on elliptic curve cryptography, Shor's algorithm and its variants can be used for breaking these systems.

For building a post-quantum cryptosystem, there are several directions to pursue. The following are the major interesting research directions:

- *Hash-Based Cryptography* Hash-based cryptography aims toward designing new signature schemes based on the security of hash functions, e.g., SHA-3. New hash functions (e.g., [196]) have been proposed recently to resist quantum attacks. The usual recommendation is to increase the output size of the hash function. In the context of blockchain, Grover's algorithm can be useful to attack the blockchain by searching for the hash collisions and replacing the entire blockchain blocks. Grover's algorithm can also be used to accelerate the mining in blockchains (e.g., in Bitcoin). On the other hand, Shor's algorithm can also impact hash functions. It can be used to forge digital signatures and steal assets if the hash function of a blockchain is already broken. Current blockchain hash functions such as SHA256, Ethash, and Scrypt provide 128-bit post-quantum security.
- *Code-Based Cryptography* The security of this system relies on the hardness of problems related to coding theory. Hence, these systems are based on error-correcting codes. McEliece's cryptosystem [197] is an example of a code-based cryptosystem and its security is based on syndrome decoding problem [198]. Another hard problem to design the cryptosystem can be learning parity with noise [199]. In the past, many post-quantum code-based signing algorithms have been proposed; some of these are based on Niederreiter scheme [200] and CFS (Courtois-Finiasz-Sendrier) scheme [201]. The signature of these systems is of short size and efficiently verifiable compared to traditional McEliece's cryptosystem.
- *Lattice-Based Cryptography* This cryptosystem guarantees its security based on the hard problems on lattice such as finding a short vector in high-dimensional lattices (shortest vector problem (SVP)). The other problems on lattices are the closest vector problem (CVP) or the shortest independent vectors problem (SIVP) [202], which cannot be solved efficiently through quantum computers. The most promising lattice-based cryptosystems are based on the learning with error (LWE) problem [203] and the polynomial algebra [204]. Lattice-based cryptosystems are the most active research direction in recent years. These systems enable some powerful cryptographic primitives as homomorphic

encryption and functional encryption. Many lattice-based schemes [205, 206] have been constructed to be used in blockchain.

- *Multivariate-Based Cryptography* The security of these systems relies on solving the complex multivariate system of equations which have been demonstrated as NP-complete or NP-hard problems [207]. These systems suffer from the large key size and ciphertext overhead despite having the resistance to quantum attacks. Some of the most popular multivariate-based schemes rely on Matsumoto-Imai's algorithm, on Isomorphism of Polynomials (IP) [208], or on variants of HFE, which can generate signatures with a size comparable to the currently used RSA or ECC-based signatures [209]. Other constructions on the multivariate-based signature scheme are based on pseudorandom multivariate quadratic equations or Rainbow-like signing schemes [210].

Based on the above different cryptosystems, there have been several post-quantum blockchain proposals (e.g., [211]). Many blockchain platforms (e.g., [212]) are going to adopt the post-quantum cryptosystem to make their blockchain resistant to quantum attacks.

Research Problem Construct a new blockchain mechanism that has comparably efficient public key addresses and comparably small digital signatures as the currently used ones but that is based on post-quantum cryptographic schemes. The blockchain should be less computation demanding and should provide fast execution of transactions. □

3 Conclusion

The underlying design concepts of cryptography make the blockchain function. Cryptographic primitives not only are the core of blockchain but also bring a wide spectrum of applicability of blockchain in the real world. This chapter describes a wide spectrum of cryptographic primitives that are or can be embraced with blockchain. To overcome the possible challenges such as different attacks, many efficient constructions of cryptographic primitives have been proposed since the introduction of blockchain. Nevertheless, blockchain still faces many research challenges. Therefore, many cryptographic primitives and their designs are needed to be thoroughly studied and scrutinized to improve upon the current designs and to solve the challenging problems.

The chapter also presents some interesting challenging problems. Some of these problems are already published in the literature and some of the problems are newly formulated research problems. Cryptographers interested in blockchain can choose a challenging research problem and can further work toward finding an efficient solution for the problem.

References

1. A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction* (Princeton University Press, Princeton, 2016)
2. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2009). http://bitcoin.org/bitcoin.pdf
3. C. Lee, Litecoin (2011)
4. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Keccak, in *Advances in Cryptology – EUROCRYPT 2013*, ed. by T. Johansson, P.Q. Nguyen (Springer Berlin Heidelberg, 2013), pp. 313–314
5. G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow Paper (2014)
6. A. Biryukov, D. Khovratovich, Equihash: asymmetric proof-of-work based on the generalized birthday problem. Ledger **2**, 1–30 (2017)
7. E. Duffield, D. Diaz, Dash: a payments-focused cryptocurrency. Whitepaper (2018). https://github.com/dashpay/dash/wiki/Whitepaper
8. The Monero Project, Monero (2014)
9. D. Hopwood, S. Bowe, T. Hornby, N. Wilcox, *Zcash Protocol Specification*. (GitHub, San Francisco, 2016)
10. E. Duffield, K. Hagan, Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system. bitpaper.info (2014)
11. I. Dinur, N. Nadler, Time-memory tradeoff attacks on the mtp proof-of-work scheme, in *Advances in Cryptology – CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 375–403
12. E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza, T. Dryja, Cryptanalysis of curl-P and other attacks on the IOTA cryptocurrency. *IACR Cryptology ePrint Archive*, 344 (2019)
13. D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ECDSA). Int. J. Informat. Secur. **1**, 36–63 (2001)
14. S. Josefsson, I. Liusvaara, Edwards-curve digital signature algorithm (EDDSA), in *Internet Research Task Force, Crypto Forum Research Group, RFC*, vol. 8032 (2017)
15. C.P. Schnorr, Efficient identification and signatures for smart cards, in *Advances in Cryptology — CRYPTO' 89 Proceedings*, ed. by G. Brassard (Springer, New York, 1990), pp. 239–252
16. Libra Association, The Libra Blockchain (2019). https://developers.libra.org/docs/assets/papers/the-libra-blockchain.pdf
17. R.L. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in *Advances in Cryptology — ASIACRYPT 2001*, ed. by C. Boyd (Springer, Berlin, 2001), pp. 552–565
18. N. van Saberhagen, Cryptonote (2013)
19. G. Maxwell, Confidential transactions (2015). https://people.xiph.org/greg/confidentialvalues.txt. Accessed 09 May 2016
20. S. Noether, Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015). https://eprint.iacr.org/2015/1098
21. K. Itakura, K. Nakamura, A public-key cryptosystem suitable for digital multisignatures. *NEC Research and Development* (1983)
22. G. Maxwell, A. Poelstra, Y. Seurin, P. Wuille, Simple schnorr multi-signatures with applications to bitcoin. Designs Codes Cryptogr. **87**(9), 2139–2164 (2019)
23. M. Bellare, G. Neven, Multi-signatures in the plain public-key model and a general forking lemma, in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06 (Association for Computing Machinery, New York, 2006), pp. 390–399
24. D. Boneh, M. Drijvers, G. Neven, Compact multi-signatures for smaller blockchains, in *Advances in Cryptology – ASIACRYPT 2018*, ed. by T. Peyrin, S. Galbraith (Springer International Publishing, Cham, 2018), pp. 435–464

25. D. Boneh, B. Lynn, H. Shacham, Short signatures from the weil pairing, in *Advances in Cryptology — ASIACRYPT 2001*, ed. by C. Boyd (Springer, Berlin, 2001), pp. 514–532
26. D. Chaum, *Blind Signature System* (Springer US, Boston, 1984), pp. 153–153
27. L. Valenta, B. Rowan, Blindcoin: Blinded, accountable mixes for bitcoin, in *Financial Cryptography and Data Security*, ed. by M. Brenner, N. Christin, B. Johnson, K. Rohloff (Springer, Berlin, 2015), pp. 112–126
28. Q. ShenTu, J. Yu, A blind-mixing scheme for bitcoin based on an elliptic curve cryptography blind digital signature algorithm. CoRR **abs/1510.05833** (2015).
29. E. Heilman, F. Baldimtsi, S. Goldberg, Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions, in *Financial Cryptography and Data Security*, ed. by J. Clark, S. Meiklejohn, P.Y. Ryan, D. Wallach, M. Brenner, K. Rohloff (Springer, Berlin, 2016), pp. 43–60
30. H. Tian, F. He, L. Fu, A privacy preserving fair contract signing protocol based on public block chains. J. Cryptolog. Res. **4**(2), 187–198 (2017)
31. Y. Liu, Q. Wang, An e-voting protocol based on blockchain. IACR Cryptol. ePrint Arch. **2017**, 1043 (2017)
32. J.P. Cruz, Y. Kaji, E-voting system based on the bitcoin protocol and blind signatures. IPSJ Trans. Math. Model. Appl. **10**(1), 14–22 (2017)
33. F. Sheer Hardwick, A. Gioulis, R. Naeem Akram, K. Markantonakis, E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (2018), pp. 1561–1567
34. C. Stathakopoulous, C. Cachin, Threshold signatures for blockchain systems. Swiss Federal Instit. Technol. **30**, 1 (2017)
35. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S.W. Cocco, J. Yellick, Hyperledger fabric: A distributed operating system for permissioned blockchains, in *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18 (Association for Computing Machinery, New York, 2018)
36. J.H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, K. Wehrle, Coinparty: Secure multi-party mixing of bitcoins, in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15 (Association for Computing Machinery, New York, 2015), pp. 75–86
37. O. Shlomovits, I.A. Seres, Sharelock: mixing for cryptocurrencies from multiparty ecdsa. IACR Cryptol. ePrint Arch. **2019**, 563 (2019)
38. S. Goldfeder, R. Gennaro, H. Kalodner, J. Bonneau, J.A. Kroll, E.W. Felten, A. Narayanan, Securing bitcoin wallets via a new DSA/ECDSA threshold signature scheme (2015)
39. A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in *2016 IEEE Symposium on Security and Privacy (SP)* (2016), pp. 839–858
40. B. Bünz, S. Agrawal, M. Zamani, D. Boneh, Zether: Towards privacy in a smart contract world, in *International Conference on Financial Cryptography and Data Security* (Springer, Berlin, 2020), pp. 423–443
41. S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, H. Wu, Zexe: Enabling decentralized private computation, in *2020 IEEE Symposium on Security and Privacy (SP)* (IEEE, Piscataway, 2020), pp. 947–964
42. E. Syta, P. Jovanovic, E.K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M.J. Fischer, B. Ford, Scalable bias-resistant distributed randomness, in *2017 IEEE Symposium on Security and Privacy (SP)* (IEEE, Piscataway, 2017), pp. 444–460
43. A. Cherniaeva, I. Shirobokov, O. Shlomovits, Homomorphic encryption random beacon. IACR Cryptol. ePrint Arch. **2019**, 1320 (2019)

44. B. Bünz, S. Goldfeder, J. Bonneau, Proofs-of-delay and randomness beacons in ethereum, in *IEEE Security and Privacy on the Blockchain (IEEE S&B)* (2017)
45. L. Chen, W.-K. Lee, C.-C. Chang, K.-K.R. Choo, N. Zhang, Blockchain based searchable encryption for electronic health record sharing. Future Gener. Comput. Syst. **95**, 420–429 (2019)
46. H. Li, H. Tian, F. Zhang, J. He, Blockchain-based searchable symmetric encryption scheme. Comput. Electr. Eng. **73**, 32–45 (2019)
47. H. Cui, Z. Wan, X. Wei, S. Nepal, X. Yi, Pay as you decrypt: decryption outsourcing for functional encryption using blockchain. IEEE Trans. Informat. Forens. Secur. **15**, 3227–3238 (2020)
48. A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, R. Sirdey, Towards better availability and accountability for iot updates by means of a blockchain, in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)* (2017), pp. 50–58
49. Y. Rahulamathavan, R.C. Phan, M. Rajarajan, S. Misra, A. Kondoz, Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption, in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (2017), pp. 1–6
50. J. Benaloh, M. De Mare, One-way accumulators: A decentralized alternative to digital signatures, in *Workshop on the Theory and Application of of Cryptographic Techniques* (Springer, Berlin, 1993), pp. 274–285
51. J. Li, N. Li, R. Xue, Universal accumulators with efficient nonmembership proofs, in *International Conference on Applied Cryptography and Network Security* (Springer, Berlin, 2007), pp. 253–269
52. H. Lipmaa, Secure accumulators from euclidean rings without trusted setup, in *International Conference on Applied Cryptography and Network Security* (Springer, Berlin, 2012), pp. 224–240
53. A. Mashatan, S. Vaudenay, A fully dynamic universal accumulator. Proc. Romanian Acad. Ser. A-Math. Phys. Tech. Sci. Informat. Sci. **14**, 269–285 (2013)
54. L. Nguyen, Accumulators from bilinear pairings and applications, in *Cryptographers' Track at the RSA Conference* (Springer, Berlin, 2005), pp. 275–292
55. M.H. Au, P.P. Tsang, W. Susilo, Y. Mu, Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems, in *Cryptographers' Track at the RSA Conference* (Springer, Berlin, 2009), pp. 295–308
56. J. Camenisch, M. Kohlweiss, C. Soriente, An accumulator based on bilinear maps and efficient revocation for anonymous credentials, in *International Workshop on Public Key Cryptography* (Springer, Berlin, 2009), pp. 481–500
57. T. Acar, L. Nguyen, Revocation for delegatable anonymous credentials, in *International Workshop on Public Key Cryptography* (Springer, Berlin, 2011), pp. 423–440
58. K. Nyberg, Fast accumulated hashing, in *International Workshop on Fast Software Encryption* (Springer, Berlin, 1996), pp. 83–87
59. A. Buldas, P. Laud, H. Lipmaa, Accountable certificate management using undeniable attestations, in *Proceedings of the 7th ACM Conference on Computer and Communications Security* (2000), pp. 9–17
60. A. Buldas, P. Laud, H. Lipmaa, Eliminating counterevidence with applications to accountable certificate management 1. J. Comput. Secur. **10**(3), 273–296 (2002)
61. P. Camacho, A. Hevia, M. Kiwi, R. Opazo, Strong accumulators from collision-resistant hashing, in *International Conference on Information Security* (Springer, Berlin 2008), pp. 471–486
62. D. Boneh and H. Corrigan-Gibbs, Bivariate polynomials modulo composites and their applications, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2014), pp. 42–62
63. D. Boneh, B. Bünz, B. Fisch, Batching techniques for accumulators with applications to iops and stateless blockchains. Cryptology ePrint Archive, Report 2018/1188 (2018). https://eprint.iacr.org/2018/1188

64. A. Ozdemir, R. Wahby, B. Whitehat, D. Boneh, Scaling verifiable computation using efficient set accumulators, in *29th {USENIX} Security Symposium ({USENIX} Security 20)* (2020), pp. 2075–2092
65. G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci. **37**(2), 156–189 (1988)
66. T.P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in *Advances in Cryptology — CRYPTO '91*, ed. by J. Feigenbaum (Springer, Berlin, 1992), pp. 129–140
67. I. Miers, C. Garman, M. Green, A.D. Rubin, Zerocoin: Anonymous distributed e-cash from bitcoin, in *2013 IEEE Symposium on Security and Privacy* (2013), pp. 397–411
68. C. Ganesh, C. Orlandi, D. Tschudi, Proof-of-stake protocols for privacy-aware blockchains, in *Advances in Cryptology – EUROCRYPT 2019*, ed. by Y. Ishai, V. Rijmen (Springer International Publishing, Cham, 2019), pp. 690–719
69. H.S. Galal, A.M. Youssef, Verifiable sealed-bid auction on the ethereum blockchain, in *Financial Cryptography and Data Security*, ed. by A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, M. Sala (Springer, Berlin 2019), pp. 265–278
70. S.-F. Sun, M.H. Au, J.K. Liu, T.H. Yuen, Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero, in *Computer Security – ESORICS 2017*, ed. by S.N. Foley, D. Gollmann, E. Snekkenes (Springer International Publishing, Cham, 2017), pp. 456–474
71. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, D. Khovratovich, Aggregatable subvector commitments for stateless cryptocurrencies. IACR Cryptol. ePrint Arch. **2020**, 527 (2020)
72. A. Kate, G.M. Zaverucha, I. Goldberg, Constant-size commitments to polynomials and their applications, in *Advances in Cryptology - ASIACRYPT 2010*, ed. by M. Abe (Springer, Berlin, 2010), pp. 177–194
73. S. Gorbunov, L. Reyzin, H. Wee, Z. Zhang, Pointproofs: Aggregating proofs for multiple vector commitments. IACR Cryptol. ePrint Arch. **2020**, 419 (2020)
74. B. Libert, S. Ramanna, M. Yung, Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions, in *43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)* (2016)
75. M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, L. Nizzardo, Vector commitment techniques and applications to verifiable decentralized storage. IACR Cryptol. ePrint Arch. **2020**, 149 (2020)
76. P. Schindler, A. Judmayer, N. Stifter, E. Weippl, Hydrand: Efficient continuous distributed randomness, in *2020 IEEE Symposium on Security and Privacy (SP)* (IEEE, Piscataway, 2020), pp. 73–89
77. I. Cascudo, B. David, Albatross: Publicly attestable batched randomness based on secret sharing, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2020), pp. 311–341
78. A. Bhat, N. Shrestha, A. Kate, K. Nayak, Randpiper-reconfiguration-friendly random beacons with quadratic communication. IACR Cryptol. ePrint Arch. **2020**, 1590 (2020)
79. E. Kokoris Kogias, D. Malkhi, A. Spiegelman, Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures, in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020), pp. 1751–1767
80. G. Zyskind, O. Nathan, A. Pentland, Decentralizing privacy: Using blockchain to protect personal data, in *2015 IEEE Security and Privacy Workshops* (2015), pp. 180–184
81. F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, L. Reyzin, Can a public blockchain keep a secret? in *Theory of Cryptography*, ed. by R. Pass, K. Pietrzak (Springer International Publishing, Cham, 2020), pp. 260–290
82. R.K. Raman, L.R. Varshney, Distributed storage meets secret sharing on the blockchain, in *2018 Information Theory and Applications Workshop (ITA)* (2018), pp. 1–6

83. J.-H. Hsiao, R. Tso, C.-M. Chen, M.-E. Wu, Decentralized e-voting systems based on the blockchain technology, in *Advances in Computer Science and Ubiquitous Computing*, ed. by J.J. Park, V. Loia, G. Yi, Y. Sung (Springer, Singapore, 2018), pp. 305–309
84. M.A. Cyran, Blockchain as a foundation for sharing healthcare data. Blockchain Healthcare Today **1**, 1–6 (2018)
85. S. Halevi, Y. Ishai, A. Jain, I. Komargodski, A. Sahai, E. Yogev, Non-interactive multiparty computation without correlated randomness, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2017), pp. 181–211
86. G. Almashaqbeh, F. Benhamouda, S. Han, D. Jaroslawicz, T. Malkin, A. Nicita, T. Rabin, A. Shah, E. Tromer, Gage MPC: bypassing residual function leakage for non-interactive mpc. IACR Cryptol. ePrint Arch. **2021**, 256 (2021)
87. M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, in *2014 IEEE Symposium on Security and Privacy* (2014), pp. 443–458
88. Y. Zhu, X. Song, S. Yang, Y. Qin, Q. Zhou, Secure smart contract system built on smpc over blockchain, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (2018), pp. 443–458
89. M. Luongo, C. Pon, The keep network: A privacy layer for public blockchains, Tech. Rep. (2018). https://www.coinpare.io/whitepaper/keep-network.pdf
90. O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems. J. Cryptol. **7**, 1–32 (1994)
91. E.B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: Decentralized anonymous payments from bitcoin, in *2014 IEEE Symposium on Security and Privacy* (2014), pp. 459–474
92. M. Blum, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88 (Association for Computing Machinery, New York, 1988), pp. 103–112
93. J. Groth, On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Report 2016/260 (2016). https://eprint.iacr.org/2016/260
94. R. Gennaro, C. Gentry, B. Parno, M. Raykova, Quadratic span programs and succinct nizks without pcps. Cryptology ePrint Archive, Report 2012/215 (2012). https://eprint.iacr.org/2012/215
95. S. Setty, B. Braun, V. Vu, A.J. Blumberg, B. Parno, M. Walfish, Resolving the conflict between generality and plausibility in verified computation. Cryptology ePrint Archive, Report 2012/622 (2012). https://eprint.iacr.org/2012/622
96. E. Ben-Sasson, A. Chiesa, E. Tromer, M. Virza, Succinct non-interactive zero knowledge for a von neumann architecture. Cryptology ePrint Archive, Report 2013/879 (2013). https://eprint.iacr.org/2013/879
97. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, I. Miers, Updatable and universal common reference strings with applications to zk-snarks. Cryptology ePrint Archive, Report 2018/280 (2018). https://eprint.iacr.org/2018/280
98. M. Campanelli, D. Fiore, A. Querol, Legosnark: Modular design and composition of succinct zero-knowledge proofs. Cryptology ePrint Archive, Report 2019/142 (2019). https://eprint.iacr.org/2019/142
99. M. Maller, S. Bowe, M. Kohlweiss, S. Meiklejohn, Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099 (2019). https://eprint.iacr.org/2019/099
100. A. Kosba, D. Papadopoulos, C. Papamanthou, D. Song, Mirage: Succinct arguments for randomized algorithms with applications to universal zk-SNARKs. Cryptology ePrint Archive, Report 2020/278 (2020). https://eprint.iacr.org/2020/278
101. E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018). https://eprint.iacr.org/2018/046

102. S. Ames, C. Hazay, Y. Ishai, M. Venkitasubramaniam, Ligero: Lightweight sublinear arguments without a trusted setup, in *Proceedings of the 2017 ACM Sigsac Conference on Computer and Communications Security* (2017), pp. 2087–2104
103. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, N.P. Ward, Aurora: Transparent succinct arguments for r1cs. Cryptology ePrint Archive, Report 2018/828 (2018). https://eprint.iacr.org/2018/828
104. D. Galindo, J. Liu, M. Ordean, J.-M. Wong, Fully distributed verifiable random functions and their application to decentralised random beacons. IACR Cryptol. ePrint Arch. **2020**, 96 (2020)
105. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling byzantine agreements for cryptocurrencies, in *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17 (Association for Computing Machinery, New York, 2017), p. 51–68
106. M. Franklin, H. Zhang, Unique group signatures, in *European Symposium on Research in Computer Security* (Springer, Berlin, 2012), pp. 643–660
107. T. Hanke, M. Movahedi, D. Williams, Dfinity technology overview series, consensus system (2018). Preprint arXiv:1805.04548
108. M. Buser, R. Dowsley, M.F. Esgin, S.K. Kermanshahi, V. Kuchta, J.K. Liu, R. Phan, Z. Zhang, Post-quantum verifiable random function from symmetric primitives in pos blockchain. IACR Cryptol. ePrint Arch. **2021**, 302 (2021)
109. Z. Li, T.G. Tan, P. Szalachowski, V. Sharma, J. Zhou, Post-quantum VRF and its applications in future-proof blockchain system (2021). Preprint arXiv:2109.02012
110. S. Micali, M. Rabin, S. Vadhan, Verifiable random functions, in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)* (1999), pp. 120–130
111. Y. Zhao, X. Deng, C.H. Lee, H. Zhu, Resettable zero-knowledge in the weak public-key model, in *Advances in Cryptology — EUROCRYPT 2003*, ed. by E. Biham (Springer, Berlin, 2003), pp. 123–139
112. S. Micali, R.L. Rivest, Micropayments revisited, in *Cryptographers' Track at the RSA Conference* (Springer, Berlin, 2002), pp. 149–163
113. A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in *Advances in Cryptology – CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 357–388
114. I. Meckler, E. Shapiro, Coda: Decentralized cryptocurrency at scale. O (1) Labs Whitepaper. May, vol. 10 (2018), p. 4
115. L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al., Chainlink 2.0: Next steps in the evolution of decentralized oracle networks (2021)
116. D. Boneh, J. Bonneau, B. Bünz, B. Fisch, Verifiable delay functions, in *Advances in Cryptology – CRYPTO 2018*, ed. by H. Shacham, A. Boldyreva (Springer International Publishing, Cham, 2018), pp. 757–788
117. A. Juels, Client puzzles: A cryptographic countermeasure against connection depletion attacks, in *Proceedings of the Networks and Distributed System Security Symposium (NDSS), 1999* (1999)
118. B. Wesolowski, Efficient verifiable delay functions, in *Advances in Cryptology – EUROCRYPT 2019*, ed. by Y. Ishai, V. Rijmen (Springer International Publishing, Cham, 2019), pp. 379–407
119. K. Pietrzak, Simple verifiable delay functions, in *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, ed. by A. Blum. Leibniz International Proceedings in Informatics (LIPIcs), vol. 124 (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, 2018), pp. 60:1–60:15
120. P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, E. Weippl, Randrunner: distributed randomness from trapdoor vdfs with strong uniqueness. IACR Cryptol. ePrint Arch. **2020**, 942 (2020)
121. R. Han, J. Yu, H. Lin, Randchain: decentralised randomness beacon from sequential proof-of-work. IACR Cryptol. ePrint Arch. **2020**, 1033 (2020)

122. R. Khalil, N. Dulay, Short paper: Posh proof of staked hardware consensus. ePrint (2020)
123. J. Long, R. Wei, Nakamoto consensus with verifiable delay puzzle (2019). Preprint arXiv:1908.06394
124. V. Attias, L. Vigneri, V. Dimitrov, Preventing denial of service attacks in IoT networks through verifiable delay functions, in *GLOBECOM 2020–2020 IEEE Global Communications Conference* (IEEE, Piscataway, 2020), pp. 1–6
125. F. Baldimtsi, V. Madathil, A. Scafuro, L. Zhou, Anonymous lottery in the proof-of-stake setting. IACR Cryptol. ePrint Arch. **2020**, 533 (2020)
126. T. Kerber, A. Kiayias, M. Kohlweiss, V. Zikas, Ouroboros crypsinous: Privacy-preserving proof-of-stake, in *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 157–174
127. S. Jarecki, H. Krawczyk, J. Resch, Updatable oblivious key management for storage systems, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19 (Association for Computing Machinery, New York, 2019), pp. 379–393
128. J.B. Nielsen, A threshold pseudorandom function construction and its applications, in *Advances in Cryptology — CRYPTO 2002*, ed. by M. Yung (Springer, Berlin, 2002), pp. 401–416
129. R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, D. Song, Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts, in *2019 IEEE European Symposium on Security and Privacy (EuroS P)* (2019), pp. 185–200
130. I.B. Damgård, On the randomness of legendre and jacobi sequences, in *Advances in Cryptology — CRYPTO' 88*, ed. by S. Goldwasser (Springer, New York, 1990), pp. 163–172
131. L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, N.P. Smart, MPC-friendly symmetric key primitives, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16 (Association for Computing Machinery, New York, 2016), pp. 430–443
132. D. Feist, Legendre pseudo-random function (2019)
133. O. Goldreich, Towards a theory of software protection and simulation by oblivious rams, in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87 (Association for Computing Machinery, New York, 1987), pp. 182–194
134. R. Ostrovsky, Efficient computation on oblivious rams, in *Proceedings of the twenty-second annual ACM symposium on Theory of computing* (1990), pp. 514–523
135. O. Goldreich, R. Ostrovsky, Software protection and simulation on oblivious rams. J. ACM **43**, 431–473 (1996)
136. E. Shi, T.H.H. Chan, E. Stefanov, M. Li, Oblivious ram with $O((\log N)^3)$ worst-case cost, in *Advances in Cryptology – ASIACRYPT 2011*, ed. by D.H. Lee, X. Wang (Springer, Berlin, 2011), pp. 197–214
137. Z. Chang, D. Xie, F. Li, Oblivious ram: a dissection and experimental evaluation. Proc. VLDB Endow. **9**, 1113–1124 (2016)
138. G. Asharov, I. Komargodski, W.-K. Lin, K. Nayak, E. Peserico, E. Shi, Optorama: Optimal oblivious ram, in *Advances in Cryptology – EUROCRYPT 2020*, ed. by A. Canteaut, Y. Ishai (Springer International Publishing, Cham, 2020), pp. 403–432
139. E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, E. Shi, Solidus: Confidential distributed ledger transactions via pvorm, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17 (Association for Computing Machinery, New York, 2017), pp. 701–717
140. J. Gancher, A. Groce, A. Ledger, Externally verifiable oblivious ram. Proc. Privacy Enhancing Technol. **2017**(2), 149–171 (2017)
141. H. Cao, R. Li, W. Tian, Z. Xu, W. Xiao, Blockchain-based accountability for multi-party oblivious ram. J. Parall. Distribut.Comput. **137**, 224–237 (2020)
142. D.V. Le, L.T. Hurtado, A. Ahmad, M. Minaei, B. Lee, A. Kate, A tale of two trees: one writes, and other reads: optimized oblivious accesses to bitcoin and other UTXO-based blockchains. Proc. Privacy Enhanc. Technol. **2**, 2020 (2020)
143. D. Mansy, P. Rindal, Endemic oblivious transfer, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19 (Association for Computing Machinery, New York, 2019), pp. 309–326

144. A.C. Yao, How to generate and exchange secrets, in *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)* (1986), pp. 162–167
145. O. Goldreich, S. Micali, A. Wigderson, How to play any mental game, in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87 (Association for Computing Machinery, New York, 1987), pp. 218–229
146. C. Dong, L. Chen, Z. Wen, When private set intersection meets big data: An efficient and scalable protocol, in *Proceedings of the 2013 ACM SIGSAC Conference on Computer, Communications Security*, CCS '13 (Association for Computing Machinery, New York, 2013), p.p. 789–800
147. J. Bringer, H. Chabanne, A. Patey, Privacy-preserving biometric identification using secure multiparty computation: an overview and recent trends. IEEE Signal Process. Magaz. **30**(2), 42–52 (2013)
148. D.C. Sánchez, Raziel: private and verifiable smart contracts on blockchains. CoRR, **abs/1807.09484** (2018)
149. F. Knirsch, A. Unterweger, G. Eibl, D. Engel, *Privacy-Preserving Smart Grid Tariff Decisions with Blockchain-Based Smart Contracts*(Springer International Publishing, Cham, 2018), pp. 85–116
150. T. Li, W. Ren, Y. Xiang, X. Zheng, T. Zhu, K.-K.R. Choo, G. Srivastava, Faps: a fair, autonomous and privacy-preserving scheme for big data exchange based on oblivious transfer, ether cheque and smart contracts. Informat. Sci. **544**, 469–484 (2021)
151. L. Fournier, Scriptless bitcoin lotteries from oblivious transfer (2016). https://telaviv2019.scalingbitcoin.org/files/scriptless-lotteries-on-bitcoin-from-oblivious-transfer.pdf
152. E.V. Mangipudi, K. Rao, J. Clark, A. Kate, Towards automatically penalizing multimedia breaches, in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (IEEE, Piscataway, 2019), pp. 340–346
153. I. Damgård, H. Haagh, C. Orlandi, Access control encryption: Enforcing information flow with cryptography, in *Theory of Cryptography Conference* (Springer, Berlin, 2016), pp. 547–576
154. B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290 (2008). https://eprint.iacr.org/2008/290
155. S. Wang, Y. Zhang, Y. Zhang, A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access **6**, 38437–38450 (2018)
156. J. Li, N. Li, A construction for general and efficient oblivious commitment based envelope protocols, in *Information and Communications Security*, ed. by P. Ning, S. Qing, N. Li (Springer, Berlin, 2006), pp. 122–138
157. S. Alansari, F. Paci, V. Sassone, A distributed access control system for cloud federations, in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), pp. 2131–2136
158. A. Shamir, Identity-based cryptosystems and signature schemes, in *Advances in Cryptology*, ed. by G.R. Blakley, D. Chaum (Springer, Berlin, 1985), pp. 47–53
159. D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in *Advances in Cryptology — CRYPTO 2001*, ed. by J. Kilian (Springer, Berlin, 2001), pp. 213–229
160. D. Boneh, X. Boyen, E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 2005), pp. 440–456
161. D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 2004), pp. 506–522
162. L. Ducas, V. Lyubashevsky, T. Prest, Efficient identity-based encryption over ntru lattices, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2014), pp. 22–41
163. B. Zhou, H. Li, L. Xu, An authentication scheme using identity-based encryption & blockchain, in *2018 IEEE Symposium on Computers and Communications (ISCC)* (IEEE, Piscataway, 2018), pp. 00556–00561

164. S. Wei, S. Li, P. Liu, M. Liu, Bavp: Blockchain-based access verification protocol in leo constellation using ibe keys. Secur. Commun. Netw. **2018**, 1–14 (2018)
165. S. Bose, M. Raikwar, D. Mukhopadhyay, A. Chattopadhyay, and K.-Y. Lam, Blic: A blockchain protocol for manufacturing and supply chain management of ICS, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (IEEE, Piscataway, 2018), pp. 1326–1335
166. M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography: The case of hashing and signing, in *Advances in Cryptology — CRYPTO '94*, ed. by Y.G. Desmedt (Springer, Berlin, 1994), pp. 216–233
167. M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography and application to virus protection, in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing* (1995), pp. 45–56
168. M. Bellare, D. Micciancio, A new paradigm for collision-free hashing: Incrementality at reduced cost, in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 1997), pp. 163–192
169. I. Mironov, O. Pandey, O. Reingold, G. Segev, Incremental deterministic public-key encryption. J. Cryptol. **31**(1), 134–161 (2018)
170. V. Arte, M. Bellare, L. Khati, Incremental cryptography revisited: Prfs, nonces and modular design, in *International Conference on Cryptology in India* (Springer, Berlin, 2020), pp. 576–598
171. R. Zhu, C. Ding, Y. Huang, Efficient publicly verifiable 2PC over a blockchain with applications to financially-secure computations, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19 (Association for Computing Machinery, New York, 2019), pp. 633–650
172. A. Sutton, R. Samavi, Blockchain enabled privacy audit logs, in *The Semantic Web – ISWC 2017*, ed. by C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, J. Heflin (Springer International Publishing, Cham, 2017), pp. 645–660
173. W. Martino, Kadena: The first scalable, high performance private blockchain (2016)
174. W.J. Buchanan, S. Li, R. Asif, Lightweight cryptography methods. J. Cyb. Secur. Technol. **1**(3–4), 187–201 (2017)
175. T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, L. Uhsadel, A survey of lightweight-cryptography implementations. IEEE Design Test Comput. **24**(6), 522–533 (2007)
176. K. McKay, L. Bassham, M. Sönmez Turan, N. Mouha, Report on lightweight cryptography, Technical Report, National Institute of Standards and Technology (2016)
177. S. Popov, The tangle, *cit. on* (2016), p. 131
178. W.F. Silvano, R. Marcelino, Iota tangle: A cryptocurrency to communicate internet-of-things data. Future Gener. Comput. Syst. **112**, 307–319 (2020)
179. A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, LSB: A lightweight scalable blockchain for IoT security and anonymity. J. Parall. Distrib. Comput. **134**, 180–197 (2019)
180. U. Khalid, M. Asim, T. Baker, P.C. Hung, M.A. Tariq, L. Rafferty, A decentralized lightweight blockchain-based authentication mechanism for IoT systems. Cluster Comput. **23**, 1–21 (2020)
181. L. Xu, L. Chen, Z. Gao, S. Xu, W. Shi, EPBC: Efficient public blockchain client for lightweight users, in *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers* (2017), pp. 1–6
182. S. Chow, P. Eisen, H. Johnson, P.C. Van Oorschot, White-box cryptography and an aes implementation, in *Selected Areas in Cryptography*, ed. by K. Nyberg, H. Heys (Springer, Berlin, 2003), pp. 250–270
183. S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, A white-box des implementation for DRM applications, in *Digital Rights Management*, ed. by J. Feigenbaum (Springer, Berlin, 2003), pp. 1–15
184. B. Wyseur, White-box cryptography (2011)

185. C. Liem, E. Abdallah, C. Okoye, J. O'Connor, M.S.U. Alam, S. Janes, Runtime self-protection in a trusted blockchain-inspired ledger, in *15th ESCAR EUROPE* (2017)
186. Y. Desmedt, Society and group oriented cryptography: a new concept, in *Advances in Cryptology — CRYPTO '87*, ed. by C. Pomerance (Springer, Berlin, 1988), pp. 120–127
187. Y.G. Desmedt, Threshold cryptography. Eur. Trans. Telecommun. **5**(4), 449–458 (1994)
188. A. De Santis, Y. Desmedt, Y. Frankel, M. Yung, How to share a function securely, in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing* (1994), pp. 522–533
189. I. Abraham, G. Gueta, D. Malkhi, Hot-stuff the linear, optimal-resilience, one-message BFT devil. CoRR **abs/1803.05069** (2018)
190. P. Dikshit, K. Singh, Efficient weighted threshold ECDSA for securing bitcoin wallet, in *2017 ISEA Asia Security and Privacy (ISEASP)* (IEEE, Piscataway, 2017), pp. 1–9
191. O. Shlomovits, I.A. Seres, Sharelock: Mixing for cryptocurrencies from multiparty ecdsa. Cryptology ePrint Archive, Report 2019/563 (2019). https://eprint.iacr.org/2019/563
192. B. Gong, X. Lu, L. W. Fat, M.H. Au, Blockchain-based threshold electronic voting system, in *Security and Privacy in Social Networks and Big Data*, ed. by W. Meng, S. Furnell (Springer, Singapore, 2019), pp. 238–250
193. L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, Z. Zhang, Creditcoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. IEEE Trans. Intell. Transport. Syst. **19**(7), 2204–2220 (2018)
194. P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
195. L.K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (1996), pp. 212–219
196. S. Krendelev, P. Sazonova, Parametric hash function resistant to attack by quantum computer, in *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)* (IEEE, Piscataway, 2018), pp. 387–390
197. R.J. McEliece, A public-key cryptosystem based on algebraic. Coding Thv. **4244**, 114–116 (1978)
198. E. Berlekamp, R. McEliece, H. Van Tilborg, On the inherent intractability of certain coding problems (corresp.). IEEE Trans. Informat. Theory **24**(3), 384–386 (1978)
199. K. Pietrzak, Cryptography from learning parity with noise, in *International Conference on Current Trends in Theory and Practice of Computer Science* (Springer, Berlin, 2012), pp. 99–114
200. H. Niederreiter, Knapsack-type cryptosystems and algebraic coding theory. Prob. Control Inf. Theory **15**(2), 159–166 (1986)
201. N.T. Courtois, M. Finiasz, N. Sendrier, How to achieve a mceliece-based digital signature scheme, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2001), pp. 157–174
202. J. Blömer, S. Naewe, Sampling methods for shortest vectors, closest vectors and successive minima. Theoret. Comput. Sci. **410**(18), 1648–1665 (2009)
203. O. Regev, Lattice-based cryptography, in *Annual International Cryptology Conference* (Springer, Berlin, 2006), pp. 131–141
204. J. Hoffstein, J. Pipher, J.H. Silverman, NTRU: A ring-based public key cryptosystem, in *International Algorithmic Number Theory Symposium* (Springer, Berlin, 1998), pp. 267–288
205. Y.-L. Gao, X.-B. Chen, Y.-L. Chen, Y. Sun, X.-X. Niu, Y.-X. Yang, A secure cryptocurrency scheme based on post-quantum blockchain. IEEE Access **6**, 27205–27213 (2018)
206. P. Pessl, L.G. Bruinderink, Y. Yarom, To BLISS-B or not to be: Attacking strongswan's implementation of post-quantum signatures, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1843–1855
207. D.J. Bernstein, T. Lange, Post-quantum cryptography. Nature **549**(7671), 188–194 (2017)
208. J. Patarin, Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms, in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 1996), pp. 33–48

209. A. Petzoldt, M.-S. Chen, B.-Y. Yang, C. Tao, J. Ding, Design principles for HFEv-based multivariate signature schemes, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, Berlin, 2015), pp. 311–334
210. J. Ding, B.-Y. Yang, C.-H.O. Chen, M.-S. Chen, C.-M. Cheng, New differential-algebraic attacks and reparametrization of rainbow, in *International Conference on Applied Cryptography and Network Security* (Springer, Berlin, 2008), pp. 242–257
211. W. van der Linde, P. Schwabe, A. Hülsing, Y. Yarom, L. Batina, Post-quantum blockchain using one-time signature chains, Radboud Univ., Nijmegen, The Netherlands, Tech. Rep (2018)
212. A. Foundation, Abelian coin (ABE) – a quantum-resistant cryptocurrency balancing privacy and accountability (2018). https://www.abelianfoundation.org/

Part II
Popular Blockchains

Bitcoin Blockchain System: An Overview of Security and Privacy Aspects

Mauro Conti, Ankit Gangwal, Chhagan Lal, and Sushmita Ruj

1 Introduction

Since the start of the cryptocurrency concept, Bitcoin is the biggest commodity that hit the international market. It facilitates several things altogether which include a decentralized and secure payment system, a technology that facilitates the permanent storage of data, and a tool for verification and auditing of information. A Bitcoin is a virtual currency (thus intangible) facilitating a decentralized peer-to-peer (P2P) network consisting of Bitcoin users. It is not an organization or a product from a specific company, and it is not owned by some government body. The Bitcoin users are not completely anonymous, and it was not developed for malicious or criminal actors—despite this, Bitcoin is mostly in news headlines due to these malicious or criminal actors. As per Satoshi Nakamoto (a presumed pseudonym used by the developer(s) of the Bitcoin system who released its open software in 2009), it is a shared platform where two individuals that do not trust or know each other can transact currency by using Internet facilities. The trust component is facilitated with the help of cryptographic proofs. Also it does not involve a third

M. Conti
Department of Mathematics, University of Padova, Padova, Italy
e-mail: conti@math.unipd.it

A. Gangwal
International Institute of Information Technology, Hyderabad, India
e-mail: gangwal@iiit.ac.in

C. Lal (✉)
Department of Intelligent Systems, Delft University of Technology (TU Delft), Delft, Netherlands

S. Ruj
School of Computer Science and Engineering, University of New South Wales, Sydney, Kensington, NSW, Australia
e-mail: sushmita.ruj@unsw.edu.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_3

party, thus omitting the payment of any intermediate costs. In particular, Bitcoin relies on proof-of-work concept (i.e., a cryptographic protocol) that is used as a consensus protocol in a distributed network of miners to allow users to mine, store, and transact coins. In recent years, Bitcoin has emerged as one of the most used and trusted cryptocurrencies. The Blockchain.com reported 52 million wallets in August 2020 and approximately 75 million in August 2021, i.e., increasing by 28% in 1 year. It also accounted that the number of UTXOs (aka unspent coins) increased from 68 million to 78 million in 1 year. At present, Bitcoins are accepted by a huge number of vendors as a payment method. However, this increase in value and business of the Bitcoin system also attracts adversaries and criminals to attack and use it, and the lack of any monitoring centralized authority further fuels their activities [1]. Recently, authors in [2] presented a detailed study that specifically targets the security and privacy (S&P) features of the Bitcoin system, including its underlying technologies such as blockchain and consensus algorithms. It discusses the existing attack vector which consists of various threats to its users' security and anonymity, as these threats could limit (or threaten) the continuity (or applicability) of Bitcoin usage in real-life scenarios and services. This chapter aims to provide an up-to-date (i) overview of the Bitcoin system, which includes the discussion about its core components (e.g., consensus protocol, transaction and block structure, ledger, and scripting language) and their interactions with each other, and (ii) survey of the key security and privacy threats to the Bitcoin system along with their countermeasures.

1.1 Organization

The rest of this chapter is organized as follows: Sect. 2 provides an overview of the Bitcoin system and discusses the working methodology of its key components and underlying technologies such as proof-of-work (PoW), distributed ledger, scripts, and transaction or wallet addresses. In Sect. 3, we provide an overview of Bitcoin-related security threats along with their possible countermeasures. Section 4 provides a discussion on the privacy aspects and method for its improvement. Finally, we conclude the chapter in Sect. 5.

2 Overview of Bitcoin System

In this section, we present an overview of the Bitcoin system exploring its fundamental concepts and principles. Section 2.1 introduces Bitcoin transactions and blocks, Sect. 2.2 elucidates proof-of-work (PoW), Sect. 2.3 discusses the blockchain, Sect. 2.5 focuses scripting language used in Bitcoin system, and Sect. 2.4 explains Bitcoin address.

2.1 Transactions and Blocks

Unlike the traditional banking's account-based model, Bitcoin uses unspent transaction output (UTXO) model to reflect the balance in *addresses*. In particular, coins are represented as a chain of transactions, while the addresses act as pseudonyms for the users. The key fields of a Bitcoin transaction are shown in Fig. 1, which include Bitcoin protocol version, a flag whose value is related to the witnesses field, nos of inputs, input list, nos of output, output list, and locktime.[1] In the simplest form, each input to a transaction belongs to a distinct user and includes (i) the hash pointer (that serves as an identifier) to a transaction that consists of the output, which we are using as an input to our current transaction; (ii) an index to identify the particular output that we want to spend; (iii) the length of an unlocking script; and (iv) the unlocking script that must satisfy and unlock the previous unspent output. On the other hand, a transaction output mainly includes (i) the length of a locking script; (ii) the locking script that defines the conditions to be met in spending the coins; and (iii) the number of coins being transferred. In the simplest form, an authorization to use an unspent output (as a transaction input) requires the user to provide the public key—to which the unspent output was sent to—along with a corresponding cryptographic signature. We will discuss the locking/unlocking scripts in Sect. 2.5. Multiple inputs are often clubbed in a single transaction. Since there is no concept of *change*, all the input Bitcoins should be spent entirely, i.e., the total value of input Bitcoins in a transaction should match with the total value of its outputs (excluding any transaction fee).

As explained above, the entire input value to a transaction must be spent completely; there may arise a situation when the sender has a higher input value than the required output value. For example, assume Bob has 100 coins in his only Bitcoin address and he wants to transfer 10 coins to Alice. In such scenario, Bob will create a transaction that inputs his 100 coins and sends it 2 output addresses: first, to Alice (10 coins) and, second, to his own new address (the remaining 90 coins) called a *change* address. With this approach, the Bitcoin system implements the idea of handling the change remaining in transactions.

In contrast to the traditional centralized banking system, where transactions are processed, verified, and stored in centralized books, Bitcoin system employs a distributed ledger technology called *blockchain*. In blockchain system, each node keeps a local copy of the transactions and acts as a bank. But this approach of storing same transaction at several network nodes introduces new vulnerabilities, e.g., maintaining a consistent global view of the system. Furthermore, a user (say, Alice) could create two distinct transactions at the same time by using the same set of input coins and send these transactions to two distinct recipients (say, Bob and Carol). Such a malicious attempt to spend the funds twice is called *double-spending*. In case both the recipients independently process the transactions relying only on

[1] It defines the earliest time, in terms of blockchain length, when a transaction becomes spendable.

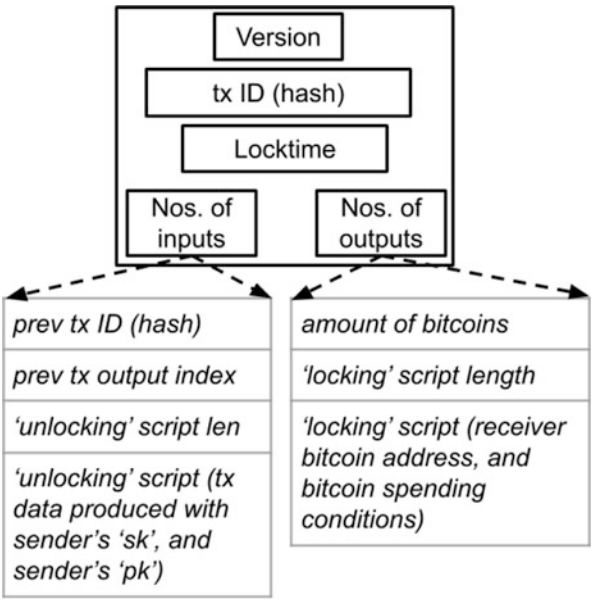


Fig. 1 The key fields in a Bitcoin transaction

their local copy of the chain, and assuming that the verification of transaction is successful, it will lead to an inconsistent blockchain. An approach to avoid such inconsistencies involve (i) distributing the transaction verification process over the network for ensuring the correctness of each transaction and (ii) ensuring that a successfully processed transaction is quickly received by each network node to keep a consistent ledger state across the network. To this end, Bitcoin uses the PoW-based distributed consensus algorithm.

With a distributed transaction verification procedure in the Bitcoin network, participating nodes (called *miners*) verify each transaction and its legitimacy before including it in a data unit (called *blocks*), which is added to the blockchain. When an inconsistent state arises in the blockchain, the nodes opt for the state upon which most miners agree. In this way, a correct and consistent state of the ledger is achieved by a consensus process. Once a block is favorably added in the append-only blockchain (i.e., most miners agree on the block's legitimacy), the miner who first proposed the block is compensated with newly generated coins. As of August 2021, the value of the reward is 6.25 Bitcoins. However, this will decrease to approx 3.12 BTC per blocks after the Bitcoin halving event in 2024. A central authority does not give away the mining rewards. Instead, the rewards come from the block generation procedure, in which the miner inserts a special reward allocating transaction, called coinbase transaction. The miner sets its own wallet address in the output of the coinbase and lists it as the first transaction in the block. If the miner is able to successfully mine the block, then it receives the mining reward through the coinbase transaction it inserted in this block. It is worth highlighting that the reward is the only way to create new coins in Bitcoin system. The reward is halved every

210,000 blocks. The total supply in Bitcoin system is limited to 21 million Bitcoins. The last Bitcoin is likely to be mined around the year 2140 due to the mining reward halving event that takes place every 4 years.

2.2 Proof-of-Work (PoW)

Bitcoin is a decentralized system, and it does not rely on any trusted third party for transaction processing. In particular, the miners in the Bitcoin system communicate over a peer-to-peer network and contribute to construct as well as maintain the blockchain without any central authority. Nevertheless, an individual miner might work against the common goal, crash, or act maliciously. Successfully adding a new block in the blockchain requires the miners in this decentralized system to agree as well as achieve a consensus on it.

Bitcoin uses PoW-based model to achieve its consensus. The approach of using PoW is not unique to Bitcoin; Hashcash [3] is one of the best examples that employed such proof of working but of course for a different purpose. Before verifying a transaction and spreading the news about it over the Bitcoin network, the participating miners have to present evidence of their work in this regard. The work in Bitcoin's PoW-based distributed validation process consists of solving a computationally difficult cryptographic puzzle.

In Bitcoin, new transactions are gossiped over the network. The valid (i.e., error-free, protocol-compliant, and well-formatted) transactions are accumulated to form a block. Now, the miners race to solve the cryptographic puzzle by keeping on computing the block hash together with a *nonce* (a random value that changes in each computation round) until it gets a hash value which is equal to or lower than a predefined *target* value. If a miner finds such a nonce, then the block and the nonce are spread over the network, so that other miners can verify it. Naturally, all the network miners aim to solve the puzzle. In such a situation, the chances of solving the puzzle before anyone else are proportional to the computing power a miner has. It is required that solving the puzzle is hard enough to present the miners with an opportunity to reach to consensus over the appending of next block in the chain in the presence of an adversary with bounded computational power. On the other hand, the puzzle difficulty is set in a way that makes it easy enough to be solvable regularly to enable the miners to make progress. With time, the number of miners or the computing power increases and so should the difficulty of these puzzles. Bitcoin system performs the difficulty adjustment to ensure the constant block generation rate in the face of population or hash rate changes in the network. Authors in [4] present a formal analysis of Bitcoin's target (re)calculation function in the cryptographic setting, i.e., considering the strongest adversary setting trying to subvert the protocol's properties. Specifically, [4] provide a set of mandatory rules considering the way the population (i.e., number of miners or hash rate) evolves in the Bitcoin network. Based on these rules, authors in [5] proposed "Bitcoin backbone with chains of variable difficulty" protocol, which provides a

robust ledger in the presence of an active adversary, if the adversary controls below 50% of network hash rate at any instant of the execution. Moreover, authors in [6] establishes a security threshold that is optimized for the Bitcoin protocol by taking into account several factors like adversary hash rate, benign hash rate, and network delays.

Bitcoin uses SHA256 [7, 8] as the hashing function. Let's consider a simple puzzle, where we must find an SHA256 hash value of the text string "blockchain" that meets the following target criteria:

```
SHA256("blockchain" + Nonce) = A hash that starts with
"000000"
```

In our example, the string is appended with numeric-only nonce values and then hashed together. For different nonce values (values are highlighted for the convenience of the reader), we get the following outcomes:

```
SHA256("blockchain0") = 0xbd4824d8ee63fc82392a6441444
166d22ed84eaa6dab11d4923075975acab938
SHA256("blockchain1") = 0xdb0b9c1cb5e9c680dfff7482f1a
8efad0e786f41b6b89a758fb26d9e223e0a10
SHA256("blockchain2") = 0x8f0532cd22055fb7599aa48f385
01dcd46e61712ab49a02f840f5545830e9260
..
..
SHA256("blockchain10730895") = 0x000000ca1415e0bec568
f6f605fcc83d18cac7a4e6c219a957c10c6879d67587
```

Starting from the nonce value 0, solving our puzzle took 10,730,895 guesses before finding the correct value. Now, let's increase our target by one additional leading zero (i.e., **"0000000"**). It took 934,224,174 guesses to solve the puzzle:

```
SHA256("blockchain934224174") = 0x0000000e2ae7e4240df
80692b7e586ea7a977eacbd031819d0e603257edb3a81
```

Each additional leading zero (i.e., overall value of the target is smaller than before) increases the difficulty. The difficulty of solving the cryptographic puzzle depends on the target value. The higher the target, the more solutions exist, the less difficult the puzzle becomes, and vice versa. Currently, there is no known shortcut to find the solution. Nevertheless, verifying the solution is simple and takes a single hashing operation. In Bitcoin, the term *difficulty* is used in expressing how difficult it is to find a hash satisfying the target value. This difficulty is also artificially changed every 2016 blocks (i.e., once in every 2 weeks' time), and it is adjusted as

$$diff_{\text{next}} = (diff_{\text{prev}} * 2016 * 10)/(actual_time_{\text{last_2016_blocks}}). \quad (1)$$

The difficulty is adjusted to reflect the overall hash rate in the system. If the last 2016 blocks were mined longer than 2 weeks, it suggests that the total network hash rate has decreased. Thus, the difficulty of solving the PoW should also be decreased.

Using a computationally difficult puzzle also helps the Bitcoin system combat the Sybil attack [9], where an attacker creates multiple identities to create influence and hold control. PoW model handles it by shifting the focus of the network influence

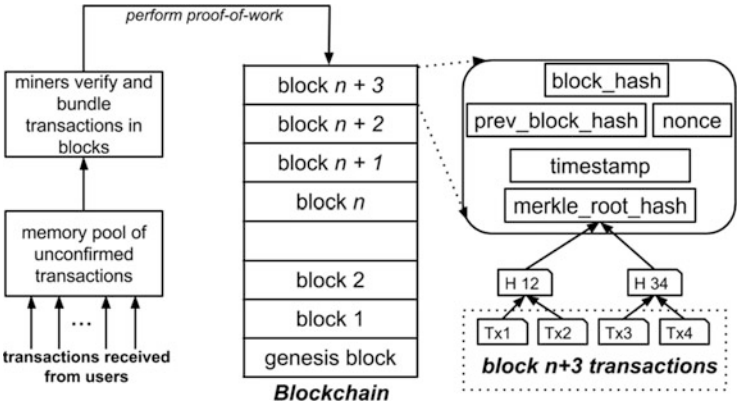


Fig. 2 The linking of the blocks in the blockchain

from identities (that are usually free to create) to the computational power (i.e., hardware that costs money) one has.

2.3 Blockchain

The Bitcoin blockchain is public (i.e., anyone can participate in the consensus process). To store transaction data (in block units), it uses link-list like append-only data structure. Inside a block, the transactions are organized using a Merkle tree [10]. In particular, a Merkle root is incorporated in the block header to secure the transactions in it by forming a hash tree over the TXIDs, i.e., paring and hashing of transaction hashes. In cases where a partner for paring is not available, the transaction's hash is hashed with its own hash. This process is hierarchically repeated until a single hash remains, which is named the Merkle root. The Merkle tree structure helps the clients confirm if a transaction is part of the block. It is done by beginning at the corresponding leaf and traversing the branches toward the root. If the final hash matches the Merkle root, the transactions must be part of the block. To ensure immutability in the storage system, each new block added in the ledger carries the hash of the previous block along with a timestamp. Figure 2 depicts an abstract approach that is used for creating as well as maintaining the blockchain. Adding a new block successfully in the blockchain requires the miners to mine (i.e., verify) a block of transactions by solving the cryptographic PoW puzzle. The idea of storing the hash of the entire block prevents any potential tampering of the transactions stored in a block. Hence, any tampering with any transaction will cause the hash of the block to change, unlinking the block from the subsequent blocks in the chain.

The blockchain keeps on growing due to continuous mining work. The procedure to add a new block is as follows: (i) miners accumulate pending transactions,

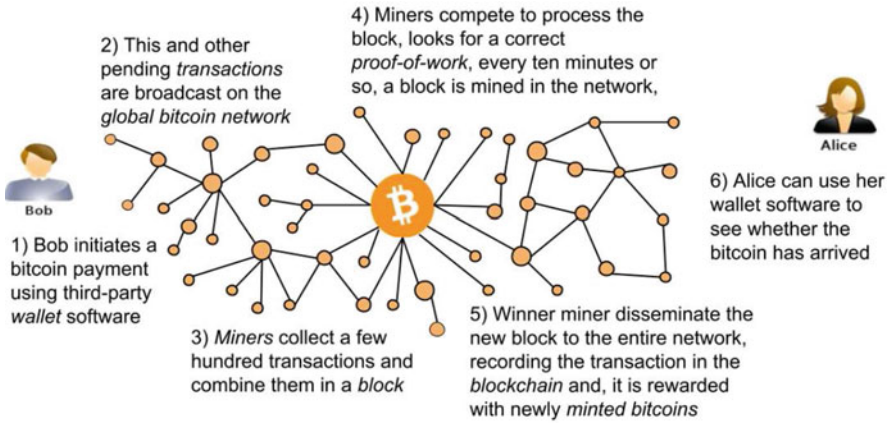


Fig. 3 Transaction processing in Bitcoin network

validate them, and form a block (the decision of forming a block depends on the number of transactions in its memory pool as a block has an upper limit on its size); (ii) miners attempt to find a hash value that satisfies the current target value for the newly formed block; (iii) upon finding a valid hash value, miners broadcast the block together with the discovered PoW solution to the network; (iv) the receiving nodes in the network quickly validate the new block against the proposed solution; and (v) nodes accept it by updating their local copy of the blockchain if the solution is correct; otherwise, they discard the block. Figure 3 presents a simplified view of transaction processing in Bitcoin network.

The distributed nature of consensus protocol creates blockchain forks due to the possibility that more than one miner could solve the PoW at the same time (i.e., two valid next blocks exist in the network). Even if the blocks are not mined exactly at the same time, the propagation of a mined block might get delayed due to network latency, causing the fork. In particular, both the cases lead to genuine forks of same length. These forks are undesirable in the blockchain; a consistent global state of the blockchain is required for correct functionality of the system. Nevertheless, forks exist, and miners are free to select a chain that wants to expand. Now that different miners work to extend different (but valid) forks of the chain based on their current view, the random nature of PoW will eventually cause miners working on one fork to find and broadcast next mined block before the others. This will make their chain longest in the network, and all the other miners will now work to extend it. Figure 4 demonstrates this phenomenon. As it is seen in Fig. 4 that at time T_2 two miners discover (i.e., solve the PoW puzzle) a new block, and it creates a fork in the network. At T_3, three miners discover a new block. Two of these blocks extend the current two forks, while the rest creates a third fork in the network. At T_4, two miners solve the PoW solution and add the new block on two of the forks, thus leaving the network with two forks instead of three. Finally, at T_5, only one miner finds the PoW solution and adds the new block on one of the two existing

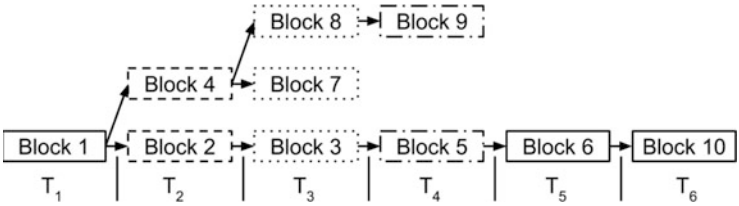


Fig. 4 Blockchain forks

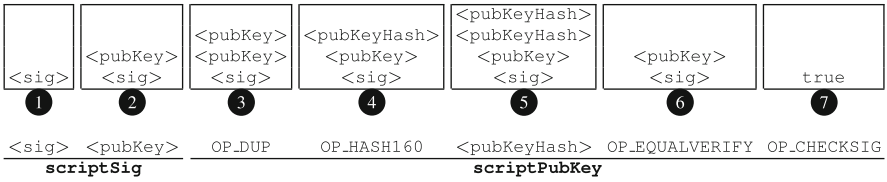


Fig. 5 Converting a public key into a Bitcoin address

forks, leaving it the longest chain in the network. In this way, the forks are resolved over time when new blocks are mined and added to the blockchain. Naturally, the presence of the forks in the blockchain creates room for malicious miners to disturb the functionality of the network. We will see the potential attacks that exploit these forks in Sect. 3.

2.4 Address

In Bitcoin system, a user's wallet address is represented as a string comprising of characters and digits that is publicly shared to receive Bitcoins. Bitcoin addresses are generally produced from the public keys of the users, and they help the recipients to avoid sharing their longer public keys with the senders. Bitcoin addresses are most commonly seen in Bitcoin transactions as the "recipients" of the funds. These addresses can signify a private/public key pair or even a payment script (such as the case of P2SH). For the sake of simplicity, let's consider the case when an address represents a public key. However, in practice, it is not exactly a public key but is derived from it via a one-way function. Figure 5 shows the process of deriving an address from a public key. For a public key K, we compute first its SHA256 hash followed by a RIPEMD160 hash of the outcome. The "double hashing" is called HASH160 in Bitcoin, and it produces a 160-bit (20-byte) hash of the public key. Bitcoin addresses are Base58Check (consists of 58 characters)-encoded public key hash. Base58Check encoding requires a prefix, called the "version byte," which acts as an identifier. For instance, P2PKH-compatible Bitcoin addresses use zero (0x00 in hex) as the prefix. Table 1 lists the version prefix and encoded result's prefix for common address types.

Table 1 Base58Check version and encoded result's prefix for common address types

Type	Version prefix (hex)	Base58Check result's prefix
P2PKH address	0x00	1
P2SH address	0x05	3
Bitcoin testnet address	0x6F	m or n
Private key WIF	0x80	5, K, or L
BIP32 extended public key	0x0488B21E	xpub
Bech32 (SegWit)	0x0303000203 (for checksum only)	bc1

Table 2 Some commonly used instructions in Script

Opcode	Description
`OP_DUP`	Replicates the top item on the stack
`OP_HASH160`	Hashes the top item twice (first SHA256 and then RIPEMD160)
`OP_EQUALVERIFY`	Checks if inputs are equal. If not, returns false and invalidates transaction
`OP_CHECKSIG`	Checks if the input signature is valid for the input public key
`OP_CHECKMULTISIG`	Validates `<m>` input signatures in `<n>` public keys of locking script
`OP_RETURN`	Terminates the execution and marks the transaction as invalid

2.5 Script

Bitcoin transactions provide a certain degree of programmability through scripting to define what exactly a transaction does. In this section, we will elaborate Bitcoin scripting language, *Script*, and explain how these scripts are used in transactions.

The Bitcoin scripting language has similarities with the Forth language. Script is a stack-based language, meaning that each instruction is executed only once and in a linear manner. It supports basic arithmetic operations, basic logic (e.g., *if* and *then*), error throwing, returning early, etc. However, it does not support programming loops, and, thus, is a Turing-incomplete language. The lack of support for the loops was intentional to make it easier to handle and to avoid infinite loops. Nevertheless, it has native support for cryptographic operations, e.g., to compute and verify signatures. Table 2 describes some commonly used instructions in Script.

Upon executing a Bitcoin script, it either executes successfully with no errors or encounters some error. The former is the case when the transaction is considered valid, while the latter invalidates the transaction, which should not be accepted into the blockchain. Script essentially creates a locking mechanism for the transaction outputs. This mechanism has two key components, locking script (i.e., *scriptPubKey*) and unlocking script (i.e., *scriptSig*). A locking script is placed on every output in a transaction to lock it. To use a previous transaction's output as input, the transaction using it must provide the corresponding unlocking script. In the simplest form, the locking script describes a public key, while the unlocking script presents a signature with that public key.

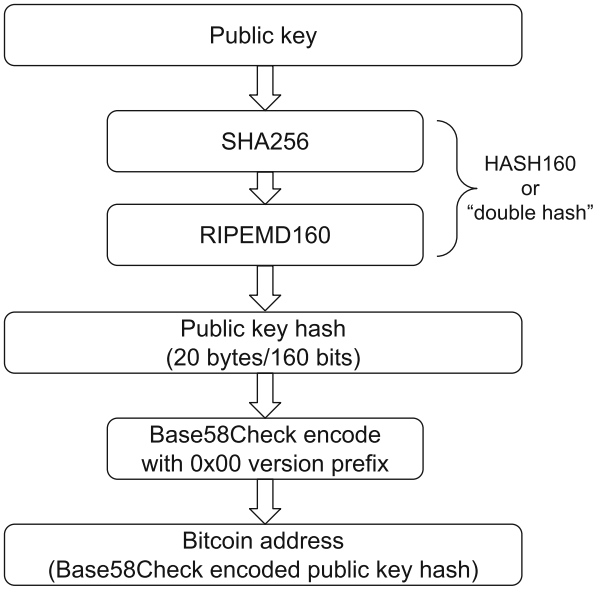


Fig. 6 Illustrative P2PKH script execution on a stack

Script execution: A stack is all we need to execute a Bitcoin script. A script can contain data instructions and opcodes. The data instructions in a script are pushed onto the stack, while opcodes execute some function, often popping the data from the stack. While validating a transaction, the unlocking script is simply concatenated with the locking script, and the resulting script—often called redeeming script—must execute successfully to make the transaction valid. Figure 6 depicts step-by-step execution of Pay-to-Public-Key-Hash (P2PKH) script on a stack.

The first two instructions, i.e., the signature and the public key for verifying that signature, are data instruction. As described above, these instructions are pushed onto the stack. The next instruction, i.e., `OP_DUP`, pushes a copy of the public key onto the top of the stack. `OP_HASH160` pops the top value, computes its hash, and pushes back the results onto stack. Next, we push onto the stack one more data instruction, i.e., the hash of the public key that the sender specified in the referenced transaction. Now, `EQUALVERIFY` checks if the top two values are equal. If not, the script will stop after throwing an error. Assuming that they are equal in our example, meaning that the recipient (and now the sender) used the correct public key, `EQUALVERIFY` will consume the top two data items. Now with a signature and a public key remaining on the stack, we have to verify if the signature is valid. `OP_CHECKSIG` instruction pops those two values and performs the signature verification in a single go. Here, a natural question arises: What is this a signature of? The answer is "the entire transaction," which is the only thing that one can sign in Bitcoin. Thus, `OP_CHECKSIG` pops the signature and the public key and verifies with the help of public key that the signature is a valid signature for the

Table 3 *scriptSig* and *scriptPubKey* for standard Bitcoin scripts

Script name	*scriptSig* (unlocking script)	*scriptPubKey* (locking script)
P2PK	`<sig>`	`<pubKey> OP_CHECKSIG`
P2PKH	`<sig> <pubKey>`	`OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`
P2MS	`OP_0 [<sig>...]`	`<m> [<pubKey>...] <n> OP_CHECKMULTISIG`
P2SH	`[<sig>...] <redeemScript>`	`OP_HASH160 <redeemScriptHash> OP_EQUAL`
NULL DATA		`OP_RETURN <data>`

entire transaction. With nothing left on the stack and provided there were no errors, the script's output will be true indicating that the transaction is valid.

Script allows us to define arbitrary conditions that must be satisfied to spend an output. However, this flexibility is used in a very limited manner, and consequently, there are not many options in terms of what scripts get used. The reason for such limitation is that the Bitcoin nodes maintain a white list of standard acceptable scripts by default and they reject non-listed scripts. Nevertheless, those non-listed scripts are still usable but just harder to get accepted by nodes. Table 3 shows *scriptSig* and *scriptPubKey* for standard Bitcoin scripts, i.e., Pay-to-Public-Key-Hash (P2PKH), Pay-to-Public-Key (P2PK), Pay-to-Script-Hash (P2SH), Pay-to-Multisig (P2MS), and NULL DATA script.

P2PK script locks the output to a public key. It is commonly found in coinbase transactions[2] in the initial blocks in blockchain. P2PK was soon replaced with P2PKH script for the sake of users' convenience. P2PKH allows using addresses instead of a public key; public keys are longer than addresses.

P2MS allows locking an output to multiple public keys and requires signatures from some (or all) of those public keys to unlock it. P2MS is realized using `OP_CHECKMULTISIG`. Due to an implementation bug, `OP_CHECKMULTISIG` pops an extra item from the stack. To avoid the cost of fixing the bug and maintain compatibility, the bug still exists and is considered a feature. The bug is handled by putting an extra dummy data (`OP_0`) in *scriptSig*. P2MS locking script can get larger in size with all the public keys in it, so it is limited to three public keys to avoid storing more data in UTXO set as well as to reduce the transaction cost. The biggest downside of P2MS is that the recipient must send the raw locking script to the sender as there is no P2MS-specific address format.

P2SH allows locking an output to the hash of a script. The original script is provided to unlock the output. It simply allows creating custom "redeem scripts" that are still easy to share with other people. As compared to P2MS, P2SH reduces transaction costs and makes smaller UTXO set. NULL DATA is used to store data

[2] `4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b` —coinbase transaction in the genesis block.

on the blockchain. `OP_RETURN` terminates the execution of the script and marks the transaction as invalid. Thus, any output with a NULL DATA locking script is provably unspendable.

3 Security Threats and Countermeasures to Bitcoin Systems

The first cryptocurrency that got huge popularity is Bitcoin, and it has dominated the cryptocurrency market[3] from the start. Due to its high market share and value, it provides a great motivation to hackers and other malicious entities to identify and exploit different vulnerabilities for financial gains. Therefore, in this section, we present a discussion on most popular security attacks on Bitcoin protocol and system design. We also discuss the countermeasures for the same. These popular security attacks include the double-spending attacks (Sect. 3.1), mining pool attacks (Sect. 3.2), Bitcoin network attacks (Sect. 3.3), and client-side attacks (Sect. 3.4).

3.1 Double-Spending Attacks

A significant security issue in most blockchain systems is the double-spend attack; however, to perform this attack, the attacker needs to own a huge amount of computing resources. In double-spend, the attacker aims to send the same Bitcoins or tokens as inputs in multiple transactions. For example, in Bitcoin, an attacker initiates a new transaction (TX_a) in which it provides an unspent transaction output (say, $UTXO_x$) as input and sends TX_a to a seller to buy some goods. Since TX_a is a valid transaction, it will be added in the pool of transactions waiting to be mined, and some honest miners will then start mining it as part of the next block. However, before the block containing the TX_a is added in the global copy of the distributed ledger, the attacker secretly mines a block by forking the current chain. It then creates another transaction (TX_b) which uses the same $UTXO_x$ as input, thus creating a conflict with TX_a. The TX_b included in the chain fork created by the attacker moves the coins to some address that belongs to the attacker. As the seller is not seeing this forked chain, therefore as soon as it receives adequate (i.e., six) block confirmations for TX_a, it releases the product to the attacker. The attacker continues its work in parallel with the rest of the miners to mine on its private chain to ensure that its chain becomes longer than the public one (by mining the blocks 2 and 3); once it happens, the attacker broadcasts its own chain to the network. Due to the fact that the Bitcoin resolves the fork by using the longest chain rule, the other miners have to agree on the attacker's chain. Thus, TX_b becomes part of the distributed ledger, while TX_b is discarded. In this way, the attacker gets the goods

[3] https://coinmarketcap.com/charts/.

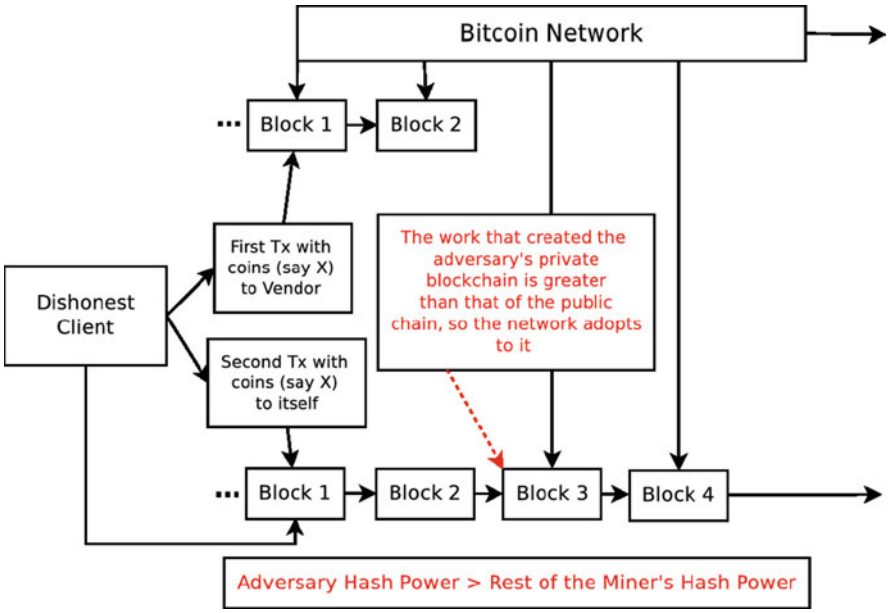


Fig. 7 Double-spend attack

and the funds. The general strategy of hiding a mined block as explained above is also known as block withholding, and it has usage beyond double-spend attack [11]. Figure 7 depicts the working methodology of double-spend attack replicating the above details.

Bitcoin uses PoW (refer to Sect. 2.2 for details) to make double-spend attacks significantly difficult for individual attackers. Due to the difficulty imposed by Bitcoin for adding a new block in the chain, it is recommended that the users wait for at least six confirmations of their transaction commitment, before they consider it completed. It is because each confirmation will add more PoW for the attacker to reverse the transaction. However, in [12], the author concluded based on their evaluation results that double-spend is always a possibility even if the attacker has a hash rate of any percentage and reducing its possibility to zero based on the number of confirmations received is not feasible. In particular, it is shown that an attacker with computing power less than 50% can successfully launch a double-spend by using brute force approach along with enough luck and an attacker with 51% computing power (called 51%attack) will surely get success. It is because in 51% attacks, the attackers have the ability to collect most of the mining rewards and decline or accept transactions of their choice. This will discourage other honest miners, which further helps attackers to increase their hash rate share in the network, thus strengthening their position in the Bitcoin system. Therefore, the 51% attack will possibly destroy the whole Bitcoin network, and thus, it is considered the worst-case scenario. All in all, sellers accepting transactions

immediately, i.e., without waiting for any confirmations, are more likely to be the victim of a double-spend attack. An analysis of double-spend threats against fast payment applications (i.e., where sellers accept transactions without waiting for any confirmations) is presented [13], and the paper also showed that not only the miners but individual users can also cheat by performing double-spend attacks.

Based on the abovementioned discussion about various types of double-spend attacks, it is safe to infer that double-spend always remains a possibility in Bitcoin. Also, the success probability of a double-spend can be increased by combining it with the Sybil attack. An adversary node can perform Sybil attack by creating many fake identities to defraud the system to break its trust and redundancy mechanism. Recently authors in [14] introduced a novel attack model that integrates Sybil and double-spend attacks, and their analysis results show that by introducing propagation delay in genuine blocks with the help of Sybil nodes, an attacker can increase its probability of converting its private chain into main chain. Thus, it can launch a successful double-spend. However, there is also a certain degree of risk associated with performing a double-spend. For example, if unsuccessful, an attacker will not only lose the reward that it could get for privately mining a block, but it will also lose the reward that it could have got if it has used this mining power to mine fairly. Therefore, the attacker should carefully weigh the benefits and the associated risks before launching an attack. Moreover, even a successful double-spend could be caught if a careful analysis and traversing of ledger transactions are done, and if detected, then the attacker could be blacklisted from the network.

3.1.1 Possible Countermeasures

Apart from waiting for a number of confirmations, there are other ways to possibly avoid or detect a double-spend attack in Bitcoin. Authors in [15] present a random mining group selection algorithm to provide certain resistance to double-spend attacks. Specifically, they propose first to create several mining groups by grouping a set of network miners and then provide mining opportunities by randomly selecting a group. In [16], the authors propose and evaluate the following three methods to detect a possible double-spend attack in fast payment applications: (i) listening period, in which the seller associates a specific time period with each received transaction and monitors the network for double-spend attempts during this time window; (ii) inserting observers, in which it extends the listening period concept by inserting a set of nodes called observers in the network which relies all the received transactions to the seller using a single hope link, which helps seller to check more number of transactions while waiting for the timer to expire; and (iii) forwarding double-spend attempts, in which all the Bitcoin miners/peers forward the double-spend transaction attempts to the intended seller instead of discarding such transactions to ensure that the seller gets them on time (i.e., before releasing the goods).

Although achieving a hash rate more than 51% is difficult, the monetary incentives motivate an attacker to create huge mining pools in the network to look

for any chance to get a wrongful award or disrupt the network. To discourage such malicious entities from forming huge mining pools, the author in [17] provides a technique called two-phase proof-of-work (2P-PoW). Next, authors in [18] provide a concept of decentralized non-equivocation contracts that can be used for double-spend detection followed by attacker punishment. The contract requires that the clients lock some amount of Bitcoins in a deposit account before sending a transaction to the seller. Later, if a double-spend done by the client is detected, then cryptographic primitives, namely, accountable assertions, will be used to unlock the coins deposited by the client; thus the malicious clients lose their deposit coins which are used to pay the seller. However, such solutions can be subjected to collusion attacks, where the client and the deposit beneficiary collude that results in transfer of deposit coins to the client's account even when he/she is caught in double-spending. To handle such collusion attacks, authors in [19] propose the concept of fair deposits, which are Bitcoin transactions ensuring that the malicious client will lose the deposit upon double-spend and the victim seller will be compensated. In particular, it extends the assertion scheme provided in [18] by ensuring that the deposit beneficiary can get the client's private key if it double-spends. To this end, a signature created using the client's private key along with another created with the seller's private key is needed to release the deposited Bitcoins to the client's account. A similar approach was also proposed in [16] to address the double-spend attack, in which the participating users create an agreement associated with a safety amount. Recently, authors in [20] propose a method that combines miner's weighted information history and the total calculation difficulty to diminish 51% attack. Furthermore, there are other research efforts that provide solutions to address the double-spend attack such as [21–24].

Since all the existing solutions aim to make the double-attack more difficult for attackers, therefore, fully addressing the issue is still an open research challenge. The simple but powerful way to avoid a double-spend in fast-payment systems is to wait for multiple confirmations before considering a transaction complete. However, for sellers the waiting period should balance the risk and goods delivery time trade-off. Similarly, for attackers there is always a possibility that their attack might fail; thus a trade-off between loss of computing resources and rewards from pre-mined blocks must be considered.

3.2 Mining Pool Attacks

In Bitcoin, to increase the computing resources which increase the chances to win reward by solving the PoW, mining pools (refer to Sect. 3.1 for details) are created. Mining pools are usually managed by the pool manager(s) that share(s) the unsolved work units with its members (i.e., miners). The members calculate and submit partial or full proofs-of-work (PPoWs or FPoWs) to the manager in the form of *shares*. In case of successful mining, the manager disseminates the correctly mined block in the network and receives the associated reward, which is later distributed with other

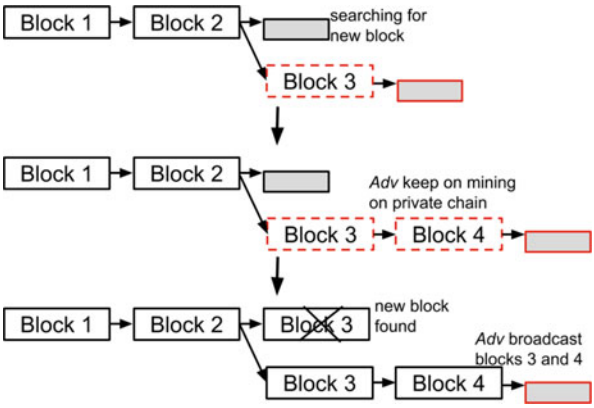


Fig. 8 Selfish mining attack

pool members depending upon the reward sharing algorithm implemented for this purpose. With the increase in the number of mining pools, the attack vector on these pools also increased. For example, a set of malicious members might launch a number of internal or external attacks on the mining pool. The attackers can launch internal attacks by performing malicious activities, to receive an extra portion of the mining reward or to disrupt the pool's normal working process to foil any successful mining attempts. The external attacks are usually performed by other mining pools to intimidate or to attack on a mining pool to drive it away from the mining process. Next we discuss some popular security threats to mining pools in the Bitcoin system.

Authors in [25] use a game-theoretic technique to depict that malicious miners can adopt to a particular type of subversive mining approach called *selfish mining*. Figure 8 shows a possible scenario of selfish mining attack. Particularly, it refers to a scenario where malicious miner(s) hide(s) mining-related information (e.g., withholding a mined block) and reveal(s) it in a selective manner with the following aims: (i) get unfair reward (i.e., more than their actual computing power spent) in the mining pool and (ii) lead other miners to waste their computing resources by sharing wrong information. A 33% of selfish miners could harm the decentralization property of Bitcoin system. An upgrade to selfish mining called *rational mining* [26] shows that even 28% of malicious miners are enough to target the Bitcoin's decentralization property. Similar to selfish mining, malicious members in a mining pool can perform block withholding (BWH) attacks as presented in [27]. In BWH, a member aims to sabotage the revenue of the whole pool by hiding information about a mined block. But the shares which consist of PPoWs are submitted instead of FPoWs. Specifically, the author in [28] presented two types of BWH scenarios known as *Sabotage* and *Lie in wait*. In the former, the attacker ensures that other members lose their Bitcoins by losing its own coins, while in the latter, the attacker does a sophisticated block concealing attack, similar to a selfish mining attack. To show the impact of the abovementioned malicious activities on mining pools, authors in [29] propose a game-theoretic technique showing an attacker can always be well-incentivized if it keeps the malicious behavior long enough. Moreover,

Fig. 9 Blacklisting via punitive forking

it implies that current pool protocols are not secure enough, and if the attacker performs the attack in a systematic way, then pools might lose a lot of revenue within a few months.

Another attack is called Fork After Withholding (FAW) [30] that works similar to BWH, except the adversary causes an intentional fork by propagating the withheld FPoW when another miner (outside the victim mining pool) discovers a genuine block. The possibility that the adversary may get an additional reward from the fork makes the FAW attack more profitable than BWH. Based on the detailed analysis on selfish mining, BWH, and FAW attacks, authors in [31] propose new attacking strategies called Power Adjusting Withholding (PAW) and Bribery Selfish Mining (BSM) to increase the reward of adversaries and also provide possible practical countermeasures for the identified attacks.

The author in [28] presents *pool hopping* attack in which the attacker keeps track of the number of submitted shares for an ongoing mining process for a specific block in the pool to launch a selfish mining attack. The plan is to keep track of the number of shares that have already been submitted to the pool manager, if this number is high, then the attacker knows that he will get a small share of the reward when this block is mined successfully. Once the attacker reaches this conclusion, it might be more beneficial for him to move to another pool or mine alone. Next, the author in [32] proposes the *bribery attack*, in which an attacker pays bribes to other miners to increase its computing resources for a small period. There could be different ways to introduce bribery, such as (i) out-of-band payment; (ii) negative-fee mining pool; and (iii) in-band payment via forking. Moreover, the attacker can also perform a DDoS attack by having a huge amount of computing power (i.e., greater than 50%) that it receives via bribery attack, and it can use it to further launch other attacks like double-spend and punitive forking (please refer to Fig. 9). Moreover, even with less than 50% hash rate, the attacker can try to launch *feather forking* attack.

3.2.1 Possible Countermeasures

An approach to tackle the selfish mining attack is proposed by authors in [33], which suggests a backward-compatible modification in the original Bitcoin protocol. Specifically, upon noticing blockchain forks of same length, the miner forwards this information to its peer nodes, and all will randomly choose a fork to mine on. It will result in a decrease in the ability of the selfish pool that honest miners will extend their forked chain. This work is further extended in [34], in which the concept of Freshness Preferred (FP) is proposed. The idea is to include authenticable

timestamps in the block headers, and the miners will prefer the blocks with the most recent timestamp. To stop malicious miners to use future timestamps, it uses the Random Beacons technique presented in [35]. A more robust approach to address selfish mining requiring no changes in current Bitcoin protocol is presented in [36]. The approach uses a fork-resolving policy, which is used first to identify and then discard the not published blocks on time. Moreover, it recognizes the new blocks that provide a pointer to their predecessors. In this way, if a miner does not publish a privately mined block before a competing block, then it will contribute to both or none of the forks. Another solution to address selfish mining is given in [37]. It forces miners to put out the intermediate blocks (aka in-blocks) that are considered valid even with a less puzzle difficulty; however these blocks do not confer any mining reward. But, with this approach, when a fork occurs, then the miners choose the fork with the highest hash rate instead of the longest chain.

Recently, authors in [38] proposed a scheme called *Silent Timestamping* (ST), in which the pool manager enforces an ordering of the shares. The approach is implemented at pool manager so it does not need networking or any changes at miners' side. First, the authors analyze the optimal behavior of the attacker against the proposed approach and the effectiveness of it against Fork After Withholding (FAW) attack. The analysis depicts that the ST effectively nullifies the attacker's gain and it forces the conversion of the adversary's BWH attack strategy from optimal to suboptimal (i.e., not submit the withheld block or look for limited incentives) or forces the attacker to switch to honest mining. Another solution to prevent block withholding that does not include the usage of timestamps is proposed in [39] called ZeroBlock, in which a time interval named *mat* is used to provide an upper bound on the period till a miner can keep a mined block unpublished. It forces the miners to publish the next block within a specified time so that it is accepted by the other miners in the network.

Due to the serious impact of pool hopping attacks on mining pools, several solutions to address it exist in the literature. For example, authors in [40] provide a methodology for analyzing the pool hopping phenomenon by detecting members that are pool-hoppers. In particular, by analyzing the Bitcoin transactions created by the pools for rewarding its members can be used to determine time epochs where the members have done mining tasks. The analysis leads to revealing the pool-hoppers, and it also shows their hopping and financial patterns. Moreover, authors in [41] apply zero-determinant (ZD) theory that suggests a new mining strategy for pools in which an incentive technique is offered to discourage miners from pool hopping. Specifically, the proposal has the following unique features: (i) fee-free, i.e., if a miner does not hop, then no fee is charged; (ii) wide applicability, i.e., the miner can be employed in prepaid/postpaid methods; and (iii) fairness, i.e., a pool cannot squeeze honest miners for financial gains.

3.3 Bitcoin Network Attacks

Since securing the peer-to-peer (P2P) network is important for correct functioning of the Bitcoin, and the subversion of the networking infrastructure can lead to security threats like Bitcoin theft, mining process manipulations, and DoS. The network-related attacks in the Bitcoin system target the Bitcoin protocol design and its P2P networking infrastructure. Attackers can launch a DDoS (e.g., by using a distributed botnet) on competing miners to take them out of the network which may result in an increase in the hash rate of the attackers. To perform the attack, the attacker exhausts the network resources, which disrupts the network access for honest users. For instance, an attacker throws a large number of bogus transactions to overwhelm an honest miner, which makes the miner to start discarding all the receiving messages including the ones coming from the honest clients. Authors in [42] show the effect of DDoS attacks on the Bitcoin network with a comprehensive empirical analysis. Their analysis involves 142 different DDoS attacks on 40 Bitcoin application services and showed that 7% of operators became victims. Moreover, the authors conclude that most of the DDoS attacks were launched against the Bitcoin exchange services and big mining pools, and the rationale behind is that a successful attack against these will help secure large revenues as compared to attacking an individual miner or smaller pools. Authors in [43] show that attackers can also launch an attack to partition the Bitcoin network (i.e., isolating one or more honest users) by reducing the reputation of honest miners. Next we discuss the *malleability* attack proposed in [44]. It could lead to DDoS attacks. One such example is that by using it an attacker can choke the transaction queue that keeps all the valid transactions waiting to be serviced. Also, the adversary sends bogus transactions with high transaction fees for miners to lure them into processing these transactions with high priority. By the time the miners knew about a bogus transaction, it might be too late as they have already wasted computing power in verifying it and also made the other genuine transactions to wait; thus it also increases the commit latency for genuine transactions [45]. Recently, authors in [46] provide an analysis of the network protocol and Bitcoin core software; they found potential vulnerabilities (that could lead to spoofing and DDoS) in the following three Bitcoin message exchanges: (i) the connection handshake; (ii) GETHEADERS/HEADERS; and (iii) MEMPOOL/INV). The paper also evaluates possible solutions such as introduction of random nonces into network message exchanges, to address the identified weaknesses and vulnerabilities.

Another attack on Bitcoin network known as *timejacking* [47] is performed by exploiting a theoretical vulnerability in handling of Bitcoin timestamps. In particular, at a node an attacker alters the network time counter, forcing it to accept an alternative chain. It can be achieved when the attacker adds multiple fake peers in the network with inaccurate timestamps. However, the attack can be prevented by using the node's system time or restricting acceptance time ranges. Next, there exist attacks on Bitcoin networks, namely, *tampering* attacks [48], in which there exists a possibility of delaying blocks [49] and transactions [50] for a short amount

of time (e.g., for 2 minutes). This attack allows an attacker to alter the information received by Bitcoin nodes, which helps to modify their views of the current state of blockchain.

As a first attempt, [51] present the impact of small- and large-scale routing attacks on the Bitcoin networks. The authors show two vital properties of Bitcoin networks which are as follows: (i) ease in manipulating routing procedure and (ii) increase in centralization due to large hash power and routing area accumulation at certain geographical areas, thus making the routing attacks a practical option for attackers. The authors also observe that an attacker with less than 100 spoofed BGP prefixes might be able to partition approximately 50% of the total network hash rate; this observation is true even in scenarios where the mining pools are largely multi-homed. Moreover, the proposed analysis shows that the attackers that are also the intermediate nodes in the peer-to-peer Bitcoin network could perform different attacks to significantly reduce block propagation time. Finally, for preventing such attacks, authors propose short-term and long-term measures. Next, authors in [52] show that an attacker can exploit the vulnerabilities in the refund policies of Bitcoin payment protocol (i.e., BIP70), to launch a refund attack. BIP70 is considered as a standardized payment protocol in the Bitcoin community; it governs the way in which payments should be done between vendors and customers. Therefore, the majority of wallets use BIP70 protocol for Bitcoin exchange services, and BIP70 is also popular because the ruling payment processors (PPs) like Coinbase and BitPay use it and together they maintain the platform that provides Bitcoin payment services to more than 100,000 market vendors. Particularly, authors in [52] present two types of refund attacks, namely, the Silkroad and Marketplace Trader attacks. The first one raises an authentication vulnerability in BIP70, while the second shows a way to exploit the refund policies in PPs.

3.3.1 Possible Countermeasures

To counter the DDoS attacks that arise due to dissemination of a huge amount of invalid blocks, authors in [53] propose a protocol called proof-of-activity (PoA), whose security is based on PoW and PoS protocols. Another solution to mitigate DDoS attacks is given in [54]; it is based on machine learning methods such as SVM and clustering which works on network traffic that is collected via continuous monitoring of network traffic generated from browsers like Tor or other user-defined web services. Once an anomaly is detected, the part in which it is detected is isolated until the security issue is rectified. Other approaches to protect the Bitcoin network from DDoS attacks include the following: (i) network configuration should be such that it can monitor the ports and prohibit the malicious packets and requests coming from them and (ii) use a third-party intrusion detection system tool that monitors the network traffic to identify malicious patterns.

The network partitioning attacks are one of the major threats to the Bitcoin network security as it heavily relies on connectivity between the network nodes, i.e., higher connectivity yields better security. In this regard, authors in [55] make

two observations: (i) existing connectivity between nodes is not adequately high enough to support optimal security, and (ii) an effort to increase the connectivity will significantly increase the bandwidth usage by the transaction dissemination protocols (TDP), thus making it prohibitively expensive to operate a Bitcoin node. To address this issue, the paper proposes a new TDP, namely, *Erlay*, that provides 40% reduction in the bandwidth usage while providing the current level of connectivity and also ensures that the bandwidth consumption remains constant even if the connectivity increases.

Several methods are highlighted in [47] to avoid timejacking attack. These include system time instead of network time to calculate the upper limit on block timestamps, thus squeezing the acceptable time ranges and allowing trusted peers. Moreover, a peer can be designed to keep many timestamps making it difficult for the attacker to alter all of them. Also, a node's timestamps could be designed in a way that they depend on the blockchain timestamps. In the eclipse attack, an attempt to isolate a user from the Bitcoin network, authors in [56] propose methods to address it. The method suggests that users should possess a list of trusted peer IP addresses to connect with, and in cases where there is a need to connect with peers not in this list, then these new peers should be kept in a *tried* variable list. In particular, a user's connection with its peers depends on a threshold value of a trust factor, and it varies over time. The users can have an intrusion detection system (IDS) to identify the malicious nodes, and such addresses can be banned from future connections.

To address the refund attacks, authors in [52] propose modifications in the payment request (PR) messages, which should include additional information about the client like registered email ID, delivery address, and goods information. Each PP will use a unique payment address, and it is associated with a key that is used for the refund. But the requirement of adding additional information can threaten the client's privacy. The involvement of the client is not required for transaction broadcast, and it is the responsibility of the merchant to process the refund when applicable. If present, then the attacker will be identified by any network node when it learns about the transactions during verification. Specifically, the aim is to provide the merchant with required details such as a set of publicly verifiable and cryptographically proven evidence that can be used to prove that the refund address belongs to the same pseudonymous client who initially authorized the payment. To address the malleability attack, a manual intervention procedure checking the withdrawal transactions is proposed in [57]. Any suspicious pending transactions in a block point to a potential attack. Moreover, all the transactions should have confirmations. Finally, authors in [48] propose solutions to address tampering attacks by forcing the mining node to attach the time spent on mining a broadcasted block. It will help other peers to estimate the average mining time for a block, thus minimizing the possibility where one can spoof by adding delays or performing timestamp tampering. Moreover, replacing static timeouts with dynamic ones is beneficial because mining time can be different for different nodes. All the senders keep track of the peer nodes' IP addresses to which they connect and avoid sending the same advertisement messages to these same peers. To this end, a sender can

keep track of the peers to which it sends its transactions for processing, and this pattern can be analyzed to see if some peer is performing malicious activities or is down. If such a case is detected, then the sender can ask other peers to process its transactions.

3.4 Client-Side Attacks

Above we discussed several security issues in which the attacker targets the Bitcoin systems, protocols, functionalities, and networking infrastructure, to launch an attack. Next we discuss some security threats concerning the client side of the Bitcoin system. With the continuous increase in the usage of Bitcoins in different real-world applications, the number of active users and new users joining the Bitcoin network to transact coins has also been increasing rapidly. Each user possesses a set of private-public key(s) to handle the functioning of their Bitcoin account (a wallet), where the wallet addresses are represented by using the public key addresses to provide pseudonyms. Since these keys are the only way to control a wallet to which these are associated, it is utmost important to use the key management techniques which are simple but secure. To become a Bitcoin user, a wallet application needs to be installed on a device (e.g., laptop or smartphone). To steal the keys, the attacks try to break the wallet by using techniques like system hacking, or installing a malicious software in the client's device. The users need to sign their transactions with a unique signature, and to generate it, the user selects a per-signature random value which is different for each transaction. As shown in [58], repeating this random value might facilitate the secret key computation; moreover, even a partially bitwise equal value is enough to derive a client's private key. Authors in [59] analyze the Bitcoin blockchain and show that some users use the same public key with the same signature nonces many times. More specifically, their report shows that 158 public keys reuse the same signature nonce to generate signatures for multiple transactions, therefore making it feasible to derive the client's secret key. Furthermore, authors in [60] provide an analysis about how broken primitives could affect the Bitcoin system. It presents the fact that there are no migration plans in the existing Bitcoin systems for the broken hash and signature schemes (i.e., the Bitcoin's RIPEMD160, SHA256, and ECDSA protocols remain vulnerable to various security attacks like collision attacks [60]). The paper also shows that the collisions over the main hash or an easy to perform attack on the signature method could lead to Bitcoin theft.

3.4.1 Possible Countermeasures

A manual method called *cold wallet*, which is proposed in [61], is used to protect client wallets. In this method, the client keeps two accounts, one online and the other offline that stores the excess of Bitcoins. The Bitcoin wallet software of the client generates a new private key that is used to transfer this excess amount to

this cold wallet. The idea is that if the system storing the cold wallet is offline, then the hackers will not be able to hack it. Similarly, authors in [62] propose BlueWallet, which is a proof-of-concept-based hardware token instead of private keys for the authorization of transactions. Specifically, it uses offline hardware devices to store the private keys. The hardware token communicates with the device through Bluetooth in which the online wallet is installed, and it reviews the transaction before signing it. In this way, the private key never leaves the BlueWallet, which can only be unlocked with the PIN that is only known to the user. Both the abovementioned techniques improve wallet security, but they also reduce its usability. Since fully ensuring the security of wallets by using the new cryptographic algorithms is still an open issue, therefore, authors in [63] refer that the US government uses multifactor security in their private Bitcoin network; it is done by incorporating biometrics as additional security metric. An optimal threshold DSA technique is presented in [64] to secure private keys. The idea of threshold signatures is borrowed from secret sharing [65]. It splits the private key into shares and a subset of these shares can provide the private key if it is equal and greater than the threshold. Recently, authors in [66] showed a TrustZone-based Bitcoin wallet is highly resistant to attacks like dictionary and side-channel attacks. But its use slows down the read/write operations due to the use of encrypted storage.

4 Privacy Issues and Countermeasures

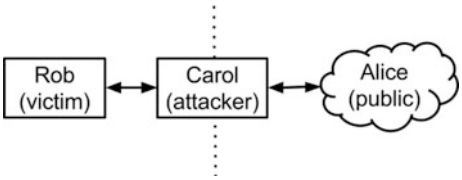
In traditional banking systems, the user privacy is maintained by the third party through limiting the accessible trading information, and the third party also manages the transactions and ledgers. In Bitcoin, on the other hand, although all the transactions are publicly accessible once they are added in the distributed ledger, limited user privacy (i.e., pseudonymity) is maintained by using the public addresses to refer to the users of the transactions to ensure that the transaction history from public ledger will not automatically reveal user identities. To provide pseudonymity, it is recommended to use a new key pair (and thus a different address value) for each transaction. To further support it, the Bitcoin clients use by default the so-called shadow addresses (aka change addresses), which are created when a user performs a transaction to spend some coins and the remaining change of this transaction is sent to his account(s) that is (are) represented by these shadow addresses. However, the use of these public addresses does provide full privacy to the users as it has been already pointed out in various research studies like [67, 68]. In particular, even when the user is hiding behind multiple pseudonyms, it is possible that it can be linked with the addresses that reveal some identifying information [69, 70]. Authors in [71] provide a detailed survey on anonymity and privacy aspects by considering different digital-cash systems. In [2], the authors provide a comprehensive survey that specifically points out the privacy issues, their implications, and possible solutions to improve privacy in the Bitcoin system. In this section, we provide a

survey on the key privacy issues (Sect. 4.1) and their proposed solutions (Sect. 4.2) related to the Bitcoin system and its users.

4.1 Privacy Concerns in Bitcoin System

The Bitcoin system does not claim to support full privacy and it only provides pseudonymity to its users. To support pseudonymity for users while they send their transactions in the network, Bitcoin allows them to use multiple Bitcoin addresses to represent their wallets, and the user device only needs to store the mapping information of a user to its addresses. Since the users are allowed to have multiple addresses (might be associated with the same wallet), the attacker trying to deanonymize these addresses needs to construct a one-to-many mapping. To this end, authors in [72] demonstrated the possibility to link Bitcoin addresses directly to IP data (i.e., IP address of the user of the Bitcoin addresses) in a peer-to-peer network. In particular, the research showed that by leveraging anomalous relaying behavior, it is possible to match the Bitcoin addresses to their possible owner's IP address. Furthermore, authors in [72] study privacy provisions in Bitcoin by analyzing the transactions done on top of both, a real-world Bitcoin implementation, and a simulator that replicates with high-fidelity. The analysis results depict that almost 40% of users' profiles can be identified irrespective of the recommended privacy measures recommended being adopted by these users. In particular, the analysis of transactions stored in the ledger undergoes three pre-processing steps (as shown in Figs. 10, 11, and 12), which result in three graphs that are as follows: (i) transaction graph; (ii) address graph; and (iii) entity or user graph. These graphs are derived based on two heuristics that are generated from Bitcoin's common practices. First is that in multi-input transactions, where these inputs are represented by different Bitcoin addresses, and each input value is lower than the output value (i.e., payment amount), it is usually the case that all these input addresses belong to the same user. Second heuristic is that the change address(es) is (are) usually associated with the same user as the input address(es). However, authors in [68] question the correctness of these heuristics with the fact that possibly the implementation of Bitcoin protocols might change over time and the future system could include more mining pools than individual users or miners.

Fig. 10 Transaction graph



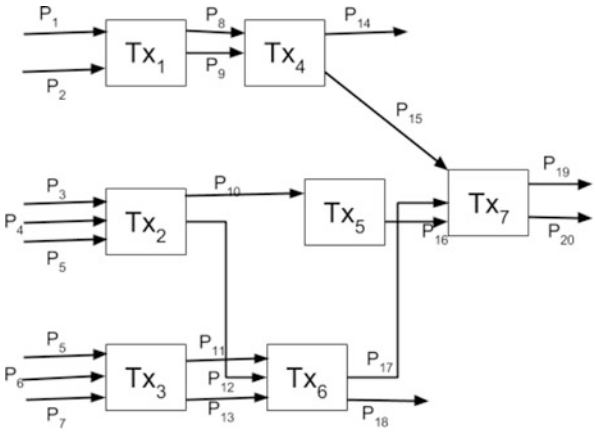


Fig. 11 Address graph

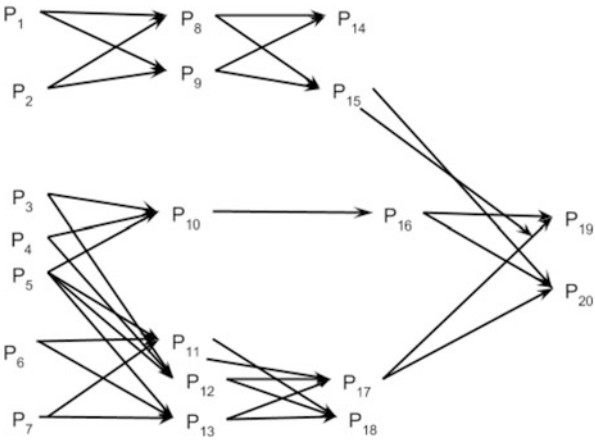


Fig. 12 Entity/user graph

Due to increasing use of Bitcoins to par for illicit products, there also exist few commercial tools such as BitIodine [73, 74] to perform automated analysis on the ledger. Moreover, the governments and law enforcement organizations also develop tools to track down illicit transactions all over the world. Therefore, with the prediction that there will be a large market for Bitcoin ledger analysis, various industrial institutes like Elliptic, Numisight, Skry, and Chainalysis, with specializations in "Bitcoin blockchain analysis" methods, are emerging rapidly. For instance, a GUI-based tool for visual analysis of Bitcoin flow in ledger is proposed in [75]. It helps to graphically monitor transactions, over a time period, starting from the Bitcoin spending transactions (i.e., transaction inputs), originating from a specific source client, all the way to depositing coins into different destination wallets (i.e., unspent transaction outputs). Furthermore, the impact on the privacy of users is analyzed by online tracking of their Bitcoin usage [76] and by analysis of network traffic that elicits from mobile-based wallets [77]. In particular, the authors show how attackers can target the users that perform online payments (in Bitcoins)

to make purchase of products by using the third-party trackers. These trackers are used to extract the required additional information about the purchase made by the client. Later, these transactions can be mapped with user cookies followed by the matching with their real identities; thus, it reveals the user's purchase history. Moreover, if the trackers link the two purchases from the same user, it is possible to link all the Bitcoin addresses and transactions to its owner on the public ledger. It can be made possible with the help of standard tracking tools and blockchain analysis methods. In the past, various research works like [72, 78, 79] have proposed multiple attacks to break the anonymity of the users. In these works, the attacker deploys nodes, namely, *supernode*, in the network which connects with other network peers to listen to transaction messages coming from genuine peers. Authors in [78] utilize the symmetric diffusion of transactions over the network to link the Bitcoin addresses to their possible IP addresses with approximately 30% accuracy. Moreover, using *supernode* to perform linking is simple; thus, it only utilizes minimal information about both, the structured randomness of diffusion and the P2P graph structure, respectively. Therefore, it is safe to assume that higher accuracies can also be achieved by employing sophisticated techniques to analyze network traffic.

4.2 Solutions to Address Privacy Concerns in Bitcoin System

To improve the privacy and anonymity in Bitcoin like cryptocurrency systems, authors in [71] categorize the proposed methods based on the following four outcomes: (i) unlinking the input addresses (i.e., unspent transactions) and output addresses in a transaction performing obfuscation by using techniques like mixing; (ii) unlinking transactions originated from the same user by using link obscuring mechanisms; (iii) hiding the Bitcoins sent or received; and (iv) hiding the IP addresses of users by using VPN tools or Onion Router (TOR). The mixing services provided by anonymous third party use P2P mixing protocols to puzzle the transaction trails to improve the anonymity of Bitcoin users. In the mixing process, the user's Bitcoins are broken into several parts and mixed with random parts of other users. Later, the user will get his Bitcoins from this pool of random Bitcoin parts; thus it will have an entirely new set of Bitcoins. Such a process of mixing weakens the linking possibilities between users and their Bitcoin transactions. The mixing service providers are not inherent entities in the Bitcoin systems, but their usage in Bitcoin is widespread to improve the unlinkability and anonymity. To break the input and output address linking, the author in [80] proposes an idea, namely, *CoinJoin*. The idea is that users make joint payments to form a transaction. In particular, first a set of input and output addresses are agreed between a set of individual users by separately signing a transaction. Second, the signatures are merged which helps achieving obfuscation by shuffling the addresses. Using this process, it can be ensured that a CoinJoin transaction cannot be distinguished from a normal transaction. The approach of CoinJoin is used in various online services

like Shared Coin,[4] in which blockchain.info takes the role of the central server that performs the mixing operations. There are also decentralized implementations for Bitcoin mixing like Coinmux.[5] Moreover, Dash[6]—an altcoin that was formerly known as Darkcoin—offers CoinJoin-like transactions natively [81]. However, there are different issues associated with both centralized and decentralized mixing services. In centralized mixing services, the server will always know the mapping between the users and their coins, while in decentralized mixing services, the participants will be aware of the mapping associated with their allies.

To address the abovementioned issues due to use of centralized and decentralized mixing, authors in [82] propose *CoinShuffle*, in which the participants from a chain of participants add their output addresses encrypted in layers. In particular, at each intermediate step, the corresponding participant removes an encryption layer and adds his/her designated destination(s) in encrypted form. The full list of destinations added by all the participants is received by the last participant who decrypts the final layer from all of them and adds his/her own destination addresses. At this stage, the transactions are constructed and signed via CoinJoin protocol. Note that both, the last peer and the intermediate peers, cannot link individual destinations because they are encrypted. An issue with the above approach of mixing arises when a user initially commits its desire to take part in a joint transaction but later stalls it by not signing. Authors in [91] propose an untrusted payment hub called TumbleBit compatible to Bitcoin, and it allows nodes to perform quick off-blockchain transactions, anonymously with the use of an untrusted third party, namely, Tumbler. Same as the eCash protocol [92], the TumbleBit provides strong guarantees by ensuring that during the mixing process, even the Tumbler cannot link a sender transaction to receiver transaction. The performance evaluation of TumbleBit that mixes the payments of 800 users depicts that TumbleBit is scalable and it supports anonymity and theft resistance. Table 4 provides a brief overview about several efforts that have been done to improve privacy and anonymity in Bitcoin systems along with their advantages and limitations. We direct the interested readers to more comprehensive studies (e.g., [2, 71, 93, 94]) that discuss privacy issues in Bitcoin in detail.

5 Conclusions

In this chapter, we discuss the very basics of the Bitcoin system that are important to understand its core working methodology, different technical components, and underlying network architecture. We further present a list of major security attacks like double-spending and 51% to discuss the vulnerabilities that exist in the Bitcoin

[4] sharedcoin.com.

[5] coinmux.com.

[6] dashpay.io.

Table 4 Privacy and anonymity improvement techniques for Bitcoin

Proposals	Type/class	Distinct features and properties	Advantages	Disadvantages
CoinJoin [80]	P2P	Multi-signature transactions to support privacy	Low transaction fee and theft protection	Anonymity level increases with number of participants, vulnerable to attacks like DoS, Sybil, and intersection
CoinShuffle [82]	P2P	Coordinates CoinJoin transactions via cryptographic mixing	Internal unlinkability, DoS/theft resistance	Lower anonymity, prone to intersection and Sybil attacks
Xim [83]	P2P	Anonymous partnering, mixing with multiple rounds	Internal unlinkability, resistance to Sybil and DoS	Higher mixing time
CoinShuffle++/DiceMix [84]	P2P	CoinJoin-based, higher anonymity	Lower mixing time, resistant to deanonymization attack, support sender anonymity and termination	Prone to DoS and Sybil, lower scalability
ValueShuffle [85]	P2P	Based on CoinShuffle++, provides privacy via confidential transactions	Unlinkability, theft resistance	Prone to DoS and Sybil, lower scalability
Dandelion [86]	P2P	Prevents network-facilitated deanonymization of users via policies	Strong anonymity	Prone to DoS and Sybil
SecureCoin [87]	P2P	Based on CoinParty, supports anonymous and unlinkable transactions	Resistance to sabotage attacks, lower mixing fee, deniability	Prone to DoS, lower scalability
CoinParty [88]	Partially P2P	Based on CoinJoin, uses threshold ECDSA and decryption mixnets	Improves robustness, anonymity, scalability, and deniability, no mixing fee	Partially prone to coin theft and DoS, higher mixing time, honest mixing peers needed
MixCoin [89]	Distributed	Third-party mixing with accountability	Resistance to DoS and Sybil	Partial internal unlinkability
BlindCoin [90]	Distributed	Based on MixCoin, anonymity via blind signature scheme	Internal unlinkability, DoS and Sybil resistance	Partial theft resistance, delays in mixing process

system and can be exploited by an attacker to harm the genuine users. For the discussed security threats, we present a list of precautions that the users should take while transacting over the Bitcoin network and several countermeasures that are proposed by various research studies to counter these threats. Finally, we provide a brief discussion over the privacy and anonymity aspects of the Bitcoin system along with the solutions that are available in the literature to improve these. One of the key aims of this chapter is to make the readers be familiar with the key technologies and major security and privacy issues associated with Bitcoin system, and we believe that many of these technologies combating these issues will hold true for other cryptocurrencies.

References

1. S. Dallyn, Cryptocurrencies as market singularities: the strange case of Bitcoin. J. Cult. Econ. **10**, 1–12 (2017)
2. M. Conti, E. Sandeep Kumar, C. Lal, S. Ruj, A survey on security and privacy issues of Bitcoin. IEEE Commun. Surv. Tutor. **20**(4), 3416–3452 (2018)
3. A. Back, Hashcash-a denial of service counter-measure, 2002
4. J.A. Garay, A. Kiayias, N. Leonardos, The Bitcoin backbone protocol with chains of variable difficulty, *Cryptology ePrint Archive, Report 2016/1048*, 2016, https://eprint.iacr.org/2016/1048
5. J. Garay, A. Kiayias, N. Leonardos, The Bitcoin backbone protocol with chains of variable difficulty, in *Advances in Cryptology—CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 291–323
6. P. Gaži, A. Kiayias, A. Russell, Tight consistency bounds for Bitcoin, in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20 (Association for Computing Machinery, New York, NY, USA, 2020), pp. 819–838
7. D. Eastlake, P. Jones, US Secure Hash Algorithm 1 (SHA1), RFC 3174, pp. 1–21, 2001
8. NIST US, Descriptions of SHA-256, SHA-384 and SHA-512, Tech. Rep., 2001
9. J.R. Douceur, The sybil attack, in *International Workshop on Peer-to-Peer Systems*, pp. 251–260, 2002
10. G. Becker, Merkle signature schemes, merkle trees and their cryptanalysis, Tech. Rep., Ruhr-University Bochum, 2008
11. N.T. Courtois, L. Bahack, On subversive miner strategies and block withholding attack in Bitcoin digital currency, 2014
12. M. Rosenfeld, Analysis of hashrate-based double spending, 2014
13. G. Karame, E. Androulaki, S. Čapkun, Double-spending fast payments in Bitcoin, in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 906–917, 2012
14. S. Zhang, J. Lee, Double-spending with a sybil attack in the Bitcoin decentralized network. IEEE Trans. Ind. Inf. **15**(10), 5715–5722 (2019)
15. J. Bae, H. Lim, Random mining group selection to prevent 51% attacks on Bitcoin, in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 81–82, 2018
16. G.O. Karame, E. Androulaki, S. Capkun, Double-spending fast payments in Bitcoin, in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS'12 (Association for Computing Machinery, New York, NY, USA, 2012), pp. 906–917
17. M. Bastiaan, Preventing the 51%-attack: a stochastic analysis of two phase proof of work in Bitcoin, 2015

18. T. Ruffing, A. Kate, D. Schröder, Liar, liar, coins on fire! penalizing equivocation by loss of Bitcoins, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'15, pp. 219–230, 2015
19. X. Yu, M.T. Shiwen, Y. Li, R. Deng Huijie, Fair deposits against double-spending for Bitcoin transactions, in *2017 IEEE Conference on Dependable and Secure Computing*, pp. 44–51, 2017
20. X. Yang, Y. Chen, X. Chen, Effective scheme against 51% attack on proof-of-work blockchain with history weighted information, in *IEEE International Conference on Blockchain*, pp. 261–265, 2019
21. G.O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, S. Čapkun, Misbehavior in Bitcoin: A study of double-spending and accountability. ACM Trans. Inf. Syst. Secur. **18**(1), 1–32 (2015)
22. A. Dmitrienko, D. Noack, M. Yung, Secure wallet-assisted offline Bitcoin payments with double-spender revocation, in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17, pp. 520–531, 2017
23. G. Ramezan, C. Leung, Analysis of proof-of-work-based blockchains under an adaptive double-spend attack. IEEE Trans. Ind. Inf. **16**(11), 7035–7045 (2020)
24. L. Kovalchuk, D. Kaidalov, A. Nastenko, M. Rodinko, O. Shevtsov, R. Oliynykov, Decreasing security threshold against double spend attack in networks with slow synchronization, in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 216–221, 2019
25. N.T. Courtois, L. Bahack, On subversive miner strategies and block withholding attack in Bitcoin digital currency, 2014
26. S. Pachal, S. Ruj, Rational mining of Bitcoin, in *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, pp. 1–8, 2019
27. S. Bag, S. Ruj, K. Sakurai, Bitcoin block withholding attack: Analysis and mitigation. IEEE Trans. Inf. Foren. Secur. **12**(8), 1967–1978 (2017)
28. M. Rosenfeld, Analysis of Bitcoin pooled mining reward systems, 2011. abs/1112.4980
29. L. Luu, R. Saha, I. Parameshwaran, P. Saxena, A. Hobor, On power splitting games in distributed computation: The case of Bitcoin pooled mining, in *2015 IEEE 28th Computer Security Foundations Symposium*, pp. 397–411, 2015
30. Y. Kwon, D. Kim, Y. Son, E. Vasserman, Y. Kim, Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on Bitcoin, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, pp. 195–209, 2017
31. S. Gao, Z. Li, Z. Peng, B. Xiao, Power adjusting and bribery racing: Novel mining attacks in the Bitcoin system, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, pp. 833–850, 2019
32. J. Bonneau, Why buy when you can rent? Bribery attacks on Bitcoin-style consensus, in *Financial Cryptography and Data Security Workshops, FC 2016, BITCOIN, VOTING, and WAHC* (LNCS, 2016), pp. 19–26
33. I. Eyal, E.G. Sirer, Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018)
34. E. Heilman, One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner (poster abstract), vol. 8438, pp. 161–162, 2014
35. M.O. Rabin, Transaction protection by beacons. J. Comput. Syst. Sci. **27**(2), 256–267 (1983)
36. R. Zhang, B. Preneel, Publish or perish: A backward-compatible defense against selfish mining in Bitcoin, pp. 277–292, 2017
37. R. Zhang, Broadcasting intermediate blocks as a defense mechanism against selfish-mine in Bitcoin, *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 518, 2015
38. S. Chang, Y. Park, Silent timestamping for blockchain mining pool security, in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, 2019
39. S. Solat, M. Potop-Butucaru, Brief announcement: Zeroblock: Timestamp-free prevention of block-withholding attack in Bitcoin, pp. 356–360, 2017
40. M. Belotti, S. Kirati, S. Secci, Bitcoin pool-hopping detection, in *4th IEEE International Forum on Research and Technology for Society and Industry (RTSI)*, pp. 1–6, 2018

41. H. Shi, S. Wang, Q. Hu, X. Cheng, J. Zhang, J. Yu, Fee-free pooled mining for countering pool-hopping attack in blockchain. IEEE Trans. Depend. Secure Comput., 1–1, 2020
42. M. Vasek, M. Thornton, T. Moore, Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem, in *Financial Cryptography and Data Security* (LNCS, 2014), pp. 57–71
43. T. Neudecker, P. Andelfinger, H. Hartenstein, A simulation model for analysis of attacks on the Bitcoin peer-to-peer network, in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1327–1332, 2015
44. C. Decker, R. Wattenhofer, Bitcoin transaction malleability and mtgox, in *Computer Security - ESORICS 2014*, ed. by M. Kutyłowski, J. Vaidya, (Springer International Publishing, Cham, 2014), pp. 313–326
45. M. Andrychowicz, S. Dziembowski, D. Malinowski, Å. Mazurek, How to deal with malleability of Bitcoin transactions, 2013
46. J. Tapsell, R. Naeem Akram, K. Markantonakis, An evaluation of the security of the Bitcoin peer-to-peer network, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1057–1062, 2018
47. A. Boverman, Timejacking and Bitcoin, Mar. 2011. [Online]. Available: http://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html
48. A. Gervais, H. Ritzdorf, G.O. Karame, S. Capkun, Tampering with the delivery of blocks and transactions in Bitcoin, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS'15, pp. 692–705, 2015
49. Bitcoin Dev Commit, 4831, net: Better askfor request management, 2015. https://github.com/bitcoin/bitcoin/pull/4831
50. A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, B. Bhattacharjee. Discovering Bitcoin's public topology and influential nodes, 2015
51. M. Apostolaki, A. Zohar, L. Vanbever, Hijacking Bitcoin: Routing attacks on cryptocurrencies, in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 375–392, 2017
52. P. McCorry, S. Shahandashti, F. Hao, Refund attacks on Bitcoin's payment protocol, pp. 581–599, 2017
53. I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, Proof of activity: Extending Bitcoin's proof of work via proof of stake, *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 452, 2014
54. P. Camelo, J. Moura, L. Krippahl, Condenser: A graph-based approachfor detecting botnets, 2014
55. G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, I. Beschastnikh, Erlay: Efficient transaction relay for Bitcoin, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, pp. 817–831, 2019
56. E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on Bitcoin's peer-to-peer network, in *Proceedings of the 24th USENIX Conference on Security Symposium*, ser. SEC'15 (USENIX Association, 2015), pp. 129–144
57. U.A.Y. Mate, How to identify transaction malleability attacks (2015)
58. N.A. Howgrave-Graham, N.P. Smart, Lattice attacks on digital signature schemes. Des. Codes Cryptogr. **23**(3), 283–290 (2001)
59. J.W. Bos, J.A. Halderman, N. Heninger, J. Moore, M. Naehrig, E. Wustrow, Elliptic curve cryptography in practice, in *Financial Cryptography and Data Security* (LNCS, 2014), pp. 157–175
60. J.J. Hoch, A. Shamir, On the strength of the concatenated hash combiner when all the hash functions are weak, *IACR Cryptol. ePrint Arch.*, vol. 2008, p. 75, 2008
61. M. Draupnir, Bitcoin cold storage guide. Available: https://www.weusecoins.com/bitcoin-cold-storage-guide/, March, 2016
62. T. Bamert, C. Decker, R. Wattenhofer, S. Welten, Bluewallet: The secure Bitcoin wallet, in *Security and Trust Management*, ed. by S. Mauw, C.D. Jensen (Springer International Publishing, Cham, 2014), pp. 65–80
63. E. Benli, I. Engin, C. Giousouf, M. Ulak, S. Bahtiyar, Biowallet: A biometric digital wallet, 2017

64. R. Gennaro, S. Goldfeder, A. Narayanan, Threshold-optimal dsa/ecdsa signatures and an application to Bitcoin wallet security, in *Applied Cryptography and Network Security*, ed. by M. Manulis, A.-R. Sadeghi, S. Schneider (Springer International Publishing, Cham, 2016), pp. 156–174
65. A. Shamir, How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
66. M. Gentilal, P. Martins, L. Sousa, Trustzone-backed Bitcoin wallet, in *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, pp. 25–28, 2017
67. D. Ron, A. Shamir, Quantitative analysis of the full Bitcoin transaction graph, in *Financial Cryptography and Data Security*, ed. by A.-R. Sadeghi, (Springer, Berlin, Heidelberg, 2013), pp. 6–24
68. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, S. Savage, A fistful of Bitcoins: Characterizing payments among men with no names, in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13 (Association for Computing Machinery, 2013), pp. 127–140
69. A. Baumann., B. Fabian, M. Lischke., Exploring the Bitcoin network, in *Proceedings of the 10th International Conference on Web Information Systems and Technologies - Volume 2: WEBIST*, INSTICC (SciTePress, 2014), pp. 369–374
70. M. Vasek, T. Moore, There's no free lunch, even using Bitcoin: Tracking the popularity and profits of virtual currency scams, in *Financial Cryptography and Data Security*, ed. by R. Böhme, T. Okamoto (Springer, Berlin, Heidelberg, 2015), pp. 44–61
71. M.C. Kus Khalilov, A. Levi, A survey on anonymity and privacy in Bitcoin-like digital cash systems. IEEE Commun. Surv. Tutor. **20**(3), 2543–2585 (2018)
72. P. Koshy, D. Koshy, P. McDaniel, An analysis of anonymity in Bitcoin using p2p network traffic, in *Financial Cryptography and Data Security* (LNCS, 2014), pp. 469–485
73. M. Spagnuolo, F. Maggi, S. Zanero, BitIodine: Extracting Intelligence from the Bitcoin Network, in *Financial Cryptography and Data Security* (LNCS, 2014), pp. 457–468
74. M. Conti, A. Gangwal, S. Ruj, On the economic significance of ransomware campaigns: a Bitcoin transactions perspective. Elsevier Comput. Secur. **79**, 162–189 (2018)
75. G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, R. Tamassia, Bitconeview: visualization of flows in the Bitcoin transaction graph, in *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8, 2015
76. S. Goldfeder, H. Kalodner, D. Reisman, A. Narayanan, When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies, 2017
77. F. Aiolli, M. Conti, A. Gangwal, M. Polato, Mind your wallet's privacy: identifying Bitcoin wallet apps and user's actions through network traffic analysis, in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1484–1491, 2019
78. A. Biryukov, D. Khovratovich, I. Pustogarov, Deanonymisation of clients in Bitcoin p2p network, in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, pp. 15–29, 2014
79. A. Biryukov, I. Pustogarov, Bitcoin over tor isn't a good idea, in *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, ser. SP '15 (IEEE Computer Society, 2015), pp. 122–134
80. G. Maxwell, Coinjoin: Bitcoin privacy for the real world, Mar. 2013. Available: https://bitcointalk.org/index.php?topic=279249.0
81. E. Duffield, K. Hagan, Darkcoin : Peer to peer crypto currency with anonymous blockchain transactions and an improved proof of work system (2014)
82. T. Ruffing, P. Moreno-Sanchez, A. Kate, Coinshuffle: Practical decentralized coin mixing for Bitcoin, in *Computer Security - ESORICS 2014* (Springer International Publishing, Cham, 2014), pp. 345–364
83. G. Bissias, A.P. Ozisik, B.N. Levine, M. Liberatore, Sybil-resistant mixing for Bitcoin, in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, ser. WPES '14 (ACM, 2014), pp. 149–158
84. T. Ruffing, P. Moreno-Sanchez, A. Kate, P2p mixing and unlinkable Bitcoin transactions, 2017

85. T. Ruffing, P. Moreno-Sanchez, Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in Bitcoin, in *Financial Cryptography and Data Security Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA* (LNCS, 2017), pp. 133–154
86. S. Bojja Venkatakrishnan, G. Fanti, P. Viswanath, Dandelion: Redesigning the Bitcoin network for anonymity. Proc. ACM Meas. Anal. Comput. Syst. **1**, 22:1–22:34 (2017)
87. M.H. Ibrahim, Securecoin: A robust secure and efficient protocol for anonymous Bitcoin ecosystem. Int. J. Network Secur. **19**, 295–312 (2017)
88. J.H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, K. Wehrle, Coinparty: Secure multi-party mixing of Bitcoins, in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '15 (ACM, 2015), pp. 75–86
89. J. Bonneau, A. Narayanan, A. Miller, J. Clark, J.A. Kroll, E.W. Felten, Mixcoin: Anonymity for Bitcoin with accountable mixes, in *18th International Conference, FC 2014* (LNCS, 2014), pp. 486–504
90. L. Valenta, B. Rowan, Blindcoin: Blinded, accountable mixes for Bitcoin, in *Financial Cryptography Workshops*, 2015
91. E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, S. Goldberg, Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub, 2017
92. D. Chaum, Blind signatures for untraceable payments, in *Advances in Cryptology*, ed. by D. Chaum, R.L. Rivest, A.T. Sherman (Springer US, Boston, MA, 1983), pp. 199–203
93. R. Henry, A. Herzberg, A. Kate, Blockchain access privacy: Challenges and directions. IEEE Secur. Privacy **16**(4), 38–45 (2018)
94. F. Tschorsch, B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. **18**(3), 2084–2123 (2016)
95. Bitcoin Mailing List - Peter Todd about how many pools use an additional relay network, 2015. http://sourceforge.net/p/bitcoin/mailman/message/34152876/
96. T. Bamert, C. Decker, R. Wattenhofer, S. Welten, Bluewallet: The secure Bitcoin wallet, in *Security and Trust Management*, ed. by S. Mauw, C.D. Jensen (Springer International Publishing, Cham, 2014), pp. 65–80
97. E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, S. Goldberg, Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub, 2016. http://eprint.iacr.org/2016/575

The Ethereum Blockchain: Implementation and Security Aspects

Alessandro Brighente, Mauro Conti, and Andrea De Salve

1 Introduction to Ethereum

Ethereum is a permissionless blockchain whose currency is named *ether* and represents a second-generation blockchain technology. The Ethereum blockchain was first introduced in late 2013 in a white paper [4]. It's main objective was to provide a blockchain technology allowing for more than just a simple money exchange. In fact, the aim was to develop a scripting language for application development that could include off-chain data (such as stocks and information on properties) in the blockchain. Therefore, compared to the Bitcoin blockchain, Ethereum allows the creation and execution of decentralized applications and smart contracts. The technical definition of the Ethereum blockchain was first introduced with the Yellow Paper [32]. Since then, multiple releases and forks succeeded, in particular in response to the DAO attack [27].

Ethereum currently uses a Proof of Work (PoW) consensus algorithm to validate blocks. That is, anyone willing to add a new block to the chain must solve a computationally intensive puzzle to prove that computational resources have been spent. However, this is going to be replaced with Proof of Stake (PoS) in the Ethereum 2.0. In this chapter we focus on the current version of Ethereum, describing Ethereum's Ethash PoW algorithm. We also provide an overview of the

A. Brighente (✉)
Department of Mathematics, University of Padova, Padova, Italy
e-mail: alessandro.brighente@unipd.it

M. Conti
Department of Mathematics, University of Padova, Padova, Italy
e-mail: conti@math.unipd.it

A. De Salve
ISASI - National Research Council (CNR), Pozzuoli, Italy

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_4

main changes to the Ethereum network that will be introduced with Ethereum 2.0. At the time of writing, Ethereum 2.0 uniquely updated the consensus algorithm and introduced shard chains. The main data structure and the structure of its components (such as the Ethereum Virtual Machine and the smart contracts) will be unaltered and are described in depth in this chapter.

2 Overview of the Ethereum Platform

In this section, we describe the building blocks of the Ethereum blockchain, focusing on both design and implementation aspects. We first describe the blockchain structure in Sect. 2.1. We then present the mining algorithm in Sect. 2.2. We then describe smart contracts in Sect. 2.3 and the relative data types in Sect. 2.4. In Sect. 2.5 we describe the functions that can be executed by smart contracts, and in Sect. 2.6 we show the set of resources that they can exploit together with tokens in Sect. 2.7. Lastly we describe the Ethereum Virtual Machine in Sect. 2.8.

2.1 Blockchain and Transactions

The Ethereum blockchain can be described via a state machine where states are composed of *account* objects and transitions (here named *messages*) are direct transfer of assets and/or information between accounts. The blockchain starts with an origin state, and subsequent states are generated according to the messages exchanged between accounts. Transactions are therefore given by *valid* arks between two accounts. Due to the non-feasibility of particular state transitions, validity checks ensure that the resulting state machine only transitions towards feasible states.

Each block in the blockchain is composed of multiple transactions, which are stored together with the previous block and an identifier for the final state. Blocks have the function of a ledger, reporting information between parties involved in multiple transactions. Two blocks are connected by means of a hash function. In particular, each block contains the hash of its generating block, referred to as *parent block*. To prevent the creation of stale blocks, besides the hash of the parent block, also the hash of the uncle blocks is written inside each block.

2.1.1 Accounts

Ethereum includes two types of accounts: Externally Owned Account (EOA) and Contract Account (CA). EOA are directly controlled by users and are therefore controlled via a private key. They contain no code, and a user can send messages from an EOA by creating and signing a transaction. CA are controlled by a smart

contract and therefore by the contract code. Each time a CA receives a message, its code is activated, allowing to read and write internal storage, send other messages, or create other contracts. Both EOA and CA are identified by a 20-byte address, and it consists of *four* fields: *(i)* a scalar value (nonce) equal to the number of transactions sent/contracts created from this account which ensures that each transaction is processed singularly; *(ii)* the account's balance measured in *ether* (ETH), i.e. the Ethereum's cryptocurrency used to pay transaction fees; *(iii)* the account's contract code in the case of CA; and *(iv)* the account's storage, which is empty by default.

2.1.2 Ethereum's Message

Messages in Ethereum are virtual objects which are never serialized, and only exist within the Ethereum execution environment. The difference between messages and transactions resides in the fact that a message can only be issued by a CA. Messages are created when a contract executes the *call* function, producing and executing a message. The receiving party is then triggered to execute its own code. Each message contains the address of the sender; address of the receiver; amount of ether to be transferred; an optional data field; and a Start Gas value. This last value represents the maximum number of computational steps allowed for the message execution. Note that messages are not signed, as contracts are not provided with a key. Furthermore, messages exchanged by CA are not stored into the blockchain. However, messages can be included into the blockchain subsequent to reception and signature by a EOA.

Messages are similar to Bitcoin's transactions. However, they provide a richer set of functionalities. In fact, Ethereum's messages can contain data, therefore reserving space for the eventual transfer of assets. Furthermore, messages can be created by both EOA and CA, providing a higher level of automation in assets transfer. Lastly, in case of CA, upon receiving a message, the account has the possibility to return a response. The functionalities provided by the messages enable the *first class citizen* property of Ethereum, i.e. EOA and CA have the same power in sending messages and creating contracts. This allows parties involved in message exchange to neglect the information regarding which type of account they are communicating to.

2.1.3 Transactions

Transactions in Ethereum are the signed data packages containing a message to be sent to a EOA. In particular, once the transaction has been created, the host computes the hash of it via SHA3. The result is then signed via Elliptic Curve Digital Signature Algorithm (ECDSA) with the private key associated with the issuing user, allowing any receiver holding the associated public key to verify the identity of the sender. Transactions are of two different types: (i) resulting in message calls and (ii) resulting in contract creation. The latter refers to those transactions whose aim is to create a new CA with its associated code. Both transaction types share a common

structure consisting of the number of transactions issued by the sender (*nonce*), the amount of ETH to be paid to the miner per unit of gas to cover all costs associated with the execution of this transaction (*Gas Price*), the maximum amount of gas to be paid for the execution of this transaction (*Gas Limit*), the 160-bit address of the transaction receiver (*Receiver's Address*), the amount of ETH to be transferred to the receiver (*Value*), and the signature needed to verify the sender's identity (*Signature*).

The difference between the structure of the two types of transactions is given by a single field. In particular, contract creation transaction includes init, i.e. the Ethereum Virtual Machine (EVM) code needed for the CA initialization. A message call transaction instead contains the data, i.e. the input data of the message call.

The *Gas Limit* is computed by the Ethereum client and is non-negotiable. This field plays an important role in the Ethereum ecosystem. In fact, by limiting the number of steps allowed for a single transaction execution, Ethereum prevents exponential blowups and infinite loops. Each computational step computed by the miner is rewarded by *Gas Price*. Therefore, the overall transaction fee F is computed as $F = Gas\ Limit \times Gas\ Price$. If, during the execution of a transaction, the issuer runs out of gas, all transaction executions except for the fees to be paid are reverted. If a transaction execution halts with some gas remaining, then this is refunded to the issuer.

Transaction fees are selected by the issuer. When an EOA creates and confirms a transaction, the latter is added to the pool of pending transactions, waiting to be executed. In general, the higher the transaction fee, the sooner the transaction is executed. Therefore, the choice of the transaction fee depends on the urgency of its execution. When a new block needs to be formed, transactions in the pool with higher transaction fee get included, while others need to wait. The amount of transactions that can be included in a block depends on the complexity of the transactions. Noticing that each block has a maximum Gas Limit, if all transactions have the same Gas limit, then we can easily find the number of transactions per block by dividing the two class limits. However, as previously stated, this number also depends on the complexity of the transactions. In fact, more complex transactions are associated to higher Gal Limits and therefore reduce the overall number of transactions per block. Therefore, the highest Gas Limit of a transaction is the Gas Limit of a block. However, such transactions are not very common, as they are very expensive for the issuer.

2.1.4 The Ethereum State Machine

Messages and transactions can be seen as transition functions in a state machine. Starting from an origin state, each transaction or message moves the machine to a *current state*, which is accepted as the current canonical version of the Ethereum world. Each state includes information such as account balance, reputation, trust arrangements, and data. The Ethereum state transition function is named *Apply* and receives as input the current state and the message/transaction to be executed. As previously discussed, not all transitions are possible. In fact, transitioning from a

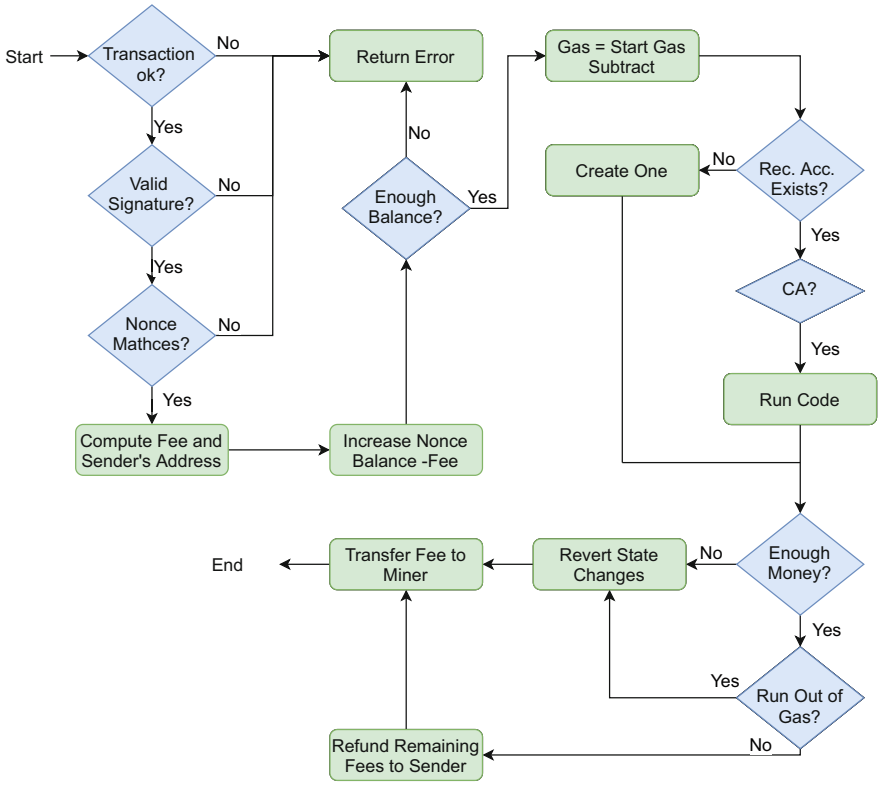


Fig. 1 Block diagram of the *Apply* function

state to another is possible only if certain conditions are met. Figure 1 shows the steps involved in the *Apply* function. The function starts by checking the validity of the issued transaction, verifying whether it is well formed, if the signature is valid, and if the nonce matches the nonce in the sender's account. If all checks are valid, then the fee F is computed. Furthermore, the sender's address is retrieved from the signature. Then, the nonce of the sender is increased, and its balance is decreased by F. The sender's account is checked to see whether it has a sufficient balance to cover all the associated fees. If this is verified, the Gas value is initialized as Start Gas, and a certain quantity of gas per byte to pay for the bytes in the transaction is removed from it. The function then verifies whether the receiver's account exists, and if not, it creates a new one. If the receiver is a CA, then the associated code is executed. The sender being a CA or an EOA, if the sender does not have enough money for the transfer, or in case that the code execution runs out of gas, all state changes are reverted, and the fees are paid to the miner. If instead none of the two are the case, the fund is transferred to the receiver, fees for all remaining gas are returned to the sender, and the fees are transferred to the miners.

By analysing the *Apply* function, we notice that, during its execution, we gain some information that is used immediately after the transaction. This is referred to as the *accrued transaction substrate*, which includes the self-destruct set, the log series, the touched accounts, and the refund balance. In particular, the self-destruct set represents a set of accounts that will be discarded after the transaction's completion. The log series instead represents a series of checkpoints in code execution allowing external onlookers to track the contract calls during the transaction execution. The set of touched accounts contains a list of all the accounts involved in the execution of the transaction, and all those which have not been touched are deleted at the end of the transaction. The refund balance is exploited to partially offset the execution cost, by allowing users to store Gas in contracts while the price is low. By destroying the contract when the gas price is high, users are allowed to use the stored gas to cover the fees of the transaction. However, at the time of writing, a proposal has been made to remove gas refunds in order to avoid populating the blockchain with meaningless contracts and avoid increasing state bloat. All these elements hence form a set of possible final states that the transaction may end up in and are managed while executing the *Apply* function.

2.2 Mining and Consensus Algorithm

As the blockchain is a decentralized system, without a central authority regulating the overall view of the network and its state, we need every participating party to have a mutually agreed and updated view of the system. A consensus algorithm is hence exploited to ensure that every node agrees on the order of transactions. The mutually agreed state accepted by all nodes is the *canonical state*. To be included in a block, a transaction must include a valid transition function from the canonical state to a new valid state. This also justifies the fact that transactions must be included in a specific order inside the block, as state transitions must account for cascades of transitions involving common accounts. The process by which a transaction is included in a block is called *mining*, and the node validating a transaction is called *miner*. At the time of writing, the Ethereum blockchain exploits PoW as a mining mechanism. In PoW, the miner is required to provide a cryptographically secure nonce that proves beyond any reasonable doubt that a certain amount of computation has been required to determine a token value.

2.2.1 The Mining Process

A miner is simply a computer running software but plays a fundamental role in the Ethereum life cycle. In fact, miners ensure that each node in the network has the same view of the transaction order. Miners have the role of solving difficult puzzles to mine transactions and to create blocks. The mining process starts when a user broadcasts a signed transaction request to the overall Ethereum network,

with the aim of including it in one of the successive blocks that will be mined. The mining process proceeds as follows: Each node in the network stores in the local mempool a list of transaction requests that have not been yet committed to a block. Upon collecting several transaction requests, a node aggregates them into a potential block such that the transaction fee obtained from their confirmation is maximized but at the same time the Gas Limit is met. The mining node then executes the *Apply* function shown in Fig. 1. Each miner starts the process of producing the PoW (we will provide more details in Sect. 2.2.2) for the potential block. As soon as the PoW is produced, the miner broadcasts the mined block to every node in the network, including the associated certificate and a checksum of the reached state. Each node receiving the block executes all the included transactions and verifies that the produced checksum matches the one claimed by the miner. If a match is obtained, nodes append the block to their ledger and update the canonical state. Lastly, each node removes all the transactions included in the mined block from their local mempool.

We notice that each transaction is mined only once but is executed at each node. This implies that each node verifies the validity of the transaction, such that each node advances to the same canonical state. When a new node joins the network, it downloads all mined blocks in sequence and executes each transaction in every block verifying every state checksum. By initializing a blank state and running the entire transaction history, the final state obtained is the canonical one.

2.2.2 Ethash: Memory-Intensive PoW

The particular implementation of Ethereum's PoW is called *Ethash*. This algorithm exploits the concept of *difficulty*, i.e. the output of the hashing process must result in a value below a certain threshold. The threshold value is dynamically adjusted such that the average number of blocks produced in a certain period is constant. In particular, if the rate at which blocks are created increases, then the threshold value is decreased to decrease the size of the set of valid hashes. On the other hand, if the rate of block creation decreases, the threshold value increases to increase the size of the set of possible valid hashes. Ethash has been designed to meet specific goals. First of all, Ethash is Application Specific Integrated Circuits (ASIC) resistant, meaning that a specialized hardware and a common computer have the same possibility of mining a block. This offers a free network for everyone to participate in the mining process. ASIC resistance is obtained by consuming nearly all the available memory access bandwidth. Ethash is Graphical Processing Unit (GPU)-friendly, meaning that it has been designed to make as easy as possible to perform mining via GPUs. Lastly, Ethash has been designed to support light clients. In particular, it provides fast mining verifiability and fast startup. However, with a light client, the process of running the algorithm shall be much slower compared to the same execution in a full client, including ASIC implementations.

Instead of requiring high computational power, Ethash is based on a memory-hard problem. This means that the time taken to carry out a computation highly

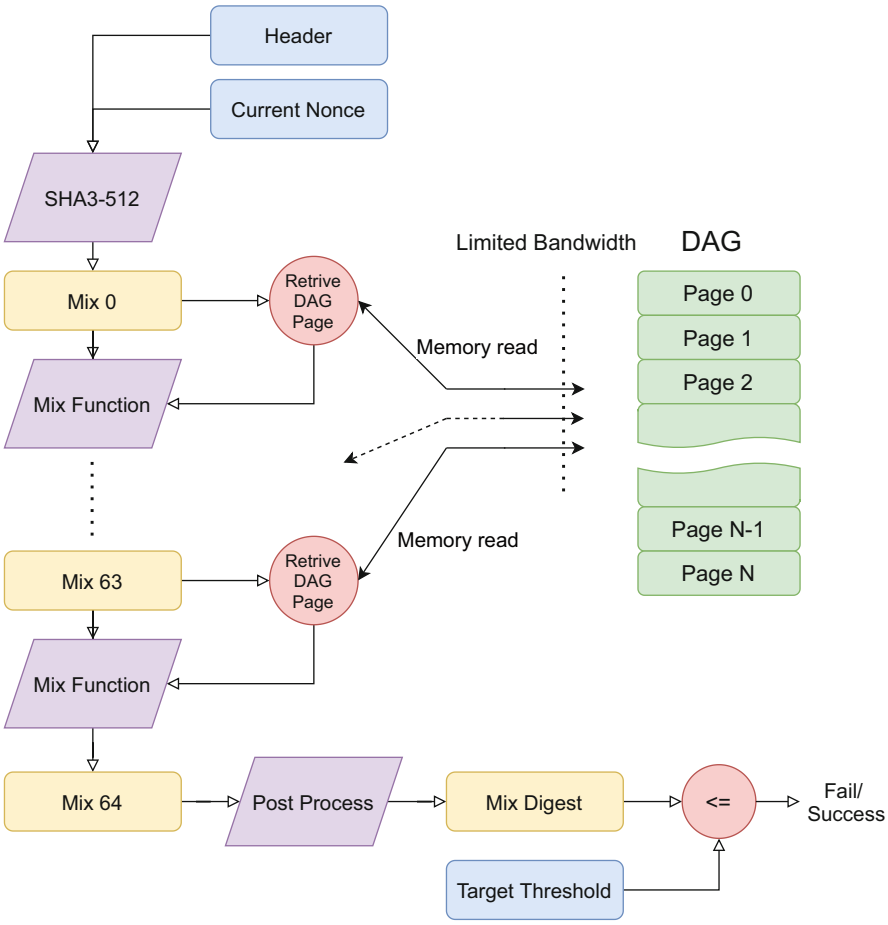


Fig. 2 Block diagram of the Ethash PoW

depends on the amount of memory required to hold the data. Considering the design goals previously discussed, this further improves ASIC resistance, as both large-scale and small-scale miners incur the comparable energy cost on memory. In order to mine a block, miners need to retrieve parts of the random data distributed on the Directed Acyclic Graph (DAG), hash randomly selected transactions on the blockchain, and then return the result from the hashing process. Therefore, each host needs to store the entire DAG in order to be able to retrieve the needed information. Figure 2 shows the block diagram of Ethash steps. Starting from each header in the blockchain's blocks up to the mining point, the node computes a seed. Starting from the seeds, a 16 MB pseudorandom cache is computed, which depends on the block number. A 1 GB size dataset is generated from the cache, such that each item depends only on a small subset of other elements in the cache. Full clients

and miners store this dataset, while light clients only store the cache. During the mining process, random slices of the dataset chosen via the Mix function are hashed together, and verification is performed by exploiting the cache to regenerate the specific pieces of the dataset needed. This implies that verification is possible by uniquely storing the cache. The process is repeated 64 times, until a final hash value is produced. If the obtained value is lower or equal to the target threshold, the mining is successful; otherwise, the process is repeated starting from a new nonce until either a valid hash is produced or the transaction is confirmed by another miner.

2.2.3 Beacon Chain

The Ethereum 2.0, also referred to as *Beacon Chain*, replaces the aforementioned Ethash PoW algorithm with a PoS consensus algorithm. The advantages given by PoS include higher energy efficiency, reduced hardware requirements, and higher decentralization. The main difference is that, in PoS, miners (here also referred to as *validators*) are not competing. In fact, miners are randomly selected to create blocks, whereas, if not selected as block creators, they can validate proposed blocks. This allows for computational power savings, as not all miners are in charge of validating a block. Ethereum users need to stake 32 ETH before being selected as validator. Starting from this amount, they are rewarded with stake while creating or validating legitimate blocks, whereas they lose stake in case of creating or validating a malicious block. At least 128 validators are required to confirm a block. Another important feature of the Beacon Chain is *sharding*, i.e. the process of splitting horizontally the ledger in order to spread the load. Besides increased transaction rate, sharding is also providing other benefits to the Ethereum network. One of such benefits is given by the fact that validators only need to store and run the shard they are validating. This is fundamentally different from the PoW process previously described, in which users needed to store and run the entire blockchain. Shard chains therefore allow validators to be equipped with generic hardware, hence making the network accessible to everyone. A further connected benefit is given by the increased network participation. In fact, as shards need lower hardware requirements, an Ethereum node could potentially be run on a personal laptop or phone. This also has the benefit of an increased security level, as it provides higher degree of decentralization.

At the time of writing, the Ethereum 2.0 has been shipped with reduced functionalities; therefore it will improve over time. In its current form, it does not handle EOA or CA. However, it coordinates the network of shards and stakes and provides extra data to the network. Ethereum plans to add more functionalities to shards, in order for them to provide the same functionalities of the mainnet.

2.3 Smart Contracts

Smart contracts are composable and deterministic computer code which can be deployed on the Ethereum blockchain. The EOA interested in creating a smart contract can send a transaction to the `null` address (`0x0...`) with the code as the data, providing enough ETH to deploy the contract. Once deployed, such code lives on the CA of the Ethereum blockchain denoted by a 42-character hexadecimal address that is derived from both the creator's address and the number of transactions sent from that address (i.e. the *nonce*). The CA is a type of Ethereum account which is not controlled by users, consisting of a *balance* and the *functions* defined by the smart contract itself. Smart contracts have access to storage space where they can store additional data fields used by functions. The data, variables, and functions within smart contracts are encoded and stored in clear on the blockchain to make them public to anyone accessing the blockchain. The execution of a smart contract is initiated by an EOA who send a transaction to the CA, specifying the function to be run. A smart contract can be programmed to interact with other existing smart contracts, to transfer funds to users, or create a new contract. Since smart contracts are stored and executed in the Ethereum blockchain, the creation of a CA has an initial cost which depends on its size. However, they benefit of zero downtime and verifiable computation, and no one can change them.

Even if smart contracts can be programmed using any languages which compile to the *EVM Assembly bytecode*, the two most active and supported languages are *Solidity* and *Vyper*. As the first language developed for writing smart contracts, Solidity is the primary language on Ethereum, providing a complete set of tools and facilities for programmers. Figure 3 depicts the general structure of a smart contract in Solidity. At a high level, a smart contract consists of the data, functions, and environmental variables.

2.4 Data

Solidity is a statically typed language and each data used by the smart contract must be assigned to a type. Table 1 summarizes the main characteristics of the data types supported by Solidity. Regardless of their types, all declared variables are initialized with a default value having all bytes equal to zeros, and the expression `type(X)` can be used to retrieve information about the type of variable X. Among the most common data types, we have the Boolean, signed/unsigned integer, and the byte. An interesting feature of the language is the possibility to define integer and byte types having a specific number of bits (such as `int64` or `bytes32`). The language introduces also the `address` type, which can be used to denote EOA and CA. The `enum` data type allows to create new types which consist of different options represented by subsequent unsigned integer values starting from 0. Fixed-size arrays can be declared by specifying the data type of each element

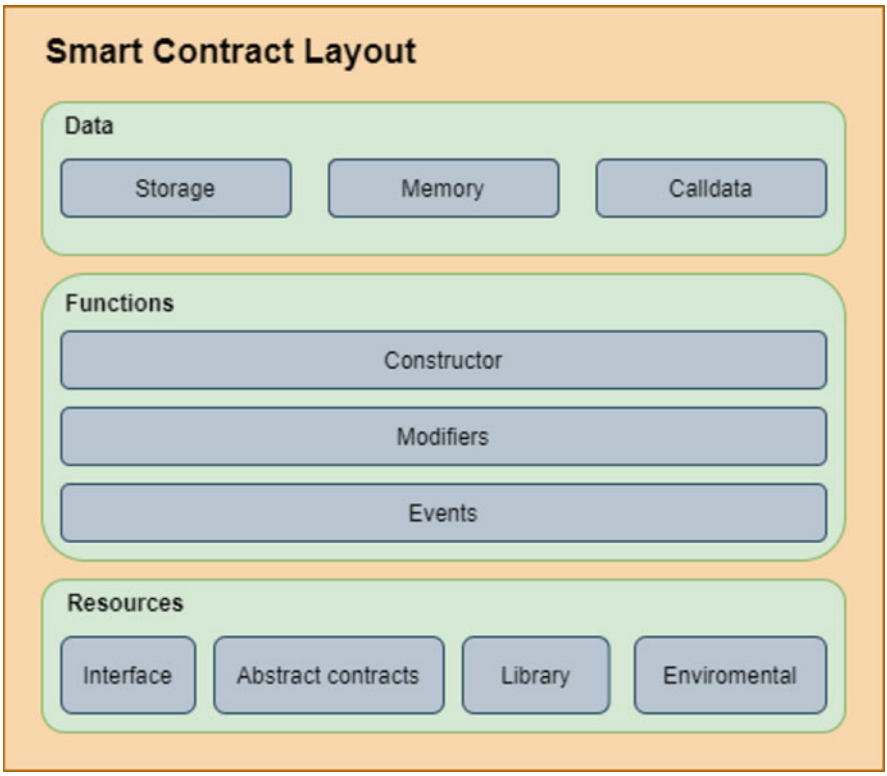


Fig. 3 Anatomy of a smart contract

followed by the number k of elements in square brackets. Each element of the array can be accessed by specifying its position `index` (e.g. $array$[`index`]), such that $0 \leq$`index`$<$ `array.length`, where `array.length` returns the fixed length of the array. Dynamically sized arrays have similar notation, but they do not specify a priori the number of elements k. Furthermore, they provide functionalities for accessing the number of elements (`length`), appending a new element to the array (`push(x)`), and clearing an element $array[x]$ at index x of the array (`delete array[x]`).

The language allows also the creation of special bytes arrays by using the types `bytes` and `string`. In contrast to the `bytes[]` array, the size required by the `bytes` data type is lower because the data are packed in a different way. Instead, the `string` type can be used to denote arbitrary-length strings (UTF-8), and it is equivalent to the type `bytes` but does not allow to push, delete, or access by index an element. Solidity provides a method to define new complex types in the form of `struct`. The `struct` data type can be used to define new composable data types which consist of one or more fields having different types. Each field of a struct a can be read or written by using its name `a.fieldName`.

Table 1 Summary of the data types provided by Solidity

Types	Data types	Size (bits)	Values
Address	`address`	160	Hexadecimal
Boolean	`bool`	8	True, false
Unsigned integer	`uint`$< M >$	M	$< M >= \{8 \cdot n \mid 1 \leq n \leq 32\}$
Signed integer	`int`$< M >$	M	$< M >= \{8 \cdot n \mid 1 \leq n \leq 32\}$
Fixed-size byte	`bytes`$< M >$	M·8	$1 \leq M \leq 32$
Enums	`enum`	8	$\{Val_1, Val_2, \ldots, Val_2 56\}$
Fixed-size arrays	`type[k]`	$k \cdot \text{size}(type)$	$k \geq 0$
Dynamically-sized arrays	`type[]`	–	Type sequence
	`bytes`	–	Bytes sequence
	`string`	–	UTF-8 sequence
Structs	`struct`	–	$\{type_1, type_2, \ldots, type_N\}$
Mapping	`mapping(keyT => valT)`	–	keyT ∈ { *address, bool, int, unit, bytes, string*} valT = *any type*
Contract	`contract`	–	–

The `mapping` data type can be used to create hash tables for associating a value to a key. They are logically initialized in such a way that every possible key is mapped to a value having all bytes equal to zeros. In particular, the keys are not directly stored in a mapping, but they are used as input of the `keccak256` hash function to look up the value. For this reason, it is not possible to enumerate the keys or values stored in a mapping. Eventually, the programmers can extend the native functionalities of Solidity by implementing more sophisticated data structures and functions through libraries or smart contract. In fact, every contract defines its own type which can be used to declare variable of the same contract type (e.g. `MyContract c`). In particular, the variable's name is used to call functions of the smart contract, and it is possible to explicitly convert an address type to the corresponding contract type.

State variables declared in a contract can have also the attribute `constant` or `immutable` to indicate that they cannot be modified after the contract creation. The value of constant variables must be fixed at compile-time, while the value of immutable variable can be assigned when the contract is created, and it cannot be changed anymore.

2.4.1 Data Location

The data used by a smart contract can be located on different areas of the blockchain, depending on their use. The *storage* contains persistent state variables whose lifetime is limited to the lifetime of the smart contract. The *memory* contains short-living variables, such as the data declared and used during a function call. Finally, the *calldata* area is where arguments of external function are stored and it cannot be modified. The previous data locations can be used in function arguments or in assignments to regulate if variables can be passed by value or by reference. In particular, in Solidity the types `address`, `bool`, `uint`, `int`, `bytes`, `contract`, and `enum` are passed by value because an independent copy of the variable is created. Instead, the `struct`, `array`, and `mapping` are reference types, and they must explicitly indicate where the type is stored by using data location.

Array can be stored either in storage (if declared as state variable of the contracts), in memory, or in the calldata area. In the first case, arrays can have fixed or dynamic length, while the memory or calldata arrays cannot be resized. Mapping can be used only as state variables because they have a storage data location. Struct can be located in storage, memory, or calldata area.

2.5 Functions

In Solidity, functions describe a set of instructions that can be executed as a result of a function call. They can be defined, inside and outside the contracts, by using the following signature:

`function`(*parametertype*)[*visibility*][*mutability*][*modifiers*]`returns`(*types*){...}

Functions can accept input parameters of different types and may return an arbitrary number of values as output. In particular, the types and the name of the output values are declared after the `returns` keyword of the function signature, while the statement `return` $(v_0, \ldots, v_n)$ is used at the end of the function to indicate the output values. Solidity supports function overloading by enabling a contract to declare multiple functions with same name but with different types of input parameters. Once defined, a function can be executed by using two different types of call:

Internal This function call can be used only by functions of the same contract instance and it is translated into simple jumps inside the EVM. An internal function call does not require the allocation of new storage because it exploits the current memory.

External This function call is translated in a message call and it must be used when a contract calls a function of another contract. All function arguments have

to be copied in memory and the message call is stored as part of the original transaction initiated by an EOA.

In Solidity, a function may have four different levels of *visibility* with respect to other contracts:

`external` functions are exposed as part of the contract interface, and they can be called externally, i.e. from other contracts and via transaction by EOA.
`internal` functions can only be accessed from the current contract or contracts deriving from it, and they do not belong to the contract interface.
`public` functions are part of the contract interface, and it can be called internally (by using directly the function name) or externally (by using the keyword `this` followed by the function name).
`private` functions are visible only to the contract's instance and they do not belong to the contract interface.

Even if private and internal functions prevent other contracts from executing them, the code executed by such functions is public and visible on the blockchain.

The *mutability* indicator of the functions can be used to provide important information about their behaviours. A `view` function promises to leave the contract status unchanged after their executions and it can recursively call view or pure functions. A `pure` function promises to not read from state variables or modify the contract's state, while a `payable` function is necessary for a contract to receive ether through regular transactions. In particular, a contract can define only two payable functions: a function named `receive` which is executed when some ethers are transferred through the `.send()` or `.transfer()` function and a `fallback`, which will be executed on ether transfer when the `receive` has not been defined. Finally, the compiler automatically creates a public getter functions for all public state variables.

2.5.1 Constructors

A constructor is an optional function which specifies the operations executed when the contract is created. By default, each contract has a default constructor which initializes the state variables of the contract to their default value. The final code of the contract is deployed to the blockchain after the constructor execution and it does not include the constructor code.

2.5.2 Function Modifiers

Modifiers are annotations that can be used in functions to change their behaviour. In general, they are used to check conditions before the execution of a function. For this reason, all variables visible from the function are visible also in the modifiers. They are declared in contracts by specifying the name of the modifier, the code to

be executed, and the symbol `_;`, a special tag used to indicate where the function body must be inserted. Once declared, a function can specify the list of modifiers that should apply to the functions.

2.5.3 Events

A contract is able to notify specific events to the interested audience. In particular, events can be declared in contracts by specifying the event's name and the list of arguments to be stored along with the events. Once declared, events can be emitted by functions using the statement `emit` followed by the event's name and the values of input parameters. Events are implemented by using the logging functionality provided by the EVM the arguments. In particular, events emitted during contract execution are stored in the blockchain as transaction's log associated with the address of the contract.

2.6 Resources

The Ethereum smart contracts can rely on a variety of resources for building complex applications. In this section, we focus on describing some of the most important resources used to improve the behaviours of smart contracts.

Interfaces Interfaces are smart contracts declaring external functions that do not have any implementations. Since interfaces cannot declare constructors or state variables, they cannot be used to create new contract instances. A contract can inherit the properties from multiple interfaces and it has to provide the implementation of all external functions exposed by the interface. The contract that inherits from interfaces can emit events declared by the interfaces and use complex data types (i.e. structs and enum) defined by the interfaces.

Abstract Contracts Abstract contracts can be used to define contracts that do not provide implementation of at least one of their functions. For this reason, an abstract contract cannot be instantiated. As well as interfaces, an abstract contract can declare events and complex data types. A contract which inherits from an abstract contract can emit the events and use the types defined by the abstract contract. In addition, the contract has to provide the implementation of all functions declared and not implemented by the abstract contract.

Libraries A library is defined outside a contract and it consists of functions, complex data types, and events. A library cannot define state variables and cannot implicitly access those of the calling contract, unless they are explicitly provided as input parameters of the function. A library is instantiated as a separated contract, and it is possible to obtain the address of the library by converting the library name to the address type. The contract that uses the library can call the

Table 2 Units and environmental variables available to smart contracts

Units		Variables			
Ether	*Time*	*Block*	*Transaction*	*Message*	*Contract*

`wei`
`gwei`
`ether`
	`seconds`
	`minutes`
	`hours days`
	`weeks`
		`chainid`
		`coinbase`
		`difficulty`
		`gaslimit`
		`number`
		`timestamp`
			`origin`
			`gasprice`
				`data`
				`sender`
				`sig`
				`value`
					`this`
					`selfdestruct`

Table 3 Predefined functions available to smart contracts

Functions			
Encoding/decoding	*Math*	*Crypto*	*Error*

`decode encode`
`encodePacket`
`encodeWithSignature`
	`addmod`
	`mulmod`
		`keccak256`
		`sha256`
		`ripemd160`
		`ecrecover`
			`assert`
			`require`
			`revert`
			`try/catch`

function by using the library name followed by the name of the function. The functions defined by the library can be called internally or using a delegate call.

Environmental Variables, Functions, and Symbols Solidity smart contracts can exploit a large set of already existing units' symbols, variables, and functions during its execution. The units' symbols and the variables are summarized in Table 2. In particular, the default unit of measure for ether is the `wei` (which is equal to 10^{-18} ETH), but smart contracts can also use `gwei` and `ether`. The basic unit of time used by Solidity is the seconds, but smart contracts can also represent time values in minutes, hours, days, and weeks.

As shown in Table 2, smart contracts can rely on a number of global variables to access information about the blocks, transactions, messages, and contracts. In particular, the keywords `block` and `tx` are used to access information related to the current block (e.g. the timestamp and the number of the current block) and transaction (e.g. the sender and the gas price), respectively. Furthermore, a smart contract function is also able to obtain complete information about the current message call by using the `msg` the variable.

Instead, Table 3 summarizes the built-in utility provided by Solidity to smart contracts. Such functions can be used to encode/decode data (as defined in the formal specification), to perform mathematical functions (e.g. addition or multiplication of two integers), or to hash input data (using *Keccak-256* hash, *SHA-256* hash, or the *RIPEMD-160* hash). Finally, Solidity allows smart contracts to manage errors occurring during their execution. Currently, a smart contract can raise or handle two types of error: error conditions that can occur when invalid parameters are passed as input (`Error`) and critical conditions which are not expected during smart contract execution (`Panic`). The `require` and the `revert` function are used to signal `Error`, while the `assert` function is used to raise `Panic` conditions. An exception is raised when the `require`,

Table 4 Summary of the main properties of the Ethereum tokens

Token standard	Number of functions	Number of events	Fungibility
ERC20	6	2	Yes
ERC223	6	2	Yes
ERC1820	10	2	No
ERC777	13	5	Yes
ERC721	9	3	No

`revert`, and `assert` functions evaluate to true. Such an exception undoes all changes of the current call (and all its sub-calls), refunds the remaining gas to the caller, and reports the error to the caller. Smart contracts are able to catch error and panic conditions raised during function call by using `try/catch` statement. The `try` block consists of an expression and its return type variables, while the `catch` block supports different kinds of error. If the execution reaches a `catch` block, all changes will be reverted, and the execution continues in the corresponding catch block.

2.7 Tokens

Tokens are digital objects which can represent an entity or an object (e.g. resources, services, or rights). They are implemented on the Ethereum blockchain by using smart contracts that allow users to create, transfer, divide, or take ownership of such tokens. In order to ensure interoperability between tokens and regulate operations on them, the Ethereum community proposed different application-level standards, describing core specifications of token. Several standards have been proposed by the Ethereum by defining the signature of a set of functions (or interfaces) that the smart contract of the token must provide. Given a smart contract, the `slither-check-erc` tool[1] can be used to verify that it is compliant with the related ERC standards. Table 4 summarizes the properties of the most relevant proposals.

ERC20 It is one of the first standards provided by the Ethereum community. A token implementing the ERC20 standard has the property of fungibility because individual units are interchangeable and indistinguishable from the others. An ERC20-compliant smart contract must implement a set of six functions and two events. The functions implemented by the smart contract allow to access the amount of tokens owned by an account or to send tokens from one address to another address. Furthermore, this standard allows a generic address to transfer

[1] slither-check-erc: https://github.com/crytic/slither/wiki/ERC-Conformance.

some tokens on behalf of the token owners. The ERC20 standard defines two events issued as a result of the transfer operation.

ERC223 This standard aims to improving the transfer of tokens also to smart contract. In particular, the smart contract designed to work with tokens needs to implement a `tokenFallback` function. In addition, the `transfer` operation checks if the target is a CA or a EOA and, in the former case, the ERC223 token contract calls the `tokenFallback` function.

ERC777 and ERC1820 The standard ERC777 improves ERC20 by providing advanced functionalities to manage tokens. The ERC777 token must provide functions for getting the balance of an address, the token name, the token symbol, and the smallest fraction of the token value. In addition, the standard introduces also the concept of operators, i.e. addresses which are allowed to send, to produce new tokens (minting), and to burn tokens on behalf of some token holders (burning). Finally, events are emitted when an address becomes an operator or a token holder revokes an existing operator.

ERC721 The ERC721 standard is used to define unique and non-fungible tokens. Indeed, an ERC721-compliant smart contract associates to each token a unique identifier (`tokenId`) and provides functions for counting the number of tokens assigned to an owner, retrieving the owner of a specific token, transferring the ownership of a token, and managing the operators that can control the assets. The ERC721 standard defines three events: one issued as a result of the token transfer and two events that denote the approval of an operator.

2.8 The Ethereum Virtual Machine

The EVM is a distributed state machine which runs on top of each node of the Ethereum network, and it is responsible for executing smart contracts and maintaining the global state of the Ethereum blockchain. The EVM is kept synchronized among all nodes through a peer-to-peer network protocol, and the EVM implementation must be compliant with the specification described in the Ethereum Yellow Paper [32].

Given a set of valid transactions and the current system state (or *world state*), the main functionality implemented by the EVM is to determine the final state of the system by using a state transition function (STF). Figure 4 shows an overview of the EVM's global state. As a result of transactions evaluation, the EVM has to execute smart contracts in terms of EVM bytecode, which are generated from the EVM compiler. For this purpose, the EVM uses a stack-based architecture having word size of 256 bits. The global state of the EVM includes the world state and the machine state. In particular, the EVM relies on three different areas for storing data during execution: the storage, the memory, and the stack.

The storage is a persistent area that contains the world state, represented as a mapping between Ethereum addresses (160-bit identifiers) and *account states*. The

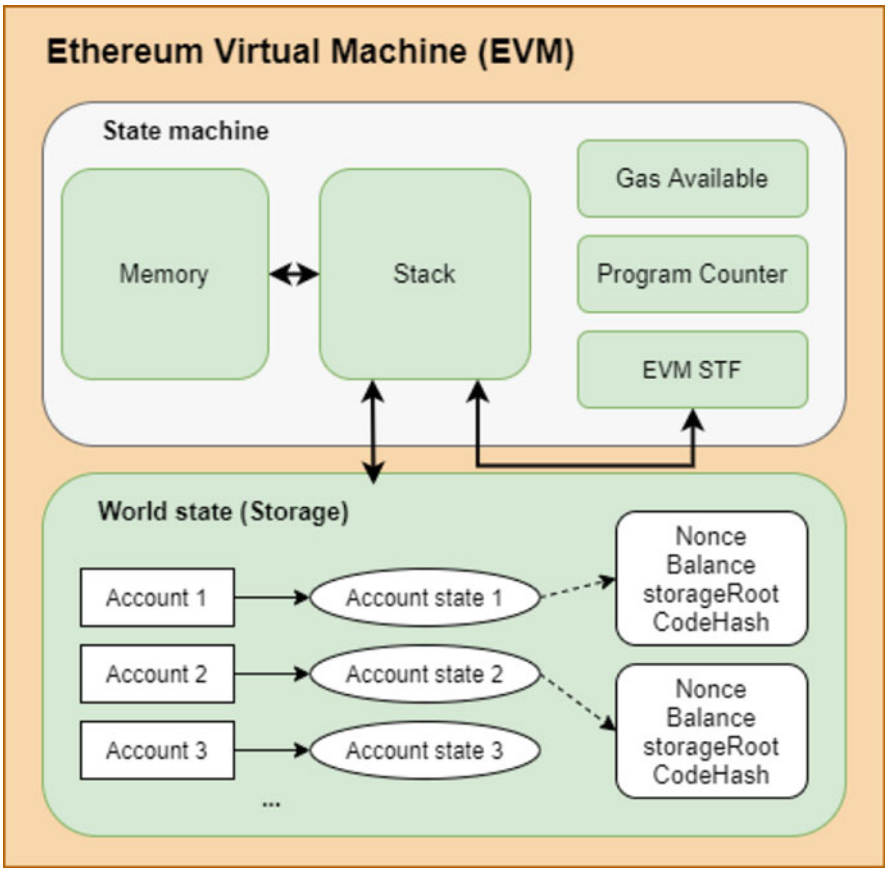


Fig. 4 Overview of the Ethereum Virtual Machine

implementation maintains this mapping in a modified Merkle Patricia tree (trie), a cryptographically authenticated data structure that can be used to store all (key, value) bindings. The root node of this structure cryptographically depends on its internal nodes. As a result, the Merkle root hash of the trie changes if any key-value pair is updated. Another important advantage of trie is its ability to provide a proof to verify that a certain key-value pair is included, without accessing the entire data structure. A trie is used to map the `sha3` of each ethereum address to an *account state*: a four-item array consisting of the nonce, balance, storage root, and the code hash. The code hash is immutable and defined only for CA, while for EOA the code hash field is the hash of an empty string. It is a hash pointer that refers to the code of the smart contract executed by the EVM. The storage root is the root of a separate storage trie which exists for each account. The storage trie is where all contract data are allocated as a mapping between 256-bit integer keys and 256-bit integer values.

Table 5 Comparison of the cost in gas for memory, storage, and stack

Stack		Memory		Storage	
Operations	Gas	Operations	Gas	Operations	Gas
PUSH	3	MLOAD	3	SLOAD	800
POP	2	MSTORE	3	SSTORE	5000–20,000

The EVM machine state consists of the available gas, the program counter, the memory contents, and the stack contents. The program counter increments on each instruction cycle and the gas limit of the transaction is reduced by the instruction's gas cost. Indeed, each EVM operation has an intrinsic relative cost in gas which is necessary for its computation, and the gas available for the proceeding computation must not exceed the transaction's gas limit. The stack has a maximum depth of 1024 items (each of 256 bits) and every function call uses at least one stack slot. Instead, the memory is a linear volatile area which consists of 2^{256} bytes initially equal to 0.

In addition to gas consumed by the EVM operations, gas may be paid due to an increase in the usage of the memory and storage. As shown in Table 5, the cost for memory is lower than the cost storage.[2] Indeed, the gas required to load or store a word on the memory is equal to 3. Instead, storage requires 800 units of gas in order to load a word, while the gas for storing data can range between 5000 and 20,000 units of gas, depending on different conditions. In particular, 20,000 units of gas is required when the storage value is set to nonzero from zero, 5000 units of gas when the storage value remains the same, and 15,000 units of gas refunded when the storage value is set to zero from nonzero. The amount of gas remaining after the computation is returned to the sender of the transaction.

3 Vulnerabilities and Known Attacks

In this section, we describe all the known vulnerabilities associated to the Ethereum platform and provide a discussion on the possible solutions. We discuss in Sect. 3.1 all the known vulnerabilities related to the Ethereum blockchain structure, while the vulnerabilities related to smart contracts are discussed in Sect. 3.2. Finally, vulnerabilities related to the consensus algorithm and the EVM are reviewed in Sects. 3.3 and 3.4, respectively.

[2] Ethervm: https://www.ethervm.io/.

3.1 Blockchain

In this section we review attacks and vulnerability issues of the blockchain structure and its network and data components. As previously mentioned, each node is identified in Ethereum by a 64-byte public key. An attacker may be able to generate an unlimited number of identities and hence nodes on the same machine, thus using them to isolate victim node from the network by monopolizing incoming and outgoing connections. This attack is known as *unlimited nodes creation* [21]. Another vulnerability that can be exploited to monopolize the connections of a victim client is given by *sole block synchronization*. Exploiting the fact that block synchronization is performed with a single node at the time due to network load considerations, a malicious node may delay the synchronization of a victim node, which will then be stalled, rejecting subsequent block [33]. Being a peer-to-peer network, Ethereum's nodes need to discover one another and this is implemented by means of research on a distributed hash table. However, since the mapping from node IDs to buckets in the table is public. Therefore, an attacker may be able to insert malicious node IDs in the bucket list of a victim node exploiting a *public peer selection* vulnerability [21].

3.2 Smart Contracts and Accounts

The lack of practical experience on developing smart contracts, as well as the characteristics of the blockchain, makes programming smart contracts in Ethereum particularly prone to errors. Indeed, developers have to be sure that smart contracts deployed on the Ethereum blockchain behave as expected because their code cannot be changed anymore. In addition, the Ethereum Yellow Paper [32], which formally defines the Ethereum protocol, has been subject to several reviews and enhancements because some concepts appear ambiguous, inaccurate, or misleading for the developers. For this reason, academic community and industrial community have investigated in more detail vulnerabilities, attacks, and defences related to smart contract development in Ethereum. Table 6 summarizes the vulnerabilities, attacks, and defences found in different related works.

Table 6 State of the art on the security of Ethereum smart contracts

Work	Vulnerabilities	Attacks	Defences
Atzei N., et al. [1]	11	8	3
Di Angelo M., et al. [9]	18	0	11
Liu J., et al. [17]	0	0	0
Praitheeshan P., et al. [25]	16	4	7
Chen H., et al. [5]	32	17	44
Wang Z., et al. [31]	–	–	–

Figure 5 summarizes the relevant vulnerabilities as well as an existing reference attack taken from the literature, if any. The vulnerabilities are grouped according to their causes, which could be related to either accounts or smart contracts. The sources of vulnerabilities in smart contracts are due to incorrect programming of smart contracts (Programming), ambiguous understanding of the Solidity language and toolchain (Language), or misunderstanding of the core design principles during smart contracts development (Design). Instead, the causes of vulnerabilities in accounts are mainly due to human factors (Human) or specific vulnerability of the underlying network (Networks). Instead, we summarize in Fig. 6 the best practices used to mitigate vulnerabilities and attacks on Ethereum smart contract.

Reentrancy This vulnerability is due to incorrect development of the smart contract. The vulnerability occurs when a smart contract C_1 calls back to a function on a malicious smart contract address C_2, received as input parameter, before a flag variable is updated. In particular, the smart contract C_2 can exploit the fallback function to call back again the function of the contract C_1.

DELEGATECALL Injection This vulnerability can be exploited on a smart contract C_1 using the DELEGATECALL operation provided by the EVM. As a result, it is due to incorrect development of the smart contract. Indeed, a malicious smart contract C_2 can use the DELEGATECALL to embed a bytecode into C_1, directly manipulating the state variables of the smart contract. This vulnerability, along with the erroneous visibility vulnerability, was exploited to implement an attack against the Parity multisignature wallet, which requires multiple private keys to unlock a wallet. The vulnerability can be prevented by implementing smart contract functionalities using library, whenever possible. Indeed, an attacker cannot modify directly the state of the library to affect the smart contracts that depend on the library's code.[3]

Frozen Ether The vulnerability exploits the fact that users deposit their money to contract accounts but they are unable to spend them. Indeed, a malicious attacker can intentionally kill or turn off the contracts providing the functions necessary for spending money. This vulnerability was observed after the attack against the Parity wallet,[4] where a user deleted the library-turned-into-wallet smart contract, making all the money in the multisignature wallet contracts unusable. The vulnerability can be prevented by avoiding to rely on other smart contracts to implement money-spending functions.

Upgrable Contract This vulnerability [1] takes advantage of the fact that smart contracts, once deployed, cannot be updated. Consequently, if a bug is found, the smart contract can be used by malicious users to carry out attacks. In order to mitigate the problem, users must carefully check the addresses with which they

[3] https://github.com/sigp/solidity-security-blog#delegatecall.

[4] https://etherscan.io/address/0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4.

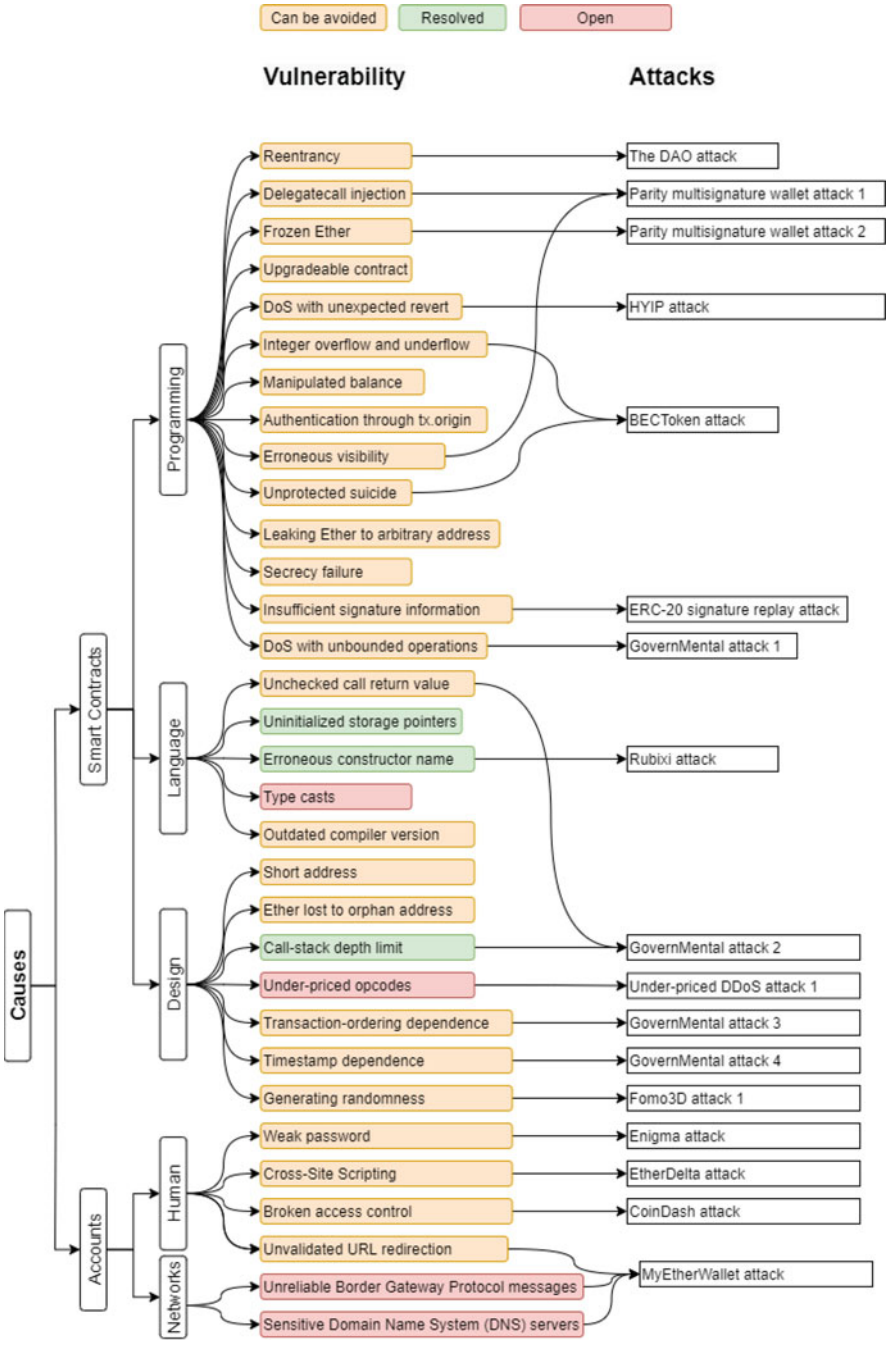


Fig. 5 Overview of the causes of vulnerabilities and attacks on Ethereum smart contract

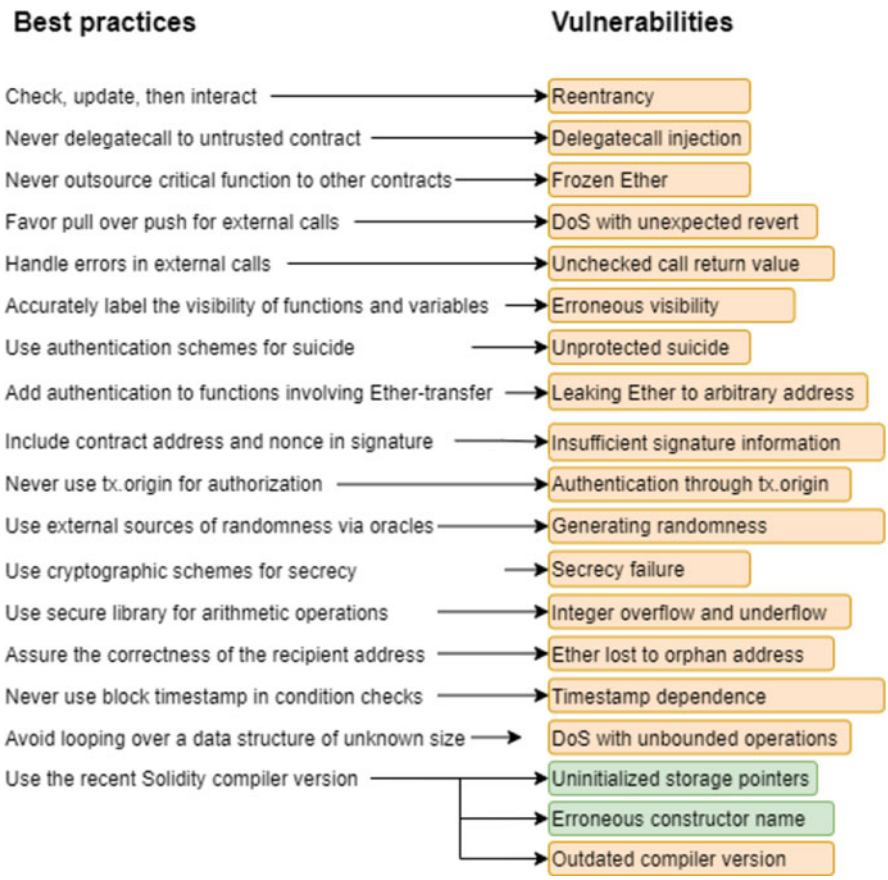


Fig. 6 Representation of the best practices used to mitigate vulnerabilities and attacks on Ethereum smart contract [5]

interact. Furthermore, to allow contract upgrading [5], developers can exploit proxy or registry design pattern that allows to correctly interface with the most updated version of a smart contract.

DoS with Unexpected Revert This vulnerability [6] occurs when a transaction is reverted due to a failure in executing an external call. The revert operation disrupts the execution and it can result in serious security issue. For example, if a smart contract iterates through an array to pay users, an attacker can intentionally fail, causing the reverting of all payments. This vulnerability can be prevented by avoiding to push funds automatically [6]. Instead, it is better to isolate each external call, allowing users to initiate transaction in order to receive funds.

Integer Overflow and Underflow This vulnerability occurs when the result of an arithmetic operation overflows or underflows the range of a Solidity data type.

Indeed, [5] validation on numeric inputs can result in unauthorized manipulation of the account balance or other state variables.

Manipulated Balance This vulnerability occurs when the balance state variable is used by a smart contract for control flow decision, and it can be prevented by not using a contract's balance in any condition check [5].

Authentication Through tx.origin This vulnerability occurs when the tx.origin, a global variable denoting the EOA that initiates a transaction, is used for authorization. For example, let's assume that a smart contract C_1 uses the above variable to validate the transfer of money to a malicious smart contract C_2. The smart contract C_2 can force the smart contract C_1 to automatically re-execute the same function by using the fallback. As a result, the tx.origin remains the same and the execution will pass the validity check used by C_1. In order to solve such vulnerability, developers can use the msg.sender variable, which refers to the immediate account that invokes the function.

Erroneous Visibility This vulnerability can occur when the function's visibility (viz. public, external, internal, and private) allows unauthorized access [5]. It was used with the DELEGATECALL injection vulnerability to attack the Parity multisignature wallet. Indeed, Solidity versions prior to 0.5.0 assume functions to be public by default, if not explicitly specified by developers. Current versions of Solidity have avoided such vulnerability by forcing developers to specify visibility of functions. This vulnerability can be prevented by correctly labelling the visibility of functions and state variables as necessary [6].

Unprotected Suicide This vulnerability can occur when a smart contract provides the suicide or self-destruct method, and it is used (together with the frozen ether vulnerability) against the Parity multisignature wallet. This vulnerability is caused by the inadequate authentication [5] checking. Indeed, this vulnerability can be prevented by enforcing access control. For example, the OpenZeppelin[5] library implements ownership in smart contracts. Each smart contract has an owner and it can specify which functions are intended only for administrative tasks.

Leaking Ether to Arbitrary Address This vulnerability occurs when a smart contract that holds assets can be transferred by a user. Consequently, a failure in checking the identity of the authorized users can result in leaking assets to arbitrary addresses. This vulnerability can be prevented by integrating a suitable access control mechanisms when programming smart contracts.[6]

Secrecy Failure This vulnerability [1] occurs when an attacker can benefit from private state variable of smart contracts. Indeed, due to the public nature of blockchain, private state variables remain visible and public on the blockchain, even if other smart contracts cannot access it directly. In order to prevent such

[5] https://docs.openzeppelin.com/contracts/4.x/access-control.

[6] https://docs.openzeppelin.com/contracts/4.x/access-control.

vulnerability, developers must protect the confidentiality of information stored in private or public state variables by using cryptographic techniques.

Insufficient Signature Information This vulnerability occurs when two similar smart contracts which act as proxies adopt the same way to verify the validity of the signature. As a result, if the signature does not include security-related information (e.g. nonce and proxy contract address), a malicious user can replay the signature multiple times to the other similar smart contracts, in order to withdraw extra payments [5]. This vulnerability was used against the UGT and MTC ERC-20 tokens,[7] and it can be prevented by including in the signature the information necessary to avoid a replay attack.[8]

DoS with Unbounded Operations This vulnerability occurs when the amount of gas that is required for executing a contract exceeds the block gas limit [5]. This vulnerability was discussed in [13] where a smart contract[9] iterates over data structure and deletes them, causing the freezing of 1100 ETH. As proposed in [5], this vulnerability can be mitigated by avoiding loops over data structures or splitting the loop through multiple transaction invocations.

Unchecked Call Return Value This vulnerability occurs when a smart contract interacts with another smart contract. Indeed, Solidity offers different methods to interact with the smart contract, such as call(), DELEGATECALL(), and send(), which never throw an exception, but will return false if the execution encounters an exception. Instead, default contract calls will automatically propagate a throw.

For example, an attack reported in [1] can exploit this vulnerability (along with the call-stack depth limit vulnerability) to make several calls before calling the target smart contract, which returns false because of the stack depth limit. If the target smart contract does not check the return value, it can lose money.

Smart contract can mitigate this vulnerability by checking the return value of a call and handling the possibility that the call will fail [6].

Uninitialized Storage Pointer This vulnerability occurs when a local variable of a function is used to store a reference of a state variable of the smart contract (e.g. struct, array, or mapping). Since the state variables are allocated in the storage area, the local variable must be explicitly initialized to point the specific area of the storage. If not initialized, the local variable points by default to slot 0 of the storage area. As a result, it can be eventually used to overwrite another area of the storage. Consequently, this vulnerability is due to the lack of proper initialization mechanisms in Solidity.

[7] https://github.com/nkbai/defcon26/blob/master/docs.

[8] https://programtheblockchain.com/posts/2018/02/17/signing-and-verifying-messages-in-ethereum/.

[9] http://etherscan.io/address/0xf45717552f12ef7cb65e95476f217ea008167ae3.

Starting from Solidity 0.5.0, this vulnerability was mitigated by forcing developers to explicitly indicate data location for all variables of struct, array, or mapping types [5].

Erroneous Constructor Name This vulnerability occurs in Solidity versions prior to 0.4.22, where the constructor of a smart contract is executed only on the creation and it is defined by a function having the same name of the smart contract. As a result, an error or inaccuracy in the name of the constructor can introduce issues in smart contract initialization because the compiler considers the constructor as a normal function, which can be invoked by any EOA. This vulnerability was successfully exploited against the DynamicPyramid smart contract, which was renamed by a developer as Rubixi without changing the original constructor's name.[10] In current version of Solidity, such vulnerability has been mitigated by introducing the new keyword for the constructor [5].

Type Casts A Solidity smart contract can declare variable that references to other smart contract C_1 of type T and can use such variables to directly call a function f on the smart contract instance C_1 [1]. However, Solidity only checks if the smart contract C_1 to which the variable refers declared a function f. As a result, Solidity cannot guarantee that the instance C_1 of the smart contract conforms to the type T. This vulnerability is caused by the type system of Solidity [5] and, currently, there are no solutions to avoid it.

Outdated Compiler Version This vulnerability occurs when a smart contract uses an outdated compiler, which contains reported bugs and thus makes the resulting bytecode vulnerable [5]. This vulnerability can be prevented by upgrading smart contracts to breaking language changes introduced by each Solidity version. For example, the Solidity-upgrade tool[11] can be used to find applicable source upgrades in smart contract.

Short Address This vulnerability is due to the encoding mechanisms used by Solidity smart contract to execute invocation of a function. Indeed, the invocation is encoded in a hexadecimal string consisting of the method identifier (4 bytes), the destination address (32 bytes), and the value to transfer (32 bytes). If the size of an input parameter is less than 32 bytes, the EVM will fill it with extra zeros. As a result, when a smart contract does not check the validity of the addresses, an attacker can provide inconsistent input parameters in order to affect the amount of funds to be sent (by adding extra zeros) [5].

In order to prevent this vulnerability, programmers have to check if parameters of the transaction (e.g. gas price and the destination address) match the expected values. Moreover, it is also possible to validate the Ethereum address checksum, if available (see EIP55).

[10] https://etherscan.io/address/0xe82719202e5965Cf5D9B6673B7503a3b92DE20be#contracts.

[11] https://docs.soliditylang.org/en/v0.8.4/using-the-compiler.htm.

Ether Lost to Orphan Address This vulnerability is due to the fact that Ethereum does not check if an address exists or not [1]. Consequently, if funds are transferred to a non-existing address, they cannot be recovered unless a user knows the private key associated with the address. In order to mitigate this vulnerability, programmers should verify the correctness of the addresses [5].

Call-Stack Depth Limit This vulnerability is related to the original Ethereum specification, where the call-stack depth cannot exceed 1024 frames. An attacker can exploit this vulnerability to reach the call-stack limit on a target smart contract. This vulnerability has been mitigated by EIP-150, a hard fork proposal that makes it impossible to reach 1024 in call-stack depth [5].

Underpriced Opcodes This vulnerability exploits the Ethereum instructions which consume less gas in order to waste computing resources of the system. Indeed, some instructions (e.g. SLOAD and EXT* opcodes) have historically been underpriced. This vulnerability was used to implement a transaction spam attack that repeatedly called the EXTCODESIZE instruction,[12] increasing the validation time to 20–60 seconds and reducing the block creation rate of the Ethereum network. With the EIP-2929, gas costs were further increased to mitigate malicious transaction spam attack. However, it is not clear whether the new amount of gas is enough to completely prevent this vulnerability.[13]

Transaction Ordering Dependence This vulnerability occurs when smart contract depends on the execution order of transactions. Indeed, the order of transactions in a block is defined by miners, who can endorse specific transactions in order to disadvantage specific users. For example, a malicious set of miners can exploit this vulnerability against an auction smart contract in order to include only their transactions in each auction [1]. As described in [5], this vulnerability can be mitigated by using both cryptographic mechanisms that hide transaction information.

Timestamp Dependence This vulnerability occurs when the execution of a smart contract depends on the timestamp information on the blockchain. Indeed, the timestamp of a block can be manipulated by miner, for at most 900 seconds from the timestamp of the parent block. This vulnerability can be used to attack smart contracts using the block timestamp as condition or as source of randomness [1]. Developer can mitigate this vulnerability avoiding the use of the block timestamp for checking conditions [5].

Generating Randomness This vulnerability occurs because of the lack of pseudorandom generator in Solidity. Indeed, a common practice in programming smart contracts is to generate a pseudorandom number based on state variables which are controlled by miners (such as timestamp, difficulty, block height, etc.). This

[12] https://blog.ethereum.org/2016/09/22/transaction-spam-attack-next-steps/.

[13] https://eips.ethereum.org/EIPS/eip-2929.

vulnerability can be prevented using external services (known as oracles [2]) to generate random numbers on the blockchain.

Weak Password This vulnerability was reported in [5], and it was used against the DApp called Enigma to compromise the password administrator and manipulate the website of the DApp. As a result, this vulnerability refers to the incorrect definition and usage of passwords, as well as their reuse and insecure storage. To mitigate this vulnerability, programmers and developers have to define passwords that resist to brute force or dictionary attack method and store them in a safe place.

Cross-Site Scripting This vulnerability exploits the browser of users to steal their private information by executing malicious JavaScript code. This vulnerability can be prevented by using the proper wallet for protecting the assets (hot wallet, cold wallet, etc.), avoiding to load private keys into the browser.

Broken Access Control This vulnerability exploits access control failures which occur at the server level and may result in unauthorized access and manipulation of a web application. In order to mitigate this vulnerability, developers should protect all components of their applications by defining suitable access control mechanisms.

Unvalidated URL Redirection This vulnerability occurs when a trusted application redirects user to untrusted endpoints which can maliciously access private information of the user. This vulnerability can be mitigated by users, checking the validity and trust of the URLs consumed by applications.

Unreliable Border Gateway Protocol Messages This vulnerability exploits the border gateway protocol messages of the underlying network infrastructure, which contain routing information that cannot be validated among the different autonomous systems. Consequently, an attacker can announce incorrect routing information to redirect traffic towards a specific peers.

Sensitive Domain Name System Services This vulnerability is related to the Domain Name System services provided by the underlying network infrastructure. Indeed, such service is responsible for resolving queries and redirecting users to the correct location. As a result, they have to provide correct and reliable information in order to protect user from potential threats. In addition, users have to check the validity of the information provided by Domain Name Systems using automatic browser tool.

3.2.1 Defences Tools

There are several defences that have been proposed for mitigating the smart contract vulnerabilities in Ethereum. Researchers found that the adoption of good development practices in programming smart contracts allows the mitigation of different vulnerabilities [5]. Figure 6 describes the recommended Solidity best practices which can be used to mitigate a large part of the identified vulnerabilities. Indeed, a set of 19 vulnerabilities can be easily identified by following 17 best practices.

In order to facilitate the vulnerabilities identification in complex smart contracts, researchers proposed several automatic smart contract analysis tools [9, 25] which can rely on the following techniques [5]: (i) abstract interpretation, which derives the semantic of a program in order to prove several properties; (ii) symbolic execution, which derives a control flow graph from a program and traverses all the feasible execution paths on the graph to identify vulnerability patterns; and (iii) model-based vulnerability detection, which represents the program by using a model (such as XML) and checks it against specific vulnerability patterns [5].

Figure 7 represents a subset of the popular available tools along with the vulnerabilities they detect:

Vandal Vandal [3] is a command-line tool written in Python, which disassembles and decompiles EVM bytecode into an intermediate representation. It can be used to detect reentrancy, manipulated balance, unprotected suicide, unchecked call return value, and authentication through tx.origin vulnerabilities.

Securify Securify [30] is a Java tool which decompiles the EVM bytecode and derives control and data flow in an abstract form. Finally, the tool searches for patterns which are compliant to different vulnerabilities.

Oyente Oyente [18] is a Python tool which can be used to detect different vulnerabilities of smart contracts. In particular, the tool retrieves the bytecode of a program and executes it symbolically, checking the execution in order to properly caught vulnerabilities.

Osiris Osiris [29] is a Python command-line tool extending Oyente capabilities. Indeed, it includes a module to detect also integer bugs, mitigating the integer overflow and underflow vulnerability.

Maian Maian [24] is a Python tool which exploits symbolic execution to trace smart contracts' behaviour. All possible execution is traced until a trace with a specific vulnerability is reached.

SmartCheck SmartCheck [28] is a Java symbolic execution tool which translates the program into an XML syntax tree and searches for specific XPath expression representing vulnerabilities.

Zeus Zeus [16] is a tool developed by IBM which takes Solidity code and a policy as inputs and checks whether the code meets the safety property described in the policy. The policy is used to detect a specific vulnerability and it has to be defined by the user through a custom language. The program is verified against the policy by using abstract interpretation.

Manticore Manticore [23] is a symbolic execution tool used from command line or via a Python API. It compiles Solidity code to bytecode and exploits solver to find computational paths corresponding to vulnerabilities.

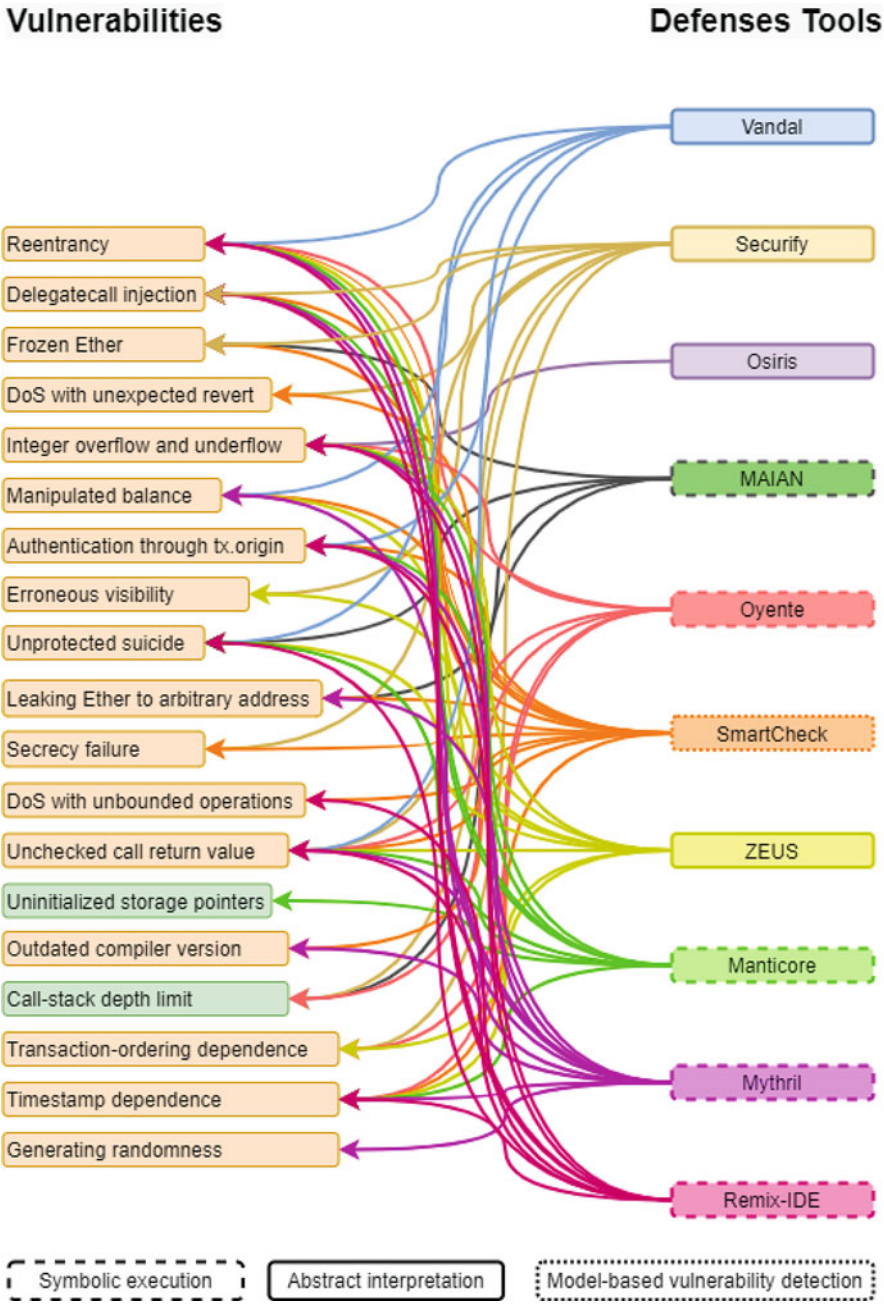


Fig. 7 Overview of the vulnerabilities identified by available tools on Ethereum smart contracts

Mythril Mythril[14] is a command-line tool developed in Python which uses symbolic execution on the EVM bytecode to generate a control flow graph. The analysis of the graph allows to search for potential vulnerabilities by exploring a finite set of possible states.

Remix IDE Remix IDE[15] is an open-source web and desktop application supporting the development of Solidity smart contracts. It provides a static analysis plugin to check for vulnerabilities and bad development practices.

3.3 Consensus Algorithm

In this section, we review known attacks to the consensus algorithm. In light of the recent development of Ethereum 2.0, we review security of both PoW and PoS consensus algorithms.

Attacks to the PoW Consensus Algorithm The 51% attack is common to all PoW implementations [7] and involves an attacker controlling the majority (51%) of the mining power. In this case, the attacker can revert transactions by creating blocks and performing double spending of assets [5]. In fact, by creating a fork and owning the majority of the computational power, the attacker is able to validate blocks in the fork at a rate higher than that of the main chain. The fork eventually becomes the main chain (due to the longest chain rule) and is controlled by the attacker. To increase the chances of a 51% attack, a malicious user can bribe honest miners, therefore enlarging its pool up to representing the majority of the network. If the fork eventually becomes the main chain, bribes will be paid by the blockchain in the form of block rewards [8]. This attack is inherent in the PoW consensus and no countermeasure can be taken. A second attack is the *outsourceable puzzle* [5] that specifically targets Ethash. In this attack a malicious user divides the search for a puzzle task into multiple subtasks and outsources the computation to other miners. In this case the attacker is able to speed up the computation, therefore gaining advantages over other miners. Alternative puzzles have been proposed [22] to cope with outsourcing attacks. Ethereum is an asynchronous network; hence no protocol can simultaneously guarantee agreement, termination, and validity. This is shown in *fixed consensus termination attack* [12], which exploits the fact that the confirmation of block persistency in the network is guaranteed with high probability after it is followed by m other blocks. In case of communication delays, applications need to wait for more than m blocks to be confirmed to gain high persistency probabilities and guarantee protection against double spending. This may cause the network to be unresponsive. The transactions to be confirmed are chosen by miners based on the gas price; therefore an attacker may be able to insert a huge number

[14] https://github.com/b-mueller/smashing-smart-contracts.

[15] https://remix-ide.readthedocs.io/.

of transactions in newly mined blocks, gaining exclusivity. Other transactions are hence abandoned by miners, causing a Denial of Service (DoS) attack [5]. The rewards for uncle blocks can be exploited by attackers. Selfish mining and double spending are incentivized by the fact that, with the uncle rewarding mechanism, a selfish miner can decide to include only its own uncle blocks or no blocks at all to enlarge the number of stale blocks and improve its revenue rate [26] .

Attacks to the PoS Consensus Algorithm PoS is vulnerable to some of the attacks described in PoW, e.g. bribery and 51% attacks [8]. In the consensus process, a user is able to stall the blockchain by skipping its validator turn remaining idle. This process, together with forking, is exploited in the *stake bleeding attack* [11]. In particular, the attacker first forks the main chain and starts to work uniquely on the forked version. The attacker remains idle each time it is entitled to validate a block, such that the main chain is slowed down. This also implies that the stake of the attacker gets incrementally decreased at each missed validator round. However, at the same time, the attacker adds transactions to the fork and is the only node that validates blocks on the fork. Eventually the attacker is able to outpace the main chain, due to a stake increase faster than that on the main chain. Finally, the attacker redistributes the stake to other validators and publishes the forged branch. A further attack which exploits the fork can be implemented against a PoS network that does not check timestamps [8]. In this case the fork is created starting from the genesis block, and blocks are minted at the same rate of the main chain. However, the attacker manipulates the timestamps, selecting values ahead of time. Since the attacker is the only validator in the fork, no other node will spot the crafted timestamps, and both forks will be valid. In the case in which timestamp forgery is not possible, the attacker needs to validate blocks at a rate higher than the main chain. Considering a retired node, i.e. a node who signed a certain number of transactions but is now retired and has nothing at stake but still holds a valid key, the attacker may be able to steal the retired node's private key or to collude with it. In both cases, the retired node is exploited to mint blocks on the fork, such that the attacker has increased chances of outpacing the main chain [8].

3.4 The Ethereum Virtual Machine

While most of the research has focused on smart contract security, the validation and the security problems of the EVM have not been well investigated. Indeed, implementing the EVM is a very complex task which involves several control structures and data structures. For this reason, vulnerabilities of the EVM are difficult to be detected and they might result in serious consequences for the entire

system. As a matter of fact, publicly disclosed cybersecurity vulnerabilities[16] related to the EVM are about 33, while those related to smart contracts are about 500.

Currently, there are 14 well-recognized EVM implementations which are developed under 9 different programming languages.[17] Since the EVM is continuously evolving, it is important that the implementation of the EVM is compliant with its specification. Officially, the VM tests are used to test the basic workings of the VM in isolation.[18] However, such test suites do not formally ensure that the specific EVM implementation is free from vulnerability because it is intended only as a basic test of VM operation. Authors of [10] investigate inconsistencies among different EVM implementations by using real-world smart contracts. Concerning the gas consumed by different implementations, they found that some EVMs (e.g. *js-evm*) use less gas than others (e.g. *Py-EVM*) because the Yellow Paper [32] did not specify gas consumption values for some instructions. Furthermore, they found that the sequences of instructions executed by the EVM implementations are not completely consistent, even if the final output is correct. In particular, the *js-evm* executes the shortest sequence of instructions, while *geth*, *aleth*, and *Py-EVM* result in high number of instructions. To formally prove security of such implementations, researchers developed semantic formalization of the EVM. For example, the author of [15] provides a formal specification of the EVM in Lem, where definition can be proved against the standard VM test suite. Instead, a different semantic formalization is used in [14], where the K Framework is used to define executable semantic specification of the EVM. Given the syntax and the semantic of the EVM, the K Framework generates formal method analysis tools for prototyping and testing updates to the EVM.[19]

Authors of [34] propose a formal framework for generating verified EVM implementations. They define a formal specification of the EVM in *Why3* and transform the verified specification of EVM into an executable *OCaml* module, which can be used in production environments.

The problem of detecting vulnerabilities of the EVM was also addressed in [10] through differential fuzz testing, providing invalid, unexpected, or random data as inputs to several programs in order to trigger EVM's bugs. Another interesting approach to solve security problems of EVM is to integrate controls that monitor the instructions executed by the EVM implementation. For example, authors of [19, 20] propose a methodology to reinforce EVM by integrating a monitoring strategy which can interrupt the EVM in the case of specific events occurring in real time. Finally, in the last years, several initiatives have been taken to strengthen the security and performance of the EVM. For example, Ethereum community currently proposed a redesign of the EVM using WebAssembly,[20] an open standard defining a

[16] https://cve.mitre.org/.

[17] EVM implementations: https://eth.wiki/concepts/evm/implementations.

[18] https://ethereum-tests.readthedocs.io/en/latest/test_types/vm_tests.html.

[19] https://jellopaper.org/.

[20] Ethereum WebAssembly: https://ewasm.readthedocs.io/.

portable code execution environment which allows to develop smart contracts using traditional programming languages (e.g. C++, Java, etc.).

4 Conclusions

In this chapter we presented the main components that characterize the Ethereum blockchain. We discussed how Ethereum represents a paradigm shift in the blockchain technology, thanks to its design targeting not only value transfer but also information sharing. Furthermore, thanks to the introduction of smart contracts, Ethereum allows for the automatic execution of operations between trusted members and for the inclusion of off-chain data in the blockchain. We also discussed the two consensus algorithms exploited in Ethereum, namely, proof of work and proof of stake. We also provided an in-depth discussion on the implementation of smart contracts and on the Ethereum Virtual Machine. Lastly, we discussed the known vulnerabilities related to all the Ethereum functions and components. In particular, we show how smart contracts are the most vulnerable part of Ethereum. Lastly we present some of the tools needed for mitigating the vulnerabilities identified for smart contracts. With this chapter we aim at providing the reader with a clear overview of the state-of-the-art Ethereum design and key aspects. Furthermore, we believe that this chapter provides a reference for smart contract designers and Ethereum blockchain deployers for a secure-by-design implementation.

References

1. N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on ethereum smart contracts (sok), in *International Conference on Principles of Security and Trust* (Springer, 2017), pp. 164–186
2. A. Beniiche, A study of blockchain oracles. Preprint (2020). arXiv:2004.07140
3. L. Brent, A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, B. Scholz, Vandal: A scalable security analysis framework for smart contracts. Preprint (2018). arXiv:1809.03981
4. V. Buterin et al., Ethereum white paper. GitHub Reposit. **1**, 22–23 (2013)
5. H. Chen, M. Pendleton, L. Njilla, S. Xu, A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. ACM Comput. Surv. (CSUR) **53**(3), 1–43 (2020)
6. ConsenSys, Ethereum smart contract best practices (2021). Available online: https://consensys.github.io/smart-contract-best-practices/
7. M. Conti, E. S. Kumar, C. Lal, S. Ruj, A survey on security and privacy issues of bitcoin. IEEE Commun. Surv. Tutor. **20**(4), 3416–3452 (2018)
8. E. Deirmentzoglou, G. Papakyriakopoulos, C. Patsakis, A survey on long-range attacks for proof of stake protocols. IEEE Access **7**, 28712–28725 (2019)
9. M. Di Angelo, G. Salzer, A survey of tools for analyzing ethereum smart contracts, in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)* (IEEE, 2019), pp. 69–78
10. Y. Fu, M. Ren, F. Ma, Y. Jiang, H. Shi, J. Sun, Evmfuzz: Differential fuzz testing of ethereum virtual machine. Preprint (2019). arXiv:1903.08483

11. P. Gaži, A. Kiayias, A. Russell, Stake-bleeding attacks on proof-of-stake blockchains, in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (IEEE, 2018), pp. 85–92
12. V. Gramoli, From blockchain consensus back to byzantine consensus. Future Gener. Comput. Syst. **107**, 760–769 (2020)
13. N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, Y. Smaragdakis, Madmax: Surviving out-of-gas conditions in ethereum smart contracts. Proc. ACM Programm. Lang. **2**(OOPSLA), 1–27 (2018)
14. E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu, et al., Kevm: A complete formal semantics of the ethereum virtual machine, in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)* (IEEE, 2018), pp. 204–217
15. Y. Hirai, Defining the ethereum virtual machine for interactive theorem provers, in *International Conference on Financial Cryptography and Data Security* (Springer, 2017), pp. 520–535
16. S. Kalra, S. Goel, M. Dhawan, S. Sharma, Zeus: Analyzing safety of smart contracts, in *Ndss*, pp. 1–12 (2018)
17. J. Liu, Z. Liu, A survey on security verification of blockchain smart contracts. IEEE Access **7**, 77894–77904 (2019)
18. L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269 (2016)
19. F. Ma, Y. Fu, M. Ren, M. Wang, Y. Jiang, K. Zhang, H. Li, X. Shi, Evm*: from offline detection to online reinforcement for ethereum virtual machine, in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (IEEE, 2019), pp. 554–558
20. F. Ma, M. Ren, Y. Fu, M. Wang, H. Li, H. Song, Y. Jiang, Security reinforcement for ethereum virtual machine. Inf. Process. Manag. **58**(4), 102565 (2021)
21. Y. Marcus, E. Heilman, S. Goldberg, Low-resource eclipse attacks on ethereum's peer-to-peer network. IACR Cryptol. ePrint Arch. **2018**, 236 (2018)
22. A. Miller, A. Kosba, J. Katz, E. Shi, Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 680–691 (2015)
23. M. Mossberg, F. Manzano, E. Hennenfent, A. Groce, G. Grieco, J. Feist, T. Brunson, A. Dinaburg, Manticore: A user-friendly symbolic execution framework for binaries and smart contracts, in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (IEEE, 2019), pp. 1186–1189
24. I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, A. Hobor, Finding the greedy, prodigal, and suicidal contracts at scale, in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 653–663 (2018)
25. P. Praitheeshan, L. Pan, J. Yu, J. Liu, R. Doss, Security analysis methods on ethereum smart contract vulnerabilities: a survey. Preprint (2019). arXiv:1908.08605
26. F. Ritz, A. Zugenmaier, The impact of uncle rewards on selfish mining in ethereum, in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (IEEE, 2018), pp. 50–57
27. D. Siegal, Understanding the dao attack (2016). Available online: https://www.coindesk.com/understanding-dao-hack-journalists
28. S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, Y. Alexandrov, Smartcheck: Static analysis of ethereum smart contracts, in *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, pp. 9–16 (2018)
29. C.F. Torres, J. Schütte, R. State, Osiris: Hunting for integer bugs in ethereum smart contracts, in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 664–676 (2018)

30. P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, M. Vechev, Securify: Practical security analysis of smart contracts, in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 67–82 (2018)
31. Z. Wang, H. Jin, W. Dai, K.-K.R. Choo, D. Zou, Ethereum smart contract security research: survey and future research opportunities. Front. Comput. Sci. **15**(2), 1–18 (2021)
32. G. Wood, Ethereum yellow paper: A formal specification of ethereum, a programmable blockchain (2018). https://github.com/ethereum/yellowpaper
33. K. Wüst, A. Gervais, Ethereum eclipse attacks. Technical report, ETH Zurich, 2016
34. X. Zhang, Y. Li, M. Sun, Towards a formally verified evm in production environment, in *International Conference on Coordination Languages and Models* (Springer, 2020), pp. 341–349

The Future Ring Confidential Transaction Protocols for Privacy-Preserving Blockchain

Muhammed F. Esgin, Joseph K. Liu, Shi-Feng Sun, and Dimaz Ankaa Wijaya

1 Introduction

The birth of Bitcoin in 2009 by Satoshi Nakamoto (pseudonym) [13] sparked a Cambrian explosion of a new asset class called cryptocurrency. At the time of writing, CoinMarketCap[1] records 3727 different cryptocurrencies with a total market value of US$393.9 billion.[2] Bitcoin has become the first open decentralized payment system that employs consensus mechanism on top of a peer-to-peer network called blockchain to replace the role of a central controller.

[1] https://coinmarketcap.com.

[2] As of 27 October 2020.

M. F. Esgin
Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Melbourne, VIC, Australia

CSIRO's Data61, Carlton, VIC, Australia
e-mail: muhammed.esgin@monash.edu

J. K. Liu (✉) · S.-F. Sun
Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Melbourne, VIC, Australia
e-mail: joseph.liu.sun@monash.edu; shifeng.sun@monash.edu

D. A. Wijaya
Deakin University, Geelong, VIC, Australia
e-mail: dimaz.wijaya@deakin.edu.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_5

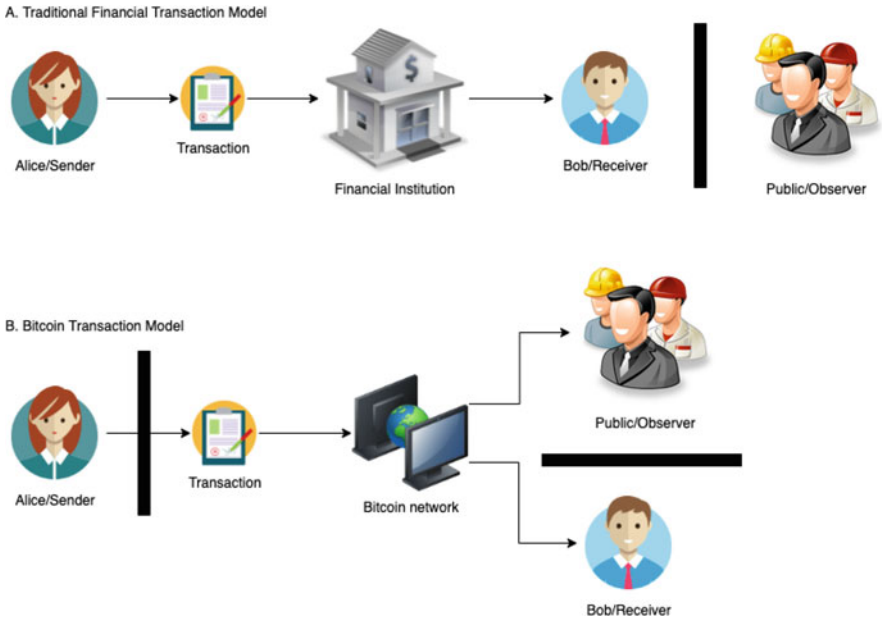


Fig. 1 Traditional financial system privacy model (**a**) compared to Bitcoin privacy model (**b**)

1.1 Bitcoin Privacy and Problems

For Bitcoin, Nakamoto devised a strategy that separates users' real identities with on-chain Bitcoin transactions by implementing public key cryptography. A hash value of the user's public key called an address represents the user's identity on the Bitcoin network. The address, similar to a bank account, can accumulate digital assets. The digital assets are transferable between users only if the owner provides a valid digital signature generated by using the associated private key.

Figure 1 describes the difference between a traditional financial system privacy model run by a centralized financial institution and the Bitcoin privacy model. Alice's (the sender) and Bob's (the receiver) identities are known to the financial institution. However, Alice's transaction that pays to Bob is not known to public (the observers). In Bitcoin, a payment transaction does not contain the sender and the receiver's identities except for their addresses. The transaction is recorded in public network where observers are able to verify the correctness.

However, existing research shows that Bitcoin users are identifiable through analyses that utilize transaction patterns and amount, address reuse, geolocation, and KYC/AML procedures by cryptocurrency exchanges. The findings signify Nakamoto's flawed privacy model that deems Bitcoin as pseudo-anonymous.

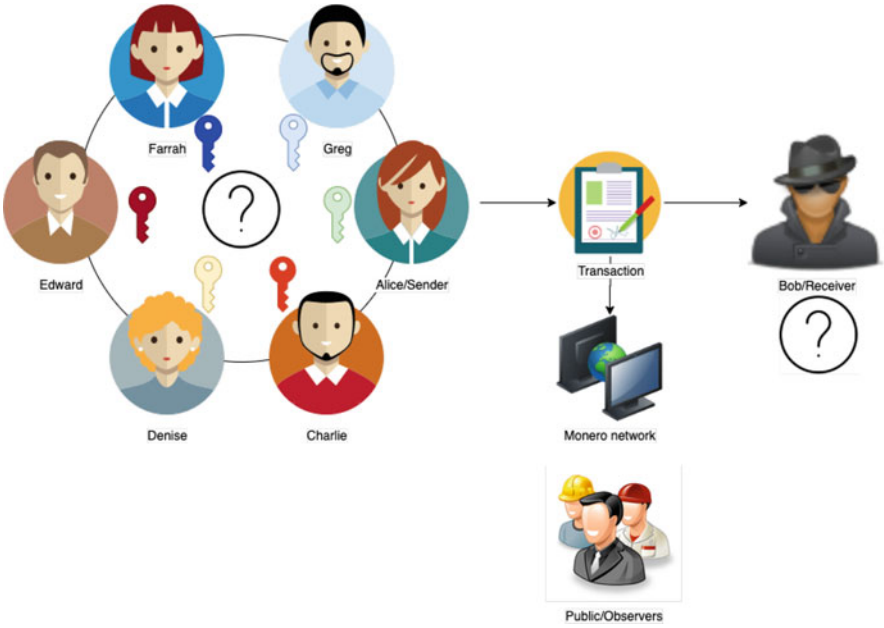


Fig. 2 Monero transaction model

1.2 Monero

Monero is one of the top privacy-preserving cryptocurrencies in the world in terms of economic value. Its market cap is currently at US$2.3 billion.[3] van Saberhagen proposed the basic protocol called CryptoNote in 2013 [21] that became the basis of Monero protocol. CryptoNote utilizes a cryptographic technique called linkable ring signature (LRS) [10] to preserve the sender's privacy and one-time public key (OTPK) to protect the receiver's privacy.

Figure 2 illustrates a Monero transaction where Alice (the sender) pays Bob (the receiver). Alice generates an LRS that takes some of the existing users' public keys from the blockchain at random. Alice that knows Bob's master key generates a new public key for Bob as the destination address for the transaction. Alice then sends the transaction to the network. The observers can see Alice's transaction in Monero blockchain; however:

1. The observers cannot distinguish who the real sender is, among a set of users: Alice, Charlie, Denise, Edward, Farrah, and Greg
2. The observers cannot identify who the real receiver is, because there is no past record of the destination public key

[3] As of 27 October 2020.

Similar to other cryptocurrencies, there are applications required to run the Monero system, namely, Monero daemon and Monero wallet. A Monero daemon is the full node keeping a full record of all transactions happening in the network, while a Monero wallet does not need to store all transactions locally. Monero wallet helps to create new transactions by connecting to a Monero daemon.

When creating a new transaction, the Monero wallet first sends a request to the Monero daemon. The purpose of the requests is to get information about potential public keys (outputs) to be used as decoys in the ring signature construction. The wallet selects a set of global indexes from the histogram provided by the daemon by using a random sampling algorithm. Then the daemon provides the corresponding outputs. The index of the real output is also in the request for two reasons: as a test whether the daemon sends the correct outputs and to obfuscate the final ring signature from a curious daemon.

Monero developers have adopted additional protection on users' privacy. The Confidential Transaction (CT) is added into ring signature construction to create the RingCT [15]. The Confidential Transaction is a Pedersen commitment to encrypting the amount of money sent from the sender to the receiver so that they cannot be visible to the world; only the respective participants can decrypt the amount [12]. RingCT was deployed in January 2017 and became mandatory since September 2017.[4]

RingCT has brought a significant change on the way users create mixins. In a non-RingCT transaction, an output can only mix with other outputs with the same amount. Before RingCT is available, an output having an unprecedented amount of coins cannot mix with other outputs. Hence zero-mixin transaction occurs. The zero-mixin transaction is a transaction containing an input that does not have any mixin, as shown in Fig. 3. A user who owns the key k_1 containing a unique amount does not find any other keys to construct a ring signature with more than one member; in this case, the user creates a zero-mixin transaction. An observer identifies that transaction TX_1 spends key k_1. Later, another user owning the key k_2 with the same amount as k_1 mixes both keys into a new ring signature R_2. Intuitively, the probability of the observer to guess which key is spent by the transaction TX_2 is 50%. However, since k_1 is already spent by transaction TX_1, therefore the observer is confident that k_2 is spent by transaction TX_2, an event called chain reaction [15].

RingCT encrypts the amount of money contained in each key. Therefore, a RingCT output can mix with any outputs because all RingCT output's coin amount is indistinguishable. RingCT prevents zero-mixin transaction and chain reaction and, hence, improves Monero privacy.

[4] https://getmonero.org/resources/moneropedia/ringCT.html.

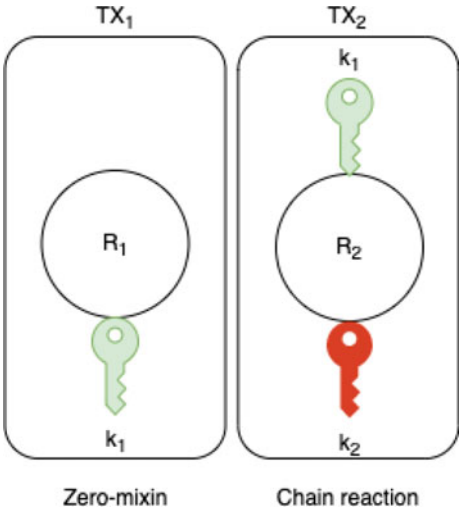


Fig. 3 Problems with monero (**a**) zero mixin transaction and (**b**) chain reaction

2 RingCT 3.0: A Scalable and Efficient RingCT Protocol

Ring Confidential Transaction (RingCT) is proposed to achieve confidentiality of transaction amounts and anonymity of senders, which is supposed to be the backbone of the most popular anonymous cryptocurrency Monero.

In this section, we present the most recent work on RingCT. We first briefly recall the security model for RingCT and then present our protocol RingCT 3.0 [22]. For the ease of presentation, we first give a basic construction of our RingCT from linkable ring signature. Then we optimize our construction to $\log(n)$-size by using the inner product argument in [2], where n is the size of the ring.

2.1 Security Model for RingCT

We give the security definitions and models for RingCT which is modified from [18]. A RingCT protocol consists of a tuple of polynomial time algorithms (**Setup**, **KeyGen**, **Mint**, **AccountGen**, **Spend**, **Verify**), the syntax of which is described as follows:

- $pp \leftarrow$ **Setup**(1^λ): It takes a security parameter $\lambda \in \mathbb{N}$ and outputs the system parameters pp. All algorithms below have implicitly pp as part of their inputs.
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow$ **KeyGen**$()$: In order to provide anonymity to the recipient, the concept of *stealth address* was used in RingCT1.0 [14]. It can be viewed as dividing the algorithm into two parts: generating long-term key pairs for each user and generating one-time key pairs for each transaction.

- **LongTermKeyGen**. It outputs a long-term secret key ltsk and a long-term public key ltpk.
- **OneTimePKGen**. On input a long-term public key ltpk, it outputs pk and the auxiliary information R_{ot}.
- **OneTimeSKGen**. On input a one-time public key pk, an auxiliary information R_{ot}, and a long-term secret key ltsk, it outputs the one-time secret key sk.

- $(cn, ck) \leftarrow$ **Mint**$(\mathsf{pk}, \mathsf{a})$: It takes as input a public key pk and an amount a and outputs a coin cn for pk as well as the associated coin key ck.
- $(act, a\mathsf{sk})/\perp \leftarrow$ **AccountGen**$(\mathsf{sk}, \mathsf{pk}, cn, ck, \mathsf{a})$: It takes as input a user key pair $(\mathsf{sk}, \mathsf{pk})$, a coin cn, a coin key ck, and an amount a. It returns $\perp$ if ck is not the coin key of cn with amount a. Otherwise, it outputs the account $act \doteq (\mathsf{pk}, cn)$ and the corresponding account secret key is $a\mathsf{sk} \doteq (\mathsf{sk}, ck, \mathsf{a})$.
- $(\mathbb{A}_{\mathsf{out}}, \pi, \mathbb{S}, \mathbb{C}k_{\mathsf{out}})/\perp \leftarrow$ **Spend**$(\mathtt{m}, \mathbb{K}_S, \mathbb{A}_S, \mathbb{A}_{\mathsf{in}}, \mathbb{O})$: It takes as input a group $\mathbb{A}_S$ of accounts together with the corresponding account secret keys $\mathbb{K}_S$, an arbitrary set $\mathbb{A}_{\mathsf{in}}$ of groups of input accounts containing $\mathbb{A}_S$, a set $\mathbb{O}$ of output public keys with the corresponding output amounts, and some transaction string $\mathtt{m} \in \{0, 1\}^*$; it outputs $\perp$ if the sum of output amount in $\mathbb{O}$ is different from the sum of input amount in $\mathbb{K}_S$. Otherwise, it outputs a set of output accounts $\mathbb{A}_{\mathsf{out}}$, a proof π, a set $\mathbb{S}$ of serial numbers, and a set of output coin keys $\mathbb{C}k_{\mathsf{out}}$. Each serial number $S_i \in \mathbb{S}$ corresponds to one account secret key $a\mathsf{sk}_i \in \mathbb{K}_S$.
- 1/0/-1 $\leftarrow$ **Verify**$(\mathtt{m}, \mathbb{A}_{\mathsf{in}}, \mathbb{A}_{\mathsf{out}}, \pi, \mathbb{S})$: It takes as input a message $\mathtt{m}$, a set of input accounts $\mathbb{A}_{\mathsf{in}}$, a set of output accounts $\mathbb{A}_{\mathsf{out}}$, a proof π, and a set of serial numbers $\mathbb{S}$; the algorithm outputs -1 if the serial numbers in $\mathbb{S}$ are spent previously. Otherwise, it checks if the proof π is valid for the transaction tx and outputs 1 or 0, meaning a `valid` or `invalid` spending, respectively.

The formal security model is given in the full version of the paper [22]. We give a high-level description as follows:

Perfect Correctness The perfect correctness property requires that a user spend any group of her accounts w.r.t. an arbitrary set of groups of input accounts, each group containing the same number of accounts as the group she intends to spend.

Anonymity The anonymity of RingCT is more complicated than the anonymity of linkable ring signatures, due to the extra knowledge of transaction amount. The previous model of anonymity in RingCT1.0 considered outsider security only (i.e., not against the recipient and other members of the ring). In the following, we define two stronger models for anonymity: anonymity against recipients (who know all the output amounts) and anonymity against ring insiders (who know some input account secret keys and their amounts).

Anonymity Against Recipients The anonymity against recipients' property requires that, without the knowledge of any input account secret key and input amount (which are within a valid *Range*: from 0 to a maximum value), the spender's

accounts are successfully hidden among all the honestly generated accounts, even when the output accounts and the output amounts are known.

Anonymity Against Ring Insiders The anonymity against ring insiders' property requires that, without the knowledge of output account secret key and output amount (which are within a valid *Range*), the spender's accounts are successfully hidden among all uncorrupted accounts.

Balance The balance property requires that any malicious user cannot (1) spend any account of an honest user, (2) spend her own accounts with the sum of input amount being different from that of output amount, and (3) double spend any of her accounts. Therefore, the balance property can be modeled by three security models: unforgeability, equivalence, and linkability.

Non-slanderability The non-slanderability property requires that a malicious user cannot prevent any honest user from spending. It is infeasible for any malicious user to produce a valid spending that shares at least one serial number with an honestly generated spending.

2.2 Our Basic Construction

In this part, we first give our basic construction with an O(n)-size proof. Then we optimize it to $\log(n)$-size by using the inner product argument in next subsection.

Our scheme uses a zero-knowledge range proof of a value committed in a Pederson commitment. Denote $\mathsf{RP} = (\mathsf{RSetup}, \mathsf{RProof}, \mathsf{RVerify})$ as a zero-knowledge range proof for the statement:

$$PoK : \{(\mathsf{a}, \kappa) : C = h_c^{\mathsf{a}} g_c^{\kappa} \wedge \mathsf{a} \in [R_{\mathsf{min}}, R_{\mathsf{max}}]\}.$$

The range proof can be instantiated by the Bulletproofs [2]. In detail, our basic construction is as follows:

Setup On input security parameter λ and the maximum size of ring n_{max}, it picks a group $\mathbb{G}$ of prime order p and some generators $g_c, h_c, g, u \in \mathbb{G}$, $\mathbf{g} = (g_1, \ldots, g_{n_{\mathsf{max}}})$, $\mathbf{h} = (h_1, \ldots, h_{n_{\mathsf{max}}}) \in \mathbb{G}^{n_{\mathsf{max}}}$. Suppose that $H_j : \{0,1\}^* \to \mathbb{Z}_p$ for $j = 1, 2, 4, 5$, $H_3 : \{0,1\}^* \to \mathbb{G}$ and $H_6 : \mathbb{G} \to \mathbb{Z}_p$ are collision-resistant hash functions. It also runs RSetup of the range proof. Assume these parameters are known in the system.

KeyGen The algorithm is divided as follows:

- **LongTermKeyGen**. The user picks his long-term secret key $\mathsf{ltsk} \doteq (x_1, x_2) \in \mathbb{Z}_p^2$ and computes his long-term public key $\mathsf{ltpk} \doteq (g^{x_1}, g^{x_2})$.
- **OneTimePKGen**. On input a long-term public key $\mathsf{ltpk} = (g^{x_1}, g^{x_2})$, it picks a random $r_{\mathsf{ot}} \in \mathbb{Z}_p$ and computes a one-time public key $\mathsf{pk} = g^{x_1} \cdot g^{H_6((g^{x_2})^{r_{\mathsf{ot}}})}$. It outputs pk and the auxiliary information $R_{\mathsf{ot}} \doteq g^{r_{\mathsf{ot}}}$.

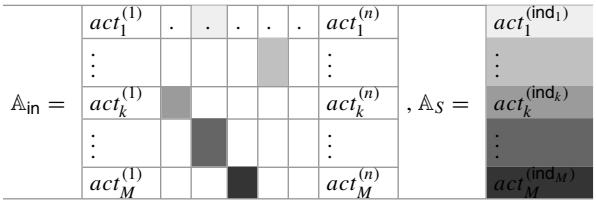
- **OneTimeSKGen**. On input a one-time public key pk, an auxiliary information R_{ot}, and a long-term secret key $\mathsf{ltsk} = (x_1, x_2)$, it checks if $\mathsf{pk} = g^{x_1} \cdot g^{H_6(R_{\mathsf{ot}}^{x_2})}$. If it is correct, then it outputs the one-time secret key $\mathsf{sk} = x_1 + H_6(R_{\mathsf{ot}}^{x_2})$.

Mint On input a public key pk and an amount $\mathsf{a} \in \mathbb{Z}_p$, the algorithm chooses $\kappa \in \mathbb{Z}_p$ uniformly at random and computes the coin $C = g_c^{\kappa} h_c^{\mathsf{a}}$. It returns the coin C and the coin key $\mathsf{cnk} = \kappa$.

AccountGen On input a user key pair $(\mathsf{sk}, \mathsf{pk})$, a coin C and a coin key $\mathsf{cnk} = \kappa$ (where the pair (pk, C) is listed as the output of a transaction), and an amount a, it checks if $C = g_c^{\kappa} h_c^{\mathsf{a}}$. If it is true, then it outputs the account $act \doteq (\mathsf{pk}, C)$ and the corresponding account secret key is $a\mathsf{sk} \doteq (\mathsf{sk}, \mathsf{cnk}, \mathsf{a})$.

Spend On input a set of M signer's input accounts $\mathbb{A}_S$ with a set of account secret keys $\{a\mathsf{sk}_k = (\mathsf{sk}_k, \kappa_{\mathsf{in},k}, \mathsf{a}_{\mathsf{in},k})\}_{k \in [1,M]}$, a set of nM input accounts $\mathbb{A}_{\mathsf{in}}$ which contains $\mathbb{A}_S$ (where $n < n_{\mathsf{max}}$ is the size of the ring), a set of N output amounts $\{\mathsf{a}_{\mathsf{out},j}\}_{j \in [1,N]}$ corresponding to N recipient's public keys $\{\mathsf{pk}_{\mathsf{out},j}\}_{j \in [1,N]}$, and a transaction message $\mathtt{m}$, it first checks the amount balance. If $\sum_{k=1}^{M} \mathsf{a}_{\mathsf{in},k} \neq \sum_{j=1}^{N} \mathsf{a}_{\mathsf{out},j}$, the transaction amount is not correct and it returns $\perp$.

Arrange $\mathbb{A}_{\mathsf{in}}$ as an $M \times n$ matrix with each row containing only one account in $\mathbb{A}_S$. Denote the column index ind_k as the position of the k-th element in $\mathbb{A}_S$ appearing in row k, column ind_k of $\mathbb{A}_{\mathsf{in}}$. The graphical representation is as follows:



The spend protocol can be roughly separated into two parts. The first part is mainly about the balance of the input and output amount. The second part is mainly about the ring signature providing anonymity of the sender.

We first give some sub-protocols for the *balance* property, as shown in the shaded part.

1. Generate One-Time Public Key: The sender converts all recipient's long-term public keys to one-time public keys by **OneTimePKGen**. The auxiliary information is appended to the transaction message $\mathtt{m}$.

(continued)

2. Generate Output Coins. It first runs $(C_{\mathsf{out},j}, \kappa_{\mathsf{out},j}) \leftarrow \mathbf{Mint}(\mathsf{a}_{\mathsf{out},j})$, for all $j \in [1, N]$. It sets $\mathbb{A}_{\mathsf{out}} = \{(\mathsf{pk}_{\mathsf{out},j}, C_{\mathsf{out},j})\}_{j\in[1,N]}$ as the set of N output accounts.
   The sender can later privately send the amount $\mathsf{a}_{\mathsf{out},j}$ and coin key $\kappa_{\mathsf{out},j}$ to each secret key owner of $\mathsf{pk}_{\mathsf{out},j}$. Denote $\mathbb{C}k_{\mathsf{out}}$ as the set of all coin keys.
3. Generate Range Proof. It runs the RProof of the range proof for all $\mathsf{a}_{\mathsf{out},j}$ where $j \in [1, N]$. Denote π_{range} as the set of outputs of RProof for all j.
4. Prepare Balance Proof. Denote the coin for $act_k^{(\mathsf{ind}_k)}$ as $C_{\mathsf{in},k}^{(\mathsf{ind}_k)}$. Recall that the coin key of $C_{\mathsf{in},k}^{(\mathsf{ind}_k)}$ is $(\mathsf{a}_{\mathsf{in},k}, \kappa_{\mathsf{in},k})$. If the sum of input amount is equal to the sum of output amount, we have $\prod_{k=1}^{M} C_{\mathsf{in},k}^{(\mathsf{ind}_k)} / \prod_{j=1}^{N} C_{\mathsf{out},j} = g_c^{\sum_{k=1}^{M} \kappa_{\mathsf{in},k} - \sum_{j=1}^{N} \kappa_{\mathsf{out},j}}$. Denote $\Delta \doteq \sum_{k=1}^{M} \kappa_{\mathsf{in},k} - \sum_{j=1}^{N} \kappa_{\mathsf{out},j}$.

Next, we give some sub-protocols for the *ring signature* part shaded below. Denote $act_k^{(i)} = (\mathsf{pk}_{\mathsf{in},k}^{(i)}, C_{\mathsf{in},k}^{(i)})$ for $i \in [1, n]$ and the signer index is ind_k. The sender runs as follows:

1. Generate One-Time Secret Key: The sender converts his long-term secret key to one-time secret keys by **OneTimeSKGen**.
2. Generate Key Images. Denote $(\mathsf{sk}_k, \cdot, \cdot)$ as the account secret key for $act_k^{(\mathsf{ind}_k)}$. It computes the key image $U_k = u^{\frac{1}{\mathsf{sk}_k}}$.
3. Ring Formation. Denote the concatenated string str as the concatenation of $\{act_k^{(1)}||\ldots||act_k^{(n)}\}_{k\in[1,M]}$. The prover computes $d_0 = H_2(0, \mathsf{str})$, $d_1 = H_2(1, \mathsf{str})$, and $d_2 = H_2(2, \mathsf{str})$. The prover sets $\mathbf{Y} = \mathbf{Y_1}||\ldots||\mathbf{Y_M}$, where:

   $$\mathbf{Y_k} = ((\mathsf{pk}_{\mathsf{in},k}^{(1)})^{d_0^{k-1}} (C_{\mathsf{in},k}^{(1)})^{d_1} g_1^{d_2}, \ldots, (\mathsf{pk}_{\mathsf{in},k}^{(n)})^{d_0^{k-1}} (C_{\mathsf{in},k}^{(n)})^{d_1} g_n^{d_2}) \quad \text{for } k \in [1, M].$$

4. Prepare Signer Index. For $k \in [1, M]$, the sender generates a binary vector $\mathbf{b_{L,k}} = (b_{k,1}, \ldots, b_{k,n})$, where $b_{k,i} = 1$ when $i = \mathsf{ind}_k$ and $b_{k,i} = 0$ otherwise. Define $\mathbf{b_L} = \mathbf{b_{L,1}}||\ldots||\mathbf{b_{L,M}}$ and $\mathbf{b_R} = \mathbf{b_L} - \mathbf{1^n}$. We will prove in zero-knowledge that $\mathbf{b_{L,k}}$ is a binary vector with only one bit equal to 1. It is equivalent to showing: $\mathbf{b_L} \circ \mathbf{b_R} = \mathbf{0}^n$, $\mathbf{b_L} - \mathbf{b_R} = \mathbf{1}^n$, $\langle \mathbf{b_{L,k}}, \mathbf{1}^n \rangle = 1$ for $k \in [1, M]$.

(continued)

5. Signature Generation. It consists of the following steps:
– *Commit 1.* It sets $h = H_3(\mathbf{Y})$, picks random $\alpha_1, \alpha_2, \beta, \rho, r_{\alpha_1}, r_{\alpha_2}, r_{\mathsf{sk}_1}, \ldots, r_{\mathsf{sk}_M}, r_\delta \in \mathbb{Z}_p$, $\mathbf{s_L}, \mathbf{s_R} \in \mathbb{Z}_p^{nM}$, and computes:

$$B_1 = h^{\alpha_1} \prod_{k=1}^{M} (\mathsf{pk}_{\mathsf{in},k}^{(\mathsf{ind}_k)})^{d_0^{k-1}} (C_{\mathsf{in},k}^{(\mathsf{ind}_k)})^{d_1} g_{\mathsf{ind}_k}^{d_2}, \quad B_2 = h^{\alpha_2} \prod_{k=1}^{M} g_{\mathsf{ind}_k}, \quad A = h^{\beta} \mathbf{h}^{\mathbf{b_R}},$$

$$S_1 = h^{r_{\alpha_1} - d_2 r_{\alpha_2}} g^{\sum_{k=1}^{M} r_{\mathsf{sk}_k} d_0^{k-1}} g_c^{d_1 r_\Delta}, \quad S_2 = h^{\rho} \mathbf{Y}^{\mathbf{s_L}} \mathbf{h}^{\mathbf{s_R}}, \quad S_3 = \prod_{k=1}^{M} U_k^{r_{\mathsf{sk}_k} d_0^{k-1}}.$$

Observe that $B_1 = h^{\alpha_1} \mathbf{Y}^{\mathbf{b_L}}$.
– *Challenge 1.* Denote the concatenated string $\mathsf{str}' = \mathbf{Y}||B_1||B_2||A||S_1||S_2||\ S_3||U_1||\ldots||U_M$. It computes $y = H_4(1, \mathsf{str})$, $z = H_4(2, \mathsf{str})$ and $w = H_4(3, \mathsf{str})$.
– *Commit 2.* It can construct two degree 1 polynomials of variable X:

$$l(X) = \mathbf{b_L} - z\mathbf{1}^{nM} + \mathbf{s_L} \cdot X,$$

$$r(X) = \mathbf{y}^{nM} \circ (w\mathbf{b_R} + wz\mathbf{1}^{nM} + \mathbf{s_R} \cdot X) + \sum_{k=1}^{M} z^{1+k} \cdot (\mathbf{0}^{(k-1)n}||\mathbf{1}^n||\mathbf{0}^{(M-k)n}).$$

Denote $t(X) = \langle l(X), r(X) \rangle$, which is a degree 2 polynomial. We can write $t(X) = t_0 + t_1 X + t_2 X^2$, and t_0, t_1, t_2 can be computed by using $(\mathbf{b_L}, \mathbf{b_R}, \mathbf{s_L}, \mathbf{s_R}, w, y, z)$. In particular, observe that:

$$t_0 = w\langle \mathbf{b_L}, \mathbf{b_R} \circ \mathbf{y}^{nM} \rangle + zw\langle \mathbf{b_L} - \mathbf{b_R}, \mathbf{y}^{nM} \rangle + \sum_{k=1}^{M} z^{1+k} \langle \mathbf{b_L}, \mathbf{0}^{(k-1)n}||\mathbf{1}^n||$$

$$\mathbf{0}^{(M-k)n} \rangle - wz^2 \langle \mathbf{1}^{nM}, \mathbf{y}^{nM} \rangle - \sum_{k=1}^{M} z^{2+k} \langle \mathbf{1}^{nM}, \mathbf{0}^{(k-1)n}||\mathbf{1}^n||\mathbf{0}^{(M-k)n} \rangle,$$

$$= \sum_{k=1}^{M} z^{1+k} + w(z - z^2)\langle \mathbf{1}^{nM}, \mathbf{y}^{nM} \rangle - \sum_{k=1}^{M} nz^{2+k}.$$

It picks random $\tau_1, \tau_2 \in \mathbb{Z}_p$ and computes: $T_1 = g^{t_1} h^{\tau_1}$, $T_2 = g^{t_2} h^{\tau_2}$.
– *Challenge 2.* It computes $x = H_5(w, y, z, T_1, T_2, \mathbb{m})$.
– *Response.* It computes:

(continued)

$$\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2, \quad \mu = \alpha_1 + \beta \cdot w + \rho \cdot x, \quad z_{\alpha_1} = r_{\alpha_1} + \alpha_1 \cdot x,$$

$$z_{\alpha_2} = r_{\alpha_2} + \alpha_2 \cdot x, \quad z_\Delta = r_\Delta + \Delta \cdot x, \quad z_{\mathsf{sk},k} = r_{\mathsf{sk},k} + \mathsf{sk}_k \cdot x \ \text{ for } k \in [1, M],$$

$$\mathbf{r} = \mathbf{y}^{nM} \circ (w\mathbf{b_R} + wz\mathbf{1}^{nM} + \mathbf{s_R} \cdot x) + \sum_{k=1}^{M} z^{1+k} \cdot (\mathbf{0}^{(k-1)n} || \mathbf{1}^n || \mathbf{0}^{(M-k)n}),$$

$$\mathbf{l} = \mathbf{b_L} - z \cdot \mathbf{1}^{nM} + \mathbf{s_L} \cdot x, \quad t = \langle \mathbf{l}, \mathbf{r} \rangle.$$

It outputs $\sigma_{\mathsf{ring}} = (B_1, B_2, A, S_1, S_2, S_3, T_1, T_2, \tau_x, \mu, z_{\alpha_1}, z_{\alpha_2}, z_{\mathsf{sk},1}, \ldots, z_{\mathsf{sk},M}, z_\Delta, \mathbf{l}, \mathbf{r}, t)$ and the key image $(U_1, \ldots, U_M)$.

Output. Denote $\mathbb{S}$ as a set of serial numbers $\{U_1, \ldots, U_M\}$. Then the output of the Spend algorithm is $(\mathbb{A}_{\mathsf{out}}, \pi = (\pi_{\mathsf{range}}, \sigma_{\mathsf{ring}}), \mathbb{S}, \mathbb{C}k_{\mathsf{out}})$.

Verify On input a message $\mathtt{m}$, a set of input accounts $\mathbb{A}_{\mathsf{in}}$, a set of output accounts $\mathbb{A}_{\mathsf{out}}$, a proof π, and a set $\mathbb{S}$ of serial numbers and a set $\mathbb{U}$ of serial numbers in the past, then it checks:

1. If there exists any U in both $\mathbb{S}$ and $\mathbb{U}$, it returns -1 and exits since it is a double spending of the previous transaction. We can use Bloom filter on $\mathbb{U}$ to speed up the detection of double spending.
2. It runs the RVerify algorithm of the range proof with input from π_{range} and the output coins in $\mathbb{A}_{\mathsf{out}}$.
3. It checks the ring signature σ_{ring} and key images $U_k \in \mathbb{S}$ for $k \in [1, M]$. It computes d_0, d_1, d_2, and $\mathbf{Y}$ as in the Ring Formation of the **Spend** protocol, using $\mathbb{A}_{\mathsf{in}}$. Denote the concatenated string $\mathsf{str} = \mathbf{Y}||B_1||B_2||A||S_1||S_2||S_3||U_1||\ldots||U_M$. It computes $h = H_3(\mathbf{Y})$, $y = H_4(1, \mathsf{str})$, $z = H_4(2, \mathsf{str})$, $w = H_4(3, \mathsf{str})$, and $x = H_5(w, y, z, T_1, T_2, \mathtt{m})$. Define $\mathbf{h'} = (h'_1, \ldots, h'_{nM}) \in \mathbb{G}^{nM}$ such that $h'_i = h_i^{y^{-i+1}}$ for $i \in [1, nM]$. It returns 1 if all of the following hold and returns 0 otherwise:

$$t = \langle \mathbf{l}, \mathbf{r} \rangle, \tag{1}$$

$$g^t h^{\tau_x} = g^{\sum_{k=1}^{M} z^{1+k}(1-nz) + w(z-z^2)\langle \mathbf{1}^{nM}, \mathbf{y}^{nM} \rangle} \cdot T_1^x \cdot T_2^{x^2}, \tag{2}$$

$$h^\mu \mathbf{Y}^{\mathbf{l}} \mathbf{h'}^{\mathbf{r}} = B_1 A^w S_2^x \mathbf{Y}^{-z \cdot \mathbf{1}^{nM}} \mathbf{h'}^{wz \cdot \mathbf{y}^{nM} + \sum_{k=1}^{M} z^{1+k} \cdot (\mathbf{0}^{(k-1)n} || \mathbf{1}^n || \mathbf{0}^{(M-k)n})}, \tag{3}$$

$$h^{z_{\alpha_1} - d_2 z_{\alpha_2}} g^{\sum_{k=1}^{M} z_{\mathsf{sk},k} d_0^{k-1}} g_c^{d_1 z_\Delta} = S_1 (B_1 \cdot \prod_{j=1}^{N} C_{\mathsf{out},j}^{d_1} \cdot B_2^{-d_2})^x, \tag{4}$$

$$\prod_{k=1}^{N} U_k^{z_{\mathsf{sk},k} d_0^{k-1}} = S_3 \cdot u^{x \sum_{k=1}^{N} d_0^{k-1}}. \tag{5}$$

2.3 Our Efficient Construction

The last step toward our final construction is to use the improved inner product argument in [2] to compress the $O(n)$-size vector $\mathbf{l}, \mathbf{r}$ in the ring signature part to a $O(\log n)$-size proof. Denote **IPProve**, **IPVerify** as the inner product argument. Details of the algorithm can be found in [2]. We give the modified **Spend**$'$ and **Verify**$'$ algorithms as follows:

- **Spend**$'$. On input $(\mathtt{m}, \mathbb{K}_S, \mathbb{A}_S, \mathbb{A}_{\mathsf{in}}, \mathbb{O})$, it runs $(\mathbb{A}_{\mathsf{out}}, \pi = (\pi_{\mathsf{range}}, \sigma_{\mathsf{ring}}), \mathbb{S}, \mathbb{C}k_{\mathsf{out}}) \leftarrow$ **Spend**$(\mathtt{m}, \mathbb{K}_S, \mathbb{A}_S, \mathbb{A}_{\mathsf{in}}, \mathbb{O}))$. For each $\sigma_{\mathsf{ring}} = (B_1, B_2, A, S_1, S_2, S_3, T_1, T_2, \tau_x, \mu, z_{\alpha_1}, z_{\alpha_2}, z_{\mathsf{sk},1}, \ldots, z_{\mathsf{sk},M}, z_\Delta, \mathbf{l}, \mathbf{r}, t)$, it computes $P = \mathbf{Y}^{\mathbf{l}}\mathbf{h}'^{\mathbf{r}}$, where $\mathbf{Y}$ and $\mathbf{h}'$ are defined in **Spend**. It runs $(\mathbf{L}, \mathbf{R}, a, b) \leftarrow$ **IPProve**$(\mathbf{Y}, \mathbf{h}', t, P, \mathbf{l}, \mathbf{r})$. Note that $\mathbf{L}, \mathbf{R}$ are vectors of $\mathbb{G}$ with size $\log n$. It sets $\sigma'_{\mathsf{ring}} = (B_1, B_2, A, S_1, S_2, S_3, T_1, T_2, \tau_x, \mu, z_{\alpha_1}, z_{\alpha_2}, z_{\mathsf{sk},1}, \ldots, z_{\mathsf{sk},M}, z_\Delta, t, P, \mathbf{L}, \mathbf{R}, a, b)$. The algorithm outputs $(\mathbb{A}_{\mathsf{out}}, \pi = (\pi_{\mathsf{range}}, \sigma'_{\mathsf{ring}}), \mathbb{S}, \mathbb{C}k_{\mathsf{out}})$.
- **Verify**$'$. On input $(\mathtt{m}, \mathbb{A}_{\mathsf{in}}, \mathbb{A}_{\mathsf{out}}, \pi = (\pi_{\mathsf{range}}, \sigma'_{\mathsf{ring}}), \mathbb{S})$, denote $\sigma'_{\mathsf{ring}} = (B_1, B_2, A, S_1, S_2, S_3, T_1, T_2, \tau_x, \mu, z_{\alpha_1}, z_{\alpha_2}, z_{\mathsf{sk},1}, \ldots, z_{\mathsf{sk},M}, z_\Delta, t, P, \mathbf{L}, \mathbf{R}, a, b)$. It runs $0/1 \leftarrow$ **IPVerify**$(\mathbf{Y}, \mathbf{h}', t, P, (\mathbf{L}, \mathbf{R}, a, b))$, where $\mathbf{Y}$ and $\mathbf{h}'$ are defined in **Verify**. It outputs 0 if **IPVerify** outputs 0. Otherwise, it runs as the **Verify** algorithm, except that Equation (3) is modified to:

$$h^{\mu} P = B_1 \cdot A^{w} \cdot S_2^{x} \cdot \mathbf{Y}^{-z \cdot \mathbf{1}^{nM}} \cdot \mathbf{h'}^{wz \cdot \mathbf{y}^{nM} + \sum_{k=1}^{M} z^{1+k} \cdot (\mathbf{0}^{(k-1)n} || \mathbf{1}^{n} || \mathbf{0}^{(M-k)n})}.$$

The security of our final construction follows from the security of the improved inner product argument in [2], which is based on the DL assumption in the random oracle model.

3 MatRiCT: The First Practical Post-quantum RingCT Protocol

Due to the significant advances in quantum computing, researchers and blockchain community are also in search of blockchain protocols that may withstand powerful attacks by scalable quantum computers. As shown in a seminal work by Shor [17], the currently deployed blockchain protocols based on discrete logarithm or integer factorization problem are vulnerable to quantum attacks and therefore do not provide security in the *post-quantum* setting, where scalable quantum computers exist. As a result, there is a strong need to design post-quantum (i.e., quantum-safe) alternatives of various cryptographic algorithms with blockchain protocols being no exception. There are in fact existing efforts to integrate post-quantum constructions into the blockchain setting. For example, Ethereum 2.0 Serenity upgrade [3] is planned to include an option for a post-quantum signature scheme.

With this goal in mind, researchers have worked on post-quantum alternatives of the RingCT protocol. In particular, as a promising post-quantum cryptography candidate, RingCT proposals relying on lattice-based computational security assumptions have been introduced [9, 19, 20]. The first attempt to design a post-quantum RingCT protocol was done by Torres et al. in [20] (called LRCT v1.0, hereafter). However, this construction only works in the single-input single-output setting where the users are only able to spend a single input account to a single output address. With this restriction, it becomes easy to satisfy the balance property. Then, in another work of Torres et al. [19] (called LRCT v2.0, hereafter), LRCT v1.0 was extended to support multiple-input multiple-output wallets. However, the concrete practical efficiency of LRCT v2.0 falls far behind practical expectations. For example, it requires proofs in the order of many MBs and user public keys of size more than 100 KB.

After these two initial works [19, 20], a substantial progress in the construction of post-quantum RingCT protocols was made in the work of Esgin et al. [9]. The authors introduced a *practical*, *scalable*, and *post-quantum* RingCT protocol, named MatRiCT [9]. Unlike the prior post-quantum constructions, MatRiCT can scale to large anonymity set sizes (i.e., ring sizes for the ring signature), requires much smaller parameters, and leads to much shorter proofs and user public keys. Let us investigate in a bit in more detail how MatRiCT works.

The security of MatRiCT is proven based on the two well-known computational lattice problems, Module Short Integer Solution (M-SIS) and Module Learning With Errors (M-LWE). These problems are generalizations of the plain SIS [1] and LWE [16] problems, which have been studied for quite a while. As a result, the MatRiCT construction is built upon strong foundations.

As outlined in earlier sections, there are three core ingredients of a RingCT protocol: (i) a homomorphic commitment scheme, (ii) a (linkable) ring signature, and (iii) a range proof. For the commitment scheme, MatRiCT uses the so-called Hashed-Message Commitment (HMC) based on M-SIS and M-LWE. This commitment scheme is widely deployed in various lattice-based proof systems (see, e.g., [6, 7]). The more crucial parts are how to realize the ring signature and the range proof *efficiently* in the lattice setting.

As *exact* lattice-based zero-knowledge proofs are very costly in communication (see, e.g., [5]), MatRiCT is completely built on *relaxed* lattice-based zero-knowledge proofs, which are also the basis for efficient lattice signatures. To give an example, a relaxed opening proof of a commitment C proves knowledge of (α, s) such that $\alpha \cdot C = \mathsf{COM}(s)$. So, the opening relation is relaxed with the involvement of an additional (secret) "relaxation factor" α. In the case of *exact* proofs, we simply have $\alpha = 1$ (or, equivalently, $C = \mathsf{COM}(s)$ with no α involved).

For the ring signature, MatRiCT makes use of an improved version of the one-out-of-many proof in [6], which itself builds on the proposal in [7]. The former proof [6] is *one-shot* in the sense that a single protocol execution is sufficient to reach a negligible soundness error. In a one-out-of-many proof, the prover convinces the verifier that she knows the secret key associated to an undisclosed public key in a

set of public keys. Such a proof naturally leads to a ring signature by making the proof noninteractive, e.g., using the Fiat-Shamir transformation.

Due to the nature of the range proofs requiring operations on large numbers (such as 2^{64}), existing lattice-based range proofs are costly in terms of communication [6]. As a result, MatRiCT employs a novel balance proof strategy that does not require range proofs on a wide range (such as 32-/64-bit ranges).

More specifically, suppose that we want to prove the following relation:

$$\sum_{i=0}^{M-1} a_{\text{in},i} = \sum_{i=0}^{S-1} a_{\text{out},i}, \tag{6}$$

for some input/output amounts $a_{\text{in},0}, \ldots, a_{\text{in},M-1}, a_{\text{out},0}, \ldots, a_{\text{out},S-1}$ with $M, S \in \mathbb{Z}^+$. Confidential validation of (6) is to commit to each amount value using a homomorphic commitment scheme and then to prove that (i) each committed message is an integer in a valid positive range and (ii) the sum of the output commitments minus the sum of the input commitments leads to a commitment to zero. As the operations in the lattice setting are performed modulo an integer q (called the system modulus), a crucial detail to take into account is that (6) must hold over $\mathbb{Z}$, not just $\mathbb{Z}_q$. This necessitates a choice of q larger than the amounts themselves (e.g., $q > 2^{64}$ if the amounts have 64-bit precision). As the efficiency of lattice-based constructions degrade with increasing q, this requirement leads to large parameters and proof lengths in the lattice setting.

With this introduction, let us see how MatRiCT performs the balance proof. Assume that we work in base β for some $\beta \geq 2$ and each amount is represented by r digits. Then, for any amount a, we can write $a = \sum_{j=0}^{r-1} \beta^j a[j]$ with its digits $a[j]$'s. This leads to the following equivalences:

$$\begin{aligned}
\sum_{i=0}^{M-1} a_{\text{in},i} = \sum_{i=0}^{S-1} a_{\text{out},i} &\iff \sum_{i=0}^{M-1}\sum_{j=0}^{r-1} \beta^j a_{\text{in},i}[j] = \sum_{i=0}^{S-1}\sum_{j=0}^{r-1} \beta^j a_{\text{out},i}[j], \\
&\iff \sum_{j=0}^{r-1} \beta^j \sum_{i=0}^{M-1} a_{\text{in},i}[j] = \sum_{j=0}^{r-1} \beta^j \sum_{i=0}^{S-1} a_{\text{out},i}[j], \\
&\iff 0 = \sum_{j=0}^{r-1} \beta^j \left(\sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] \right), \qquad (7) \\
&\iff 0 = \sum_{j=0}^{r-1} \beta^j \left(\sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] + c_j - \beta c_{j+1} \right), \qquad (8)
\end{aligned}$$

for $c_0 = c_r = 0$ and any "corrector values" $c_1, \ldots, c_{r-1} \in \mathbb{Z}$. Therefore, instead of the general approach that requires a very large system modulus, MatRiCT's balance proof proceeds as follows: For simplicity, let us look at the case of $\beta = 2$. The

prover commits to the bits of each amount as:

$$C_{\text{in},i} = \text{Com}(a_{\text{in},i}[0], \ldots, a_{\text{in},i}[r-1];\ *),$$
$$C_{\text{out},i} = \text{Com}(a_{\text{out},i}[0], \ldots, a_{\text{out},i}[r-1];\ *).$$

For ease of representation, we just write "*" for the randomness parts. Then, the prover also creates a "corrector" commitment $C = \text{Com}(c_0 - 2c_1, \ldots, c_{r-1} - 2c_r;\ *)$ with $c_0 = c_r = 0$. One can observe the pattern in the message that is similar to the relation when performing addition with carries. Finally, the prover proves that (i) $C_{\text{in},i}$s and $C_{\text{out},i}$s are commitments to binary values, (ii) the resulting commitment of $\sum_{i=0}^{S-1} C_{\text{out},i} - \sum_{i=0}^{M-1} C_{\text{in},i} + C$ is a commitment to zero, and (iii) the corrector commitment C is well-formed as above. These proofs in turn guarantee that (i) the message opening of any coin commitments represents a unique value in the range $[0, 2^r - 1]$, (ii)

$$0 = \sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] + c_j - 2c_{j+1} \tag{9}$$

for any $j \in \{0, \ldots, r-1\}$, and (iii) the corrector commitment C represents zero in this representation (i.e., does not add any value). Therefore, the prover convinces the verifier of (8) and equivalently (6). Importantly, since the message openings inside the commitments are very small integers (particularly, just bits for $\beta = 2$), there is no need to use a very large modulus. In particular, the system modulus q just needs to be large enough to make sure that (9) holds over $\mathbb{Z}$, which is already easily satisfied in typical practical lattice proof systems.

Note that (7) cannot be directly used to prove balance as, when the amounts are represented by commitments to their bit vectors, the addition of these commitments adds the corresponding bit vectors over $\mathbb{Z}_q$ where $q \gg 2$. If we denote the bit vector of a positive integer a by $\mathsf{Bits}(a)$, we have:

$$\text{Com}(\mathsf{Bits}(a_1);\ *) + \text{Com}(\mathsf{Bits}(a_2);\ *) \neq \text{Com}(\mathsf{Bits}(a_1 + a_2);\ *),$$

and therefore the proof does not work without the corrector commitment C.

3.1 Overview of MatRiCT

Thanks to the above novel balance proof, an important feature of MatRiCT is that no *wide* range proof is needed. In particular, MatRiCT uses two system moduli q (of about 31 bits) and $\hat{q}$ (of about 53 bits), both of which are much smaller than 2^{64} even though the amounts are represented by 64 bits. The operations in

MatRiCT are performed over two cyclotomic rings $R_q = \mathbb{Z}_q[X]/(X^d + 1)$ and $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d + 1)$ for a power-of-2 d. More specifically, $d = 64$ is chosen.

A user secret key sk is formed by sampling a random short vector over R_q, and a user public key is computed as a commitment to zero with sk serving as the randomness. For a confidential representation of an amount, a user commits to the bits of the amount over R_q when minting a coin. This differs from the standard technique of simply committing to the integer amount. Therefore, the new balance proof is needed to realize the balance property of the construction.

In MatRiCT, each account is comprised of a pair of a public key and a coin. To spend some of her accounts, a user Alice first mints the output coins and calculates the "corrector" values for the balance proof. These corrector values help Alice realize the balance proof as described earlier. For simplicity, let us focus on the case when there is a single input account and at most two output accounts. In this case, the corrector values are just bits. To hide her own identity, Alice collects other accounts to use in a ring signature. Assume that Alice wants to spend M accounts while hiding her identity among N users. Then, she gathers $M \cdot N$ accounts in total (including those of her own) and orders them as an $M \times N$ matrix. In particular, she picks an index $\ell \in [0, N-1]$ and puts her own accounts on the ℓ-th column.

After this initial setup, Alice executes an *aggregated* binary proof over $R_{\hat{q}}$. In this proof, she proves that (i) all output coins created are commitment to binary messages, (ii) the index ℓ identifying her is properly encoded by some elements of $R_{\hat{q}}$, and (iii) some ring elements in $R_{\hat{q}}$ properly encode the "corrector" values. Alice then runs M ring signatures for the accounts she wants to spend along with the balance proof described above. More specifically, she computes an additional ring signature on commitments $P_0, \ldots, P_{N-1}$ where

$$P_i = \sum(\text{output coins}) - \sum\begin{pmatrix}\text{input coins in}\\ i\text{-th column}\end{pmatrix} + (\text{“corrector” commitment}).$$

Observe that all of the secret values that construct P_ℓ are known by Alice, and hence she can prove knowledge of its opening. The ring signature on $P_0, \ldots, P_{N-1}$ in fact shows that P_ℓ is a commitment to zero. From here, it follows from the novel balance proof idea that the sum of input (integer) amounts equals the sum of output (integer) amounts. Finally, to prove that she is not trying to double spend, Alice calculates serial numbers as commitments to her secret keys used under a new commitment key and proves in zero-knowledge that this is indeed the case.

A bonus feature supported by MatRiCT is *auditability*, where an authority in possession of some (secret) trapdoor can revoke the anonymity of a user, e.g., to prevent illegal use of a cryptocurrency. Each user has the freedom to choose a specific auditor from a set of possible auditors or can even opt to be not audited at all. On the other hand, choosing a specific auditor can also be enforced by making sure that non-audited transactions do not pass the verification.

3.2 Algorithmic Description of MatRiCT

For a message vector $\mathbf{m}$ and a randomness vector $\mathbf{r}$, we write:

$$\mathrm{Com}_{ck}(\mathbf{m};\ \mathbf{r}) = \mathbf{G} \cdot (\mathbf{r}, \mathbf{m}) = \mathbf{A} \cdot \mathbf{r} + \mathbf{B} \cdot \mathbf{m}$$

to denote a commitment to the message $\mathbf{m}$ with randomness $\mathbf{r}$ under the commitment key $ck = \mathbf{G}$. MatRiCT makes use of three commitment keys $\mathbf{G}$, $\hat{\mathbf{G}}$, and $\mathbf{H}$ in general where $\mathbf{H}$ (defined over R_q) is used only when creating the serial numbers, $\hat{\mathbf{G}}$ is used in the binary proof for the commitments over $R_{\hat{q}}$, and $\mathbf{G}$ is used in the remaining steps, particularly, for the commitments over R_q. Next, we describe the procedures used in MatRiCT. The algorithms provided here are (almost) identical to those in [9]. We let r be the integer where any sum of the amounts can be represented by r bits. As in [9], we fix the notations in Table 1. The algorithmic descriptions concerning the "corrector" values are for the case of $(M, S) = (1, 2)$.

The discussions in this section are kept brief. For a more detailed coverage of relevant background information, the MatRiCT protocol and beyond, the reader is referred to [4].

Algorithm 1 expands a small seed ρ' to create a random matrix using an extendable output function Sam (modeled as a random oracle). The function also takes an auxiliary input string str to allow generation of different matrices from the

Table 1 Identifiers for MatRiCT

$\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$	An account comprised of a public key and a coin
$M, S \geq 1$	# of spender's input and output accounts, resp
$N \geq 2$	# of accounts to hide a single input account
A_{in}	Set of all input accounts arranged as a $M \times N$ matrix where the i-th row contains $\mathsf{R}_{\mathrm{in}}[i]$
tx	A transaction $\mathsf{tx} = (\mathsf{A}_{\mathrm{in}}, \mathsf{PK}_{\mathrm{out}}, \mathsf{CN}_{\mathrm{out}}, \Pi, \mathsf{SN})$
V	Set of all valid amounts
Π	The proof output
SN	Set of serial numbers
$\mathsf{CK}_{\mathrm{out}}$	Set of output coin keys with $\lvert\mathsf{CK}_{\mathrm{out}}\rvert = S$
$\mathsf{PK}_{\mathrm{out}}$	Set of output public keys with $\lvert\mathsf{PK}_{\mathrm{out}}\rvert = S$
$\mathsf{CN}_{\mathrm{out}}$	Set of output coins with $\lvert\mathsf{PK}_{\mathrm{out}}\rvert = S$
$R_q, R_{\hat{q}}$	Cyclotomic rings of degree d : $R_q = \mathbb{Z}_q[X]/(X^d + 1)$, $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d + 1)$
$n, \hat{n}$	Height of commitment matrices in R_q and $R_{\hat{q}}$, resp
$m, \hat{m}$	Randomness vector dimensions in R_q and $R_{\hat{q}}$, resp
$N = \beta^k$	Ring size of ring signature (and anonymity set size)
ℓ	Spender's column index with $0 \leq \ell < N$
r	Bit-length of an amount, i.e., $\mathsf{amt} \in \mathsf{V} = [0, 2^r - 1]$
$\mathcal{B}$	Max absolute coefficient of initial randomness

Algorithm 1 $\mathsf{SamMat}(\rho', v, q', n', m', \text{str})$

INPUT: ρ' for some seed $\rho' \in \{0,1\}^{256}$; $v, q', n', m' \in \mathbb{Z}^+$; str is an optional auxiliary input string.

1: $\mathbf{G} \leftarrow \mathsf{Sam}(\rho', \text{str})$ where $\mathbf{G} \in R_{q'}^{n' \times (m'+v)}$
2: **return** $\mathbf{G}$

Algorithm 2 $\mathbf{Setup}(1^\lambda)$ — λ is the security parameter

1: Choose int. params $k, \beta, r, n, m, \hat{n}, \hat{m}, n_s, d, q, \hat{q}$ such that $N = \beta^k$
2: Set w, p such that $|C^d_{w,p}| > 2^{256}$
3: $\rho \leftarrow \{0,1\}^{256}$
4: Pick a hash function $\mathcal{H} : \{0,1\}^* \rightarrow C^d_{w,p}$
5: **return** $pp = (\rho, \mathcal{H}, k, \beta, r, n, m, \hat{n}, \hat{m}, n_s, d, w, p, q, \hat{q})$

Algorithm 3 $\mathbf{KeyGen}(pp)$

1: $\mathbf{G} \leftarrow \mathsf{SamMat}(\rho, 0, q, n, m, \text{“G”})$
2: $\mathbf{r} \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$
3: $\mathbf{c} = \mathbf{G} \cdot \mathbf{r}$ in R_q^n ▷ Commitment to zero under $ck = \mathbf{G}$
4: **return** $(\mathsf{pk}, \mathsf{sk}) = (\mathbf{c}, \mathbf{r})$

Algorithm 4 $\mathbf{SerialGen}(\mathsf{sk})$ — for a secret key $\mathsf{sk} \in R_q^m$

1: $\mathbf{H} \leftarrow \mathsf{SamMat}(\rho, 0, q, n_s, m, \text{“H”})$
2: $\mathbf{c} = \mathbf{H} \cdot \mathbf{r}$ in $R_q^{n_s}$ where $\mathbf{r} = \mathsf{sk} \in R_q^m$ ▷ Commitment to zero under $\mathbf{H}$
3: **return** $\mathsf{sn} = \mathbf{c}$

same seed. Running the function with the same pair of (ρ', str) would result in the generation of the same entries.

The system parameters are set in Algorithm 2, where the ring size is assumed to be fixed to some N for ease of presentation. The following challenge space defines the range of the hash function:

$$C^d_{w,p} = \{x \in \mathbb{Z}[X] : \deg(x) < d \wedge \mathsf{HW}(x) = w \wedge \|x\|_\infty = p\},$$

where $\mathsf{HW}(\cdot)$ denotes the Hamming weight (i.e., the number of nonzero coefficients). This is the same set defined in [6].

Algorithms 3 and 4 generate a public-secret key pair and a serial number, respectively. Here, the public key is computed as a commitment to zero with the secret key used as the randomness, where the secret key itself is random *short* vector over R_q with absolute coefficients at most $\mathcal{B}$. Similarly, the serial number is also computed as a commitment to zero using the secret key as the randomness but, now, under the commitment key $\mathbf{H}$.

Minting a coin is presented in Algorithm 5, where a coin is computed as a commitment to the bit vector of an input amount. The commitment key used here is the same as the one in **KeyGen**.

Algorithm 5 Mint(amt) for amt $\in [0, 2^r - 1]$

$\mathbf{G} \leftarrow \mathsf{SamMat}(\rho, r, q, n, m, \text{“G”})$
$\mathbf{r} \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}, \quad (b_0, \ldots, b_{r-1}) \leftarrow \mathsf{Bits}(\mathsf{amt})$
$C = \mathrm{Com}_{ck}(b_0, \ldots, b_{r-1};\ \mathbf{r})$ in R_q^n where $ck = \mathbf{G}$
return $(\mathsf{cn}, \mathsf{cnk}) = (C, \mathbf{r})$

Algorithm 6 BinaryCommit ▷ Commitment Step of Binary Proof

INPUT: $t \in \mathbb{Z}^+$; $\{(s_j, \mathrm{Bool}_j, (b_0^{(j)}, \ldots, b_{s_j-1}^{(j)}), \mathcal{B}_j)\}_{j=0}^{t-1}$ where $s_j \in \mathbb{Z}^+, \mathrm{Bool}_j \in \{\text{True}, \text{False}\}, b_i^{(j)} \in \{0, 1\}; \mathcal{B}, \hat{\mathcal{B}}_{\text{big}} \in \mathbb{Z}^+$.
OUTPUT: $(\mathbf{r}_a, \mathbf{r}_b), (A, B), \{(a_0^{(j)}, \ldots, a_{s_j-1}^{(j)})\}_{j=0}^{t-1}$ where $\mathbf{r}_a, \mathbf{r}_b \in R_{\hat{q}}^{\hat{m}}, A, B \in R_{\hat{q}}^{\hat{n}}$ and $a_i^{(j)} \in R_{\hat{q}}$.

$ck = \hat{\mathbf{G}} \leftarrow \mathsf{SamMat}(\rho, v, \hat{q}, \hat{n}, \hat{m}, \text{“Gbig”})$ for $v = 2 \cdot \left(\sum_{j=0}^{t-1} s_j\right)$
$\mathbf{r}_b \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot \hat{m}}$ and $\mathbf{r}_a \leftarrow \{-\hat{\mathcal{B}}_{\text{big}}, \ldots, \hat{\mathcal{B}}_{\text{big}}\}^{d \cdot \hat{m}}$
for $j = 0, \ldots, t-1$ **do** ▷ Iterate over each bit sequence
  if $\mathrm{Bool}_j = \text{True}$ **then** ▷ The case of HW being 1.
    if $b_0^{(j)} = 0$, **then** $i^* = -1$ ▷ Out of $[1, s_j - 1]$
    else $i^* \leftarrow \{1, \ldots, s_j - 1\}$ and $a_{i^*}^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    for $i = 1, \ldots, s_j - 1$ and $i \neq i^*$ **do** ▷ i starts from 1.
      if $b_i^{(j)} = 0$, **then** $a_i^{(j)} \leftarrow \{-(\mathcal{B}_j - p), \ldots, \mathcal{B}_j - p\}^d$
      else $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    end for
    $a_0^{(j)} = -\sum_{i=1}^{s_j-1} a_i^{(j)}$
  else
    for $i = 0, \ldots, s_j - 1$ **do** ▷ i starts from 0.
      $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    end for
  end if
end for
$\mathbf{b} = \left(b_0^{(0)}, \ldots, b_{s_{t-1}-1}^{(t-1)}\right), \mathbf{a} = \left(a_0^{(0)}, \ldots, a_{s_{t-1}-1}^{(t-1)}\right)$
$\mathbf{c} = \left(a_0^{(0)}(1 - 2b_0^{(0)}), \ldots, a_{s_{t-1}-1}^{(t-1)}(1 - 2b_{s_{t-1}-1}^{(t-1)})\right)$
$\mathbf{d} = \left(-(a_0^{(0)})^2, \ldots, -(a_{s_{t-1}-1}^{(t-1)})^2\right)$
$B = \mathrm{Com}_{ck}(\mathbf{b}, \mathbf{c};\ \mathbf{r}_b), A = \mathrm{Com}_{ck}(\mathbf{a}, \mathbf{d};\ \mathbf{r}_a)$ in $R_{\hat{q}}^{\hat{n}}$ with $ck = \hat{\mathbf{G}}$
return $(\mathbf{r}_a, \mathbf{r}_b), (A, B), \{(a_0^{(j)}, \ldots, a_{s_j-1}^{(j)})\}_{j=0}^{t-1}$

Spend algorithm is presented in multiple parts, Algorithms 6–10. First, some parameters are computed in Algorithm 9. Then, the “corrector” values c_is are computed in Step 7. Once the corrector values are produced, the spender creates the output coins and executes an aggregated binary proof via Algorithm 6. In general, Algorithm 6 takes as an input t sequences of bits with s_j elements. Also, each sequence has a corresponding flag Bool_j, which indicates whether the sequence should have a fixed Hamming weight. If so, the values masking zero bits are sampled directly from the accepted distribution used in rejection sampling.

Algorithm 7 Compute $p_{i,j}$ for $k \in \{1, 2\}$

INPUT: $\ell, a_{0,0}, \ldots, a_{k-1,\beta-1}$
OUTPUT: $p_{0,0}, \ldots, p_{N-1,k-1} \in R_q$

```
if k = 1 then
    for i = 0, ..., N − 1 do
        p_{i,0} = a_{0,i_0}                          ▷ i = (i_0, ..., i_{k−1}) in base β
    end for
    return (p_{0,0}, ..., p_{N−1,0})
else if k = 2 then
    for i = 0, ..., N − 1 do
        p_{i,0} = a_{0,i_0} · a_{1,i_1}
        p_{i,1} = δ_{ℓ_0,i_0} · a_{1,i_1} + δ_{ℓ_1,i_1} · a_{0,i_0}
    end for
    return (p_{0,0}, p_{0,1}, ..., p_{N−1,0}, p_{N−1,1})
end if
```

Let $\delta_{i,j}$ denote the Kronecker's delta. In Step 14 of Algorithm 9, the spender computes the unary representation of her index ℓ in base β, where the binary representation vector has a fixed Hamming weight. Thus, the flag Bool_j of these sequences is equal to "True." After this, the corrector values and the output amount bits are added to the array **b**, which is then provided as an input to the binary proof. Here in Algorithm 6, the spender runs the binary proof idea similar to the one in [6].

In Steps 26 and 27 of Algorithm 9, the spender proves that the corrector commitment C is well-formed as described in the novel balance proof idea before. After that, M ring signatures are executed to prove ownership of an account in each row of A_{in}. Here, the spender also computes a serial number for each spent account to avoid double spending. Finally, she also runs another ring signature to realize the balance proof, where $\sum_{i=0}^{S-1} \mathsf{cn}_{\mathrm{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,\ell} + C$ is shown to be a commitment to zero for the same index $\ell \in [0, N-1]$ as before. Observe that all ring signatures are executed using the same vector **p**.

Algorithm 8 summarizes the core part of the ring signature, which is similar to the one-out-of-many proof in [6]. Additionally, Algorithm 8 returns $F_0, \ldots, F_{k-1}$ used in the verification of the serial numbers when the ring signature is used to prove ownership of an account. Finally, Algorithm 9 hashes all the public information up to that step.

In the next part of **Spend** (Algorithm 10), the spender computes masked responses of the underlying zero-knowledge proofs similar to [6] and applies rejection sampling as in [11]. Then, the final output is returned as given in Algorithm 10. The $\mathsf{CK}_{\mathrm{out}}$ part of the output and the corresponding output amounts should be privately transmitted to the recipient privately.

The verification algorithm (Algorithm 11) validates the same norm checks as in **Spend** function and reconstructs the components not provided in the proof output. The consistency of the values is checked by validating if the given/computed values hash to the given challenge.

Algorithm 8 RingCommit ▷ Commitment Step of Ring Sign.

INPUT: GenSerial $\in$ {True, False}; $(P_0, \ldots, P_{N-1})$ where $P_i \in R_q^n$; $(p_{0,0}, \ldots, p_{N-1,k-1})$ where $p_{i,j} \in R_q$; $\mathcal{B}, \mathcal{B}_{\text{big,k}} \in \mathbb{Z}^+$.
OUTPUT: $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, F_0, \ldots, E_{k-1}, F_{k-1})$ where $\boldsymbol{\rho}_j \in R_q^m$, $E_j \in R_q^n$ and $F_j \in R_q^{n_s}$. F_j's are omitted when GenSerial = False.

$ck = \mathbf{G} \leftarrow \mathsf{SamMat}(\rho, 0, q, n, m, \text{“G”})$
if GenSerial = True, **then** $\mathbf{H} \leftarrow \mathsf{SamMat}(\rho, 0, q, n_s, m, \text{“H”})$
$\boldsymbol{\rho}_0 \leftarrow \{-\mathcal{B}_{\text{big,k}}, \ldots, \mathcal{B}_{\text{big,k}}\}^{d \cdot m}$
for $j = 0, \ldots, k-1$ **do**
    $\boldsymbol{\rho}_j \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$ **if** $j \neq 0$
    $R_j = \text{Com}_{ck}(\mathbf{0};\ \boldsymbol{\rho}_j)$ in R_q^n
    $E_j = \sum_{i=0}^{N-1} p_{i,j} P_i + R_j$ in R_q^n
    if GenSerial = True, **then** $F_j = \mathbf{H} \cdot \boldsymbol{\rho}_j$ in $R_q^{n_s}$
end for
if GenSerial = True, **then**
        return $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, F_0, \ldots, E_{k-1}, F_{k-1})$
return $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, \ldots, E_{k-1})$

4 MatRiCT$^+$: More Efficient Post-quantum RingCT

Very recently, MatRiCT was significantly improved by Esgin, Steinfeld, and Zhao in [8]. The improved post-quantum RingCT protocol, named MatRiCT$^+$, has significant practical as well as foundational improvements over MatRiCT (and any other existing post-quantum RingCT protocol). In terms of the practical aspects, MatRiCT$^+$ has 2–17× smaller proofs and runs 3–8× faster than MatRiCT. Moreover, the public keys, serial numbers, and system moduli in MatRiCT$^+$ are also smaller. On the more foundational side, MatRiCT$^+$ introduces a novel balance proof technique that relies on a CRT-packing[5] technique and Galois automorphisms. Moreover, the proof length of MatRiCT$^+$ scales only logarithmically in the number of input accounts, M, while MatRiCT scales linearly in M. The work [8] further introduces a new set of widely applicable tools for lattice-based cryptography. In particular, an important technical novelty in [8] is the design of an “optimal” challenge space that satisfies several desired properties simultaneously while having challenges with minimal ℓ_1-norm, which is critical for better parameter selection. We refer the reader to [8] for further details.

[5] CRT: Chinese Remainder Theorem.

Algorithm 9 Spend-I

INPUT: $M, S \in \mathbb{Z}^+$; $\mathsf{A}_{\text{in}} = (\mathsf{act}_{0,0}, \ldots, \mathsf{act}_{M-1,N-1})$ where $\mathsf{act}_{i,j} = (\mathsf{pk}_{i,j}, \mathsf{cn}_{i,j})$ is an account; $\ell \in [0, N-1]$; $(\mathsf{ask}_{0,\ell}, \ldots, \mathsf{ask}_{M-1,\ell})$ where $\mathsf{ask}_{i,\ell} = (\mathbf{r}_{i,\ell}, \mathsf{cnk}_{i,\ell}, \mathsf{amt}_{\text{in},i}) \in R_q^m \times R_q^m \times \mathbb{Z}^+$; $\mathsf{PK}_{\text{out}} = (\mathsf{pk}_{\text{out},0}, \ldots, \mathsf{pk}_{\text{out},S-1})$ where $\mathsf{pk}_{\text{out},i} \in R_q^n$; $(\mathsf{amt}_{\text{out},0}, \ldots, \mathsf{amt}_{\text{out},S-1})$ where $\mathsf{amt}_{\text{out},i} \in [0, 2^r - 1]$.

$\mathcal{B}_a = \lceil 20 \cdot pkd \rceil, \quad \mathcal{B}_r = \lceil p(S+1)rd \rceil$
$T_g = d^3 \left(\mathcal{B}_a^4 k\beta(\beta+1) + \mathcal{B}_r^4 r(S+1)\right)/(4d)$
$\mathcal{B}_{\text{big}} = \lceil 1.2 \cdot (M+S+1)\mathcal{B}pwmd \rceil, \hat{\mathcal{B}}_{\text{big}} = \lceil 8 \cdot (M+S+1)\mathcal{B}pw\hat{m}d \rceil$
$\mathcal{B}_{\text{big,k}} = \lceil 1.2 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
$\mathcal{B}'_{\text{big,k}} = \lceil 2.4 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
for $i = 0, \ldots, r-2$ **do** ▷ $c_0 = c_r = 0$
  $c_{i+1} = \left(c_i + \sum_{j=0}^{S-1} \mathsf{amt}_{\text{out},j}[i] - \sum_{j=0}^{M-1} \mathsf{amt}_{\text{in},j}[i]\right)/2$
end for
for $i = 0, \ldots, S-1$ **do**
  $(\mathsf{cn}_{\text{out},i}, \mathsf{cnk}_{\text{out},i}) \leftarrow \mathbf{Mint}(\mathsf{amt}_{\text{out},i})$
end for
$\mathsf{CN}_{\text{out}} = (\mathsf{cn}_{\text{out},0}, \ldots, \mathsf{cn}_{\text{out},S-1})$
$\mathsf{CK}_{\text{out}} = (\mathsf{cnk}_{\text{out},0}, \ldots, \mathsf{cnk}_{\text{out},S-1})$
$\mathbf{b} = \{(\beta, \text{True}, (\delta_{\ell_j,0}, \ldots, \delta_{\ell_j,\beta-1}), \mathcal{B}_a)\}_{j=0}^{k-1}$
$\mathbf{b} = \mathbf{b} \cup (r-1, \text{False}, (c_1, \ldots, c_{r-1}), \mathcal{B}_r)$
for $j = 0, \ldots, S-1$ **do**
  $\mathbf{b} = \mathbf{b} \cup (r, \text{False}, \mathsf{Bits}(\mathsf{amt}_{\text{out},j}), \mathcal{B}_r)$
end for
$ck = \mathbf{G} \leftarrow \mathsf{SamMat}(\rho, r, q, n, m, \text{“G”})$
$\mathbf{r}_c \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}, \quad \mathbf{r}_d \leftarrow \{-\mathcal{B}_{\text{big}}, \ldots, \mathcal{B}_{\text{big}}\}^{d \cdot m}$
$(\mathbf{r}_a, \mathbf{r}_b), (A, B), (a_{0,0}, \ldots, a_{k-1,\beta-1}, a_{c,1}, \ldots, a_{c,r-1}, a_{\text{out},0}^{(0)}, \ldots, a_{\text{out},r-1}^{(S-1)}) \leftarrow$
$\mathsf{BinaryCommit}(k+1+S, \mathbf{b}, \mathcal{B}, \hat{\mathcal{B}}_{\text{big}})$ ▷ $a_{c,0} = a_{c,r} = 0$
for $i = 0, \ldots, S-1$ **do**
  $\mathbf{r}_g^{(i)} \leftarrow \{-\mathcal{B}_{\text{big}}, \ldots, \mathcal{B}_{\text{big}}\}^{d \cdot m}$
  $G_i = \text{Com}_{ck}(a_{\text{out},0}^{(i)}, \ldots, a_{\text{out},r-1}^{(i)}; \mathbf{r}_g^{(i)})$ in R_q^n
end for
$C = \text{Com}_{ck}(c_0 - 2c_1, \ldots, c_{r-1} - 2c_r; \mathbf{r}_c)$ in R_q^n
$D = \text{Com}_{ck}(a_{c,0} - 2a_{c,1}, \ldots, a_{c,r-1} - 2a_{c,r}; \mathbf{r}_d)$ in R_q^n
Compute $\mathbf{p} = (p_{0,0}, \ldots, p_{N-1,k-1})$ using Alg. 7 with $(\ell, a_{0,0}, \ldots, a_{k-1,\beta-1})$
for $i = 0, \ldots, M-1$ **do**
  $\mathsf{sn}_i = \mathbf{SerialGen}(\mathbf{r}_{i,\ell})$ ▷ Not recomputed if restarted
  $(\boldsymbol{\rho}_0^{(i)}, \ldots, \boldsymbol{\rho}_{k-1}^{(i)}), (E_0^{(i)}, F_0^{(i)}, \ldots, E_{k-1}^{(i)}, F_{k-1}^{(i)}) \leftarrow$
  $\mathsf{RingCommit}(\text{True}, (\mathsf{pk}_{i,0}, \ldots, \mathsf{pk}_{i,N-1}), \mathbf{p}, \mathcal{B}, \mathcal{B}_{\text{big,k}})$
end for
for $j = 0, \ldots, N-1$ **do**
  $P_j = \sum_{i=0}^{S-1} \mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,j} + C$ in R_q^n
end for
$(\boldsymbol{\rho}_0^{(M)}, \ldots, \boldsymbol{\rho}_{k-1}^{(M)}), (E_0^{(M)}, \ldots, E_{k-1}^{(M)}) \leftarrow$
$\mathsf{RingCommit}(\text{False}, (P_0, \ldots, P_{N-1}), \mathbf{p}, \mathcal{B}, \mathcal{B}'_{\text{big,k}})$
$x = \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}})$

Algorithm 10 Spend-II ▷ No mod q or $\hat{q}$ in this function!

OUTPUT: $\mathsf{CK}_{\text{out}}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathsf{SN}$ and Π as below where $F_1^{(0)}, \ldots, F_{k-1}^{(M)} \in R_q^{n_s}$; $B \in R_{\hat{q}}^{\hat{n}}$; $C, E_1^{(0)}, \ldots, E_{k-1}^{(M)} \in R_q^n$; $x \in C_{w,p}^d$; $\mathbf{f}_1 \in R_q^{k(\beta-1)}$; $\mathbf{f}_r \in R_q^{r-1+Sr}$; $\mathbf{z}_b \in R_{\hat{q}}^{\hat{m}}$; $\mathbf{z}_c, \mathbf{z}_{\text{out},0}, \ldots, \mathbf{z}_{\text{out},S-1}, \mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(M)} \in R_q^m$

for $j = 0, \ldots, k-1$ and $i = 0, \ldots, \beta - 1$ **do**
  $f_{j,i} = x\delta_{\ell_j,i} + a_{j,i}$ ▷ masked spender index representation
end for
for $i = 1, \ldots, r-1$ **do**
  $f_{c,i} = xc_i + a_{c,i}$ ▷ masked corrector values
end for
for $j = 0, \ldots, S-1$ and $i = 0, \ldots, r-1$ **do**
  $f_{\text{out},i}^{(j)} = x \cdot \mathsf{amt}_{\text{out},j}[i] + a_{\text{out},i}^{(j)}$ ▷ masked output amount bits
end for
$\mathbf{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ ▷ $f_{j,0}$'s are excluded
if $\|\mathbf{f}_1\|_\infty > \mathcal{B}_a - p$, **then** Go to Step 20 of Alg. 9
$\mathbf{f}_r = \left(f_{c,1}, \ldots, f_{c,r-1}, f_{\text{out},0}^{(0)}, \ldots, f_{\text{out},r-1}^{(S-1)}\right)$
if $\|\mathbf{f}_r\|_\infty > \mathcal{B}_r - p$, **then** Go to Step 20 of Alg. 9
for $j = 0, \ldots, k-1$ **do**
  if $\|f_{j,0}\| > \mathcal{B}_a\sqrt{d(\beta-1)}$, **then** Go to Step 20 of Alg. 9
end for
$\mathbf{g} = \left(f_{0,0}(x - f_{0,0}), \ldots, f_{k-1,\beta-1}(x - f_{k-1,\beta-1})\right)$
$\mathbf{g} = \mathbf{g} \cup \left(f_{c,1}(x - f_{c,1}), \ldots, f_{c,r-1}(x - f_{c,r-1})\right)$
$\mathbf{g} = \mathbf{g} \cup \left(f_{\text{out},0}^{(0)}(x - f_{\text{out},0}^{(0)}), \ldots, f_{\text{out},r-1}^{(S-1)}(x - f_{\text{out},r-1}^{(S-1)})\right)$
if $\|\mathbf{g}\| > \sqrt{T_g}$, **then** Go to Step 20 of Alg. 9
$\mathbf{z}_b = x\mathbf{r}_b + \mathbf{r}_a$ ▷ $\hat{m}$-dimensional
if $\|\mathbf{z}_b\|_\infty > \hat{\mathcal{B}}_{\text{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 9
$\mathbf{z}_c = x\mathbf{r}_c + \mathbf{r}_d$
for $i = 0, \ldots, S-1$ **do**
  $\mathbf{z}_{\text{out},i} = x\mathbf{r}_{\text{out},i} + \mathbf{r}_g^{(i)}$ where $\mathbf{r}_{\text{out},i} = \mathsf{cnk}_{\text{out},i}$
end for
if $\|(\mathbf{z}_c, \mathbf{z}_{\text{out},0}, \ldots, \mathbf{z}_{\text{out},S-1})\|_\infty > \mathcal{B}_{\text{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 9
for $i = 0, \ldots, M-1$ **do**
  $\mathbf{z}^{(i)} = x^k\mathbf{r}_{i,\ell} - \sum_{j=0}^{k-1} x^j \boldsymbol{\rho}_j^{(i)}$
  if $\|\mathbf{z}^{(i)}\|_\infty > \mathcal{B}_{\text{big,k}} - \mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 9
end for
$\mathbf{z}^{(M)} = x^k\mathbf{r}_{M,\ell} - \sum_{j=0}^{k-1} x^j \boldsymbol{\rho}_j^{(M)}$ where $\mathbf{r}_{M,\ell} = \sum_{i=0}^{S-1} \mathbf{r}_{\text{out},i} - \sum_{i=0}^{M-1} \mathsf{cnk}_{i,\ell} + \mathbf{r}_c$
if $\|\mathbf{z}^{(M)}\|_\infty > \mathcal{B}'_{\text{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 9
return $\mathsf{CK}_{\text{out}}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathsf{SN} = (\mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1})$ and $\Pi = (B, C, E_1^{(0)}, \ldots, E_{k-1}^{(M)}, F_1^{(0)}, \ldots, F_{k-1}^{(M)}, x, \mathbf{f}_1, \mathbf{f}_r, \mathbf{z}_b, \mathbf{z}_c, \mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(M)}, \mathbf{z}_{\text{out},0}, \ldots, \mathbf{z}_{\text{out},S-1})$

Algorithm 11 Verify

INPUT: $M, S \in \mathbb{Z}^+$; $\mathsf{A}_{\text{in}} = (\mathsf{act}_{0,0}, \ldots, \mathsf{act}_{M-1,N-1})$ where $\mathsf{act}_{i,j} = (\mathsf{pk}_{i,j}, \mathsf{cn}_{i,j})$ is an account; $\mathsf{PK}_{\text{out}} = (\mathsf{pk}_{\text{out},0}, \ldots, \mathsf{pk}_{\text{out},S-1})$; $\mathsf{CN}_{\text{out}} = (\mathsf{cn}_{\text{out},0}, \ldots, \mathsf{cn}_{\text{out},S-1})$; $\Pi = (B, C, E_1^{(0)}, \ldots, E_{k-1}^{(M)}, F_1^{(0)}, \ldots, F_{k-1}^{(M)}, x, \mathbf{f}_1, \mathbf{f}_r, \mathbf{z}_b, \mathbf{z}_c, \mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(M)}, \mathbf{z}_{\text{out},0}, \ldots, \mathbf{z}_{\text{out},S-1})$; $\mathsf{SN} = (\mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1})$.
OUTPUT: True/False

if $\|\mathbf{f}_1\|_\infty > \mathcal{B}_a - p$, **then return** False
if $\|\mathbf{f}_r\|_\infty > \mathcal{B}_r - p$, **then return** False
Parse $\mathbf{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ as in Alg. 10
Parse $\mathbf{f}_r = (f_{c,1}, \ldots, f_{c,r-1}, f_{\text{out},0}^{(0)}, \ldots, f_{\text{out},r-1}^{(S-1)})$ as in Alg. 10
for $j = 0, \ldots, k-1$ **do**
    $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}$
    if $\|f_{j,0}\| > \mathcal{B}_a\sqrt{d(\beta-1)}$, **then return** False
end for
Compute $\mathbf{g}$ as in Alg. 10
if $\|\mathbf{g}\| > \sqrt{T_g}$, **then return** False
if $\|\mathbf{z}_b\|_\infty > \hat{\mathcal{B}}_{\text{big}} - \mathcal{B}pw$, **then return** False
if $\|(\mathbf{z}_c, \mathbf{z}_{\text{out},0}, \ldots, \mathbf{z}_{\text{out},S-1})\|_\infty > \mathcal{B}_{\text{big}} - \mathcal{B}pw$, **then return** False
if $\|(\mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(M-1)})\|_\infty > \mathcal{B}_{\text{big,k}} - \mathcal{B}(pw)^k$, **then return** False
if $\|\mathbf{z}^{(M)}\|_\infty > \mathcal{B}'_{\text{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then return** False
$\mathbf{f} = (f_{0,0}, \ldots, f_{k-1,\beta-1}) \cup \mathbf{f}_r$ ▷ $f_{j,0}$'s are included.
$A = \text{Com}_{ck}(\hat{\mathbf{f}}, \mathbf{g};\ \mathbf{z}_b) - xB$ in $R_{\hat{q}}^{\hat{n}}$ for $ck = \hat{\mathbf{G}}$
$D = \text{Com}_{ck}(f_{c,0} - 2f_{c,1}, \ldots, f_{c,r-1} - 2f_{c,r};\ \mathbf{z}_c) - xC$ in R_q^n where $f_{c,0} = f_{c,r} = 0$ and $ck = \mathbf{G}$ here and in the rest
for $i = 0, \ldots, S-1$ **do**
    $G_i = \text{Com}_{ck}(f_{\text{out},0}^{(i)}, \ldots, f_{\text{out},r-1}^{(i)};\ \mathbf{z}_{\text{out},i}) - x\mathsf{cn}_{\text{out},i}$ in R_q^n
end for
for $l = 0, \ldots, M-1$ **do**
    $E_0^{(l)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)\mathsf{pk}_{l,i}\right] - \text{Com}_{ck}(\mathbf{0};\ \mathbf{z}^{(l)}) - \sum_{j=1}^{k-1} E_j^{(l)} x^j$ in R_q^n
    where $i = (i_0, \ldots, i_{k-1})$ in base β
    $F_0^{(l)} = x^k\mathsf{sn}_l - \mathbf{H}\cdot\mathbf{z}^{(l)} - \sum_{j=1}^{k-1} F_j^{(l)} x^j$ in $R_q^{n_s}$
end for
for $j = 0, \ldots, N-1$ **do**
    $P_j = \sum_{i=0}^{S-1}\mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1}\mathsf{cn}_{i,j} + C$ in R_q^n
end for
$E_0^{(M)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)P_i\right] - \text{Com}_{ck}(\mathbf{0};\ \mathbf{z}^{(M)}) - \sum_{j=1}^{k-1} E_j^{(M)} x^j$ in R_q^n
if $x \neq \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}})$, **then return** False
return True

References

1. M. Ajtai, Generating hard instances of lattice problems (extended abstract), in *STOC* (ACM, 1996), pp. 99–108
2. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in *IEEE S&P 2018*, pp. 315–334 (2018)

3. V. Buterin, Understanding serenity, part i: Abstraction, 2015. https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction/, accessed on Mar 2, 2021
4. M. F. Esgin, *Practice-Oriented Techniques in Lattice-Based Cryptography*. PhD thesis, Monash University, 5 2020. https://bridges.monash.edu/articles/Practice-Oriented_Techniques_in_Lattice-Based_Cryptography/12279728
5. M. F. Esgin, N. K. Nguyen, G. Seiler, Practical exact proofs from lattices: New techniques to exploit fully-splitting rings, in *ASIACRYPT (2)*, vol. 12492 of *Lecture Notes in Computer Science* (Springer, 2020), pp. 259–288
6. M. F. Esgin, R. Steinfeld, J. K. Liu, D. Liu, Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications, in *CRYPTO (1)*, vol. 11692 of *Lecture Notes in Computer Science* (Springer, 2019), pp. 115–146 . (Full version at ia.cr/2019/445)
7. M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, D. Liu, Short lattice-based one-out-of-many proofs and applications to ring signatures, in *ACNS*, *Lecture Notes in Computer Science* (Springer, 2019), pp. 67–88. (Full version at ia.cr/2018/773)
8. M. F. Esgin, R. Steinfeld, R. K. Zhao, MatRiCT+: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545, 2021. https://eprint.iacr.org/2021/545
9. M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, D. Liu, MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol, in *ACM Conference on Computer and Communications Security* (ACM, 2019), pp. 567–584. (Full version at ia.cr/2019/1287)
10. J. K. Liu, V. K. Wei, D. S. Wong, Linkable spontaneous anonymous group signature for ad hoc groups, in *Australasian Conference on Information Security and Privacy* (Springer, 2004), pp. 325–335
11. V. Lyubashevsky, Fiat-shamir with aborts: Applications to lattice and factoring-based signatures, in *ASIACRYPT* (Springer, 2009), pp. 598–616
12. G. Maxwell, Confidential transactions (2015). https://people.xiph.org/~greg/confidential_values.txt (Accessed 09/05/2016)
13. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. Report, 2008
14. S. Noether, Ring Signature Confidential Transactions for Monero. Cryptology ePrint Archive, Report 2015/1098, 2015. http://eprint.iacr.org/
15. S. Noether, A. Mackenzie, et al., Ring confidential transactions. Ledger **1**, 1–18 (2016)
16. O. Regev, On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009). Preliminary version in STOC 2005
17. P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *FOCS* (IEEE Computer Society, 1994), pp. 124–134
18. S.-F. Sun, M. H. Au, J.K. Liu, T. H. Yuen, RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero, in *ESORICS 2017*, pp. 456–474 (2017)
19. W.A.A. Torres, V. Kuchta, R. Steinfeld, A. Sakzad, J.K. Liu, J. Cheng, Lattice ringct V2.0 with multiple input and multiple output wallets, in *ACISP*, vol. 11547 of *Lecture Notes in Computer Science* (Springer, 2019), pp. 156–175
20. W.A.A. Torres, R. Steinfeld, A. Sakzad, J.K. Liu, V. Kuchta, N. Bhattacharjee, M.H. Au, J. Cheng, Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0), in *ACISP*, vol. 10946 of *Lecture Notes in Computer Science* (Springer, 2018), pp. 558–576
21. N. van Saberhagen, Cryptonote v 2.0, 2013. *CryptoNote*, pp. 04–13 (2013)
22. T. H. Yuen, S. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, D. Gu, RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security, in *Financial Cryptography and Data Security*, vol. 12059 of *LNCS* (Springer, 2020), pp. 464–483

Algorand Blockchain

Arash Mirzaei, Amin Sakzad, Ron Steinfeld, and Jiangshan Yu

1 Introduction

Bitcoin [24] is the first blockchain-based cryptocurrency where transactions are exchanged between users on the peer-to-peer Bitcoin network and are added to a publicly verifiable distributed ledger. To append a block of transactions to the ledger, the blockchain nodes, called *miners*, compete with each other where the success probability of each node is proportional to his computational power and the power amount is proved by solving a computational hash puzzle. This mechanism is called *proof of work* (PoW). Although elegantly designed, Bitcoin has some technical problems. These problems are as follows:

- **Energy waste:** Bitcoin miners spend computational resources to solve a cryptographic puzzle. Such computations cause electricity consumption and its consequent environmental issues [27].
- **Miner concentration:** Decentralization has been one of the main attractions of Bitcoin. However, now, a few mining pools own more than 50% of the total Bitcoin hashrate power [1]. This can threaten the Bitcoin's security.
- **Forks:** In Bitcoin blockchain, forks are probable, meaning that one user's view of latest blocks of the blockchain might be different from that of another user. Therefore, it is always possible that latest blocks of the blockchain will be replaced with other ones. Thus, to ensure that a particular block is not removed from the blockchain, users must wait about 1 hour for that block to get six blocks deep in the blockchain [17]. Miners also should always mine on the longest chain (longest-chain rule). As long as less than 50% of total hashing power is controlled

A. Mirzaei · A. Sakzad (✉) · R. Steinfeld · J. Yu
Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Melbourne, VIC, Australia
e-mail: amin.sakzad@monash.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_6

by malicious miners, the chain with the most accumulated work is produced by honest miners.
- **Throughput:** The rate at which transactions are added to the blockchain is called throughout. Bitcoin throughput is approximately seven transactions per second [10]. This is limited because Bitcoin blocks are mined at the rate of approximately 1 block per 10 minutes and the size of each block is at most 1MB. Increase in block size or decrease in inter-block interval increases the rate at which forks occur [12].

The *proof of stake* (PoS) [5] is a consensus mechanism that eliminates the need for solving a computational hash puzzle and hence achieves energy efficiency. In PoS systems, a cryptographic method is usually used to elect a leader from the stakeholders for each round of the system where the success probability of a node to be elected as the leader is proportional to the amount of stake he uses during the leader election procedure. The leader's role is to append a block to the ledger.

Algorand relies on Byzantine consensus protocol to avoid forks. In Byzantine consensus protocols, a relatively small *committee* runs each step of the protocol. To avoid Sybil attacks [15] and benefit from merits of PoS systems, committee for each step is selected from Algorand users based on their money in their account. If committee members are known in advance, an adversary may launch a DoS attack against them, also known as *targeted DoS* attack [30]. To avoid this attack, committee members for each step of the Algorand protocol are selected in a private and noninteractive way, meaning that only the user can privately determine if he has been selected as a committee member for the next step and can publicly prove it to other users. Table 1 provides a quick comparison of some blockchains. In this table, *consensus finality* is provided by a blockchain if it is guaranteed that a valid block, appended to the blockchain, is never eliminated from the blockchain. Also, the term *fault tolerance* specifies the faulty conditions the network can tolerate where, for PoW and PoS blockchains, it is generally determined based on adversarial hashing power and maliciously possessed stakes, respectively.

2 Algorand Overview

Similar to Bitcoin, Algorand has a logical unit called round in which one block is appended to the blockchain. Each block consists of multiple transactions and a pointer to the previous block. Please refer to Fig. 1. In each round, all users firstly execute a noninteractive algorithm called *cryptographic sortition*. Each user executes this algorithm to check if she has been selected by the algorithm as one of the candidates for block proposal (B. P.). The probability that each user is selected is proportional to the money that the user has in her account. Since we have several candidates for block proposal but only one block per round is required, block candidates are randomly prioritized by the system using a parameter called *priority*. Then, each candidate uses the pending transactions to create a block and

Table 1 Comparison of blockchain consensus protocols

Blockchain	Leader selection	Consensus	Consensus finality	Fault tolerance	Targeted DoS resistance
Bitcoin [24]	PoW	Longest-chain rule	×	< 50% hashrate	✓
Monero [25]	PoW	Longest-chain rule	×	< 50% hashrate	✓
Ethereum [29]	PoW	A variation of GHOST[a] rule	×	< 50% hashrate	✓
Cardano [11]	PoS-based election	Longest-chain rule	×	< 50% stake	✓[b]
Tendermint [6]	PoS-based round-robin	Byzantine agreement	✓	< 33% stake	×
Algorand [17]	PoS-based election	Byzantine agreement	✓	< 33% stake	✓

[a]Greedy heaviest-observed sub-tree
[b]Ouroboros Praos [11] or Ouroboros Genesis [4] protocols achieve targeted DoS resistance. The earlier version of Ouroboros [18] does not resist against targeted DoS attack

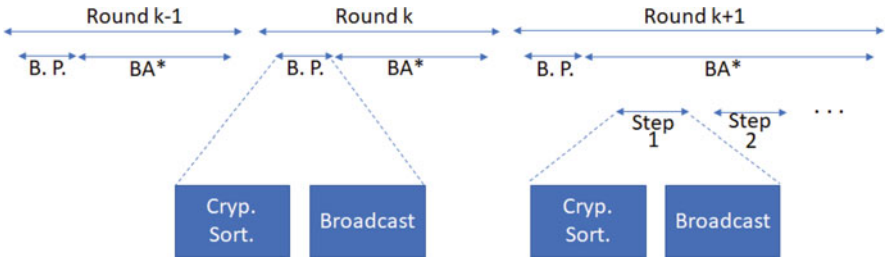


Fig. 1 High-level view of Algorand system

broadcasts it in the network, together with her priority. Since one user's observed highest-priority block might be different from that of another user, Algorand users execute an agreement protocol called $BA\star$ to agree on one of the proposed blocks. $BA\star$ consists of several steps. Due to scalability issues, $BA\star$ cannot be performed among all Algorand users. Thus, users execute the cryptographic sortition algorithm for each step of $BA\star$. By doing so, each user can check if she has been selected by the sortition algorithm as a committee member for that particular step of the agreement protocol. The committee members of the first step initialize $BA\star$ with their observed highest-priority block. In each step, committee members broadcast a message including their proof of selection. If at the end of each step an agreement has not been reached, a new step will start with new committee members selected by a new execution of cryptographic sortition algorithm. Figure 1 shows a high-level view of different stages of Algorand system. The cryptographic sortition algorithm and $BA\star$ protocol are explained in more details in Sects. 4 and 5.

3 Algorand Assumptions

Cryptographic primitives Algorand relies on security of its cryptographic primitives: (1) digital signature and (2) hash function. The digital signature is *existentially unforgeable under chosen message attacks*. It also satisfies $uniqueness$, meaning that it is hard to find two different signatures s and s' such that both are verified under a message m and a public key pk. The hash function, H, is modeled as a *random oracle*.

Communication network assumptions To provide liveness, Algorand assumes that the communication network between users is "strongly synchronous," i.e., messages sent by a large proportion of honest users (e.g., 95%) are delivered to most other honest users (e.g., 95%) within a known time bound. Algorand also achieves safety assuming that the network is "weakly synchronous." A network is called weakly synchronous if the adversary can fully control the network for a long but bounded time interval (e.g., 1 day or 1 week), but after each asynchrony period, there is a sufficiently long period (e.g., with length of few hours or a day) in which the network is strongly synchronous.

Adversary model Algorand considers an adaptive adversary that is polynomial time. The adversary can corrupt and coordinate any user at any time such that more than $\frac{2}{3}$ of the total Algorand currency is held by honest users who have not been compromised by the adversary. At the beginning of each round, the adversary can select the users she wants to control. Then, for that round, she can learn the honest uncompromised users' messages and build the compromised users' messages accordingly. However, the adversary cannot tamper with the messages submitted by honest users.

4 Cryptographic Sortition

As mentioned earlier, cryptographic sortition algorithm is executed in the following stages: (1) at the start of each round to select a subset of users as block proposal candidates and (2) at the start of each step of $BA\star$ to select committee members performing this step. Thus, contrary to Bitcoin where all miners try to win the competition of solving the cryptographic puzzle and mine the next block, for Algorand, different participants of each round are selected using the cryptographic sortition algorithm. By using cryptographic sortition, Algorand actually aims to base its integrity on the proof-of-stake (PoS) principle, where the blockchain contents are controlled based on majority of stake ownership rather than on majority of computational power as in proof-of-work (PoW)-based blockchains. Running the cryptographic sortition algorithm is the only type of computation that is performed by all Algorand holders, and since it can be completed using

lightweight computational operations, Algorand can resolve the energy waste issue of the Bitcoin.

Cryptographic sortition must select users with probability proportional to the money in their accounts. Let the amount of money or equivalently the number of Algorand tokens (called $ALGO$) in the user i's account be denoted by w_i and the total amount of Algorand tokens be shown by $W = \sum_i w_i$.

Since users are going to be selected with probability that is proportional to their money, each of w_i Algorand tokens in user i's account is considered as a sub-user of user i and might be independently selected by the cryptographic sortition algorithm with probability of $p = \frac{\tau}{W}$, where τ is the expected total number of sub-users which are going to be selected by the algorithm. Thus, the probability that k out of w_i sub-users of user i is selected follows the binomial distribution, $B(k; w_i, p) = \binom{k}{w_i} p^k (1-p)^{w_i - k}$, where $\sum_{k=0}^{w_i} B(k; w_i, p) = 1$. Figure 2 shows $B(k; w_i, p)$ for different values of k, w_i, and p. It is obvious from the figure that the higher the balance in a user's account (w_i), the higher the probability that at least one sub-user of the user i will be selected during one execution of the sortition algorithm (which is equivalent to the non-blue region in each bar).

Now, assume that user i obtains a single publicly verifiable random value from the range $[0, 1)$. If the random value belongs to the range $[0, B(0; w_i, p))$, it means that none of sub-users of user i have been selected. Similarly, if the random value is in the range $[\sum_{k=0}^{j-1} B(k; w_i, p), \sum_{k=0}^{j} B(k; w_i, p))$, then j sub-users of user i have been selected.

We have assumed that user i obtains a single random value whose validity can be verified by the public. To realize this goal, the cryptographic sortition algorithm uses a verifiable random function (VRF) (please see [23] for a syntax and security model) to generate such a random value called $hash$ of length $hashlen$ bits. The algorithm then computes $\frac{hash}{2^{hashlen}}$ to scale $hash$ to the range $[0, 1)$ and then finds the value of j for which $\frac{hash}{2^{hashlen}}$ is in the range $[\sum_{k=0}^{j-1} B(k; w_i, p), \sum_{k=0}^{j} B(k; w_i, p))$ (with $\sum_{k=0}^{-1} B(k; w_i, p) = 0$). See 2 for a visualization of these intervals for different values of w_i and p. Algorithm 1 shows the cryptographic sortition algorithm.

Algorithm 1: Sortition($sk, seed, \tau, role, w, W$)

$\langle hash, \pi \rangle \leftarrow \mathrm{VRF}_{sk}(seed, role)$
$p \leftarrow \frac{\tau}{W}$
$j \leftarrow 0$
while $\frac{hash}{2^{hashlen}} \notin [\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p))$ **do**
  j ++
end
return $\langle hash, \pi, j \rangle$

In sortition algorithm, each user performs $\langle hash, \pi \rangle \leftarrow \mathsf{VRF}_{sk}(seed, role)$ to compute $hash$ and π where sk is the user's private key; $role$ might be a combination of the round number, step number, and purpose of execution of cryptographic

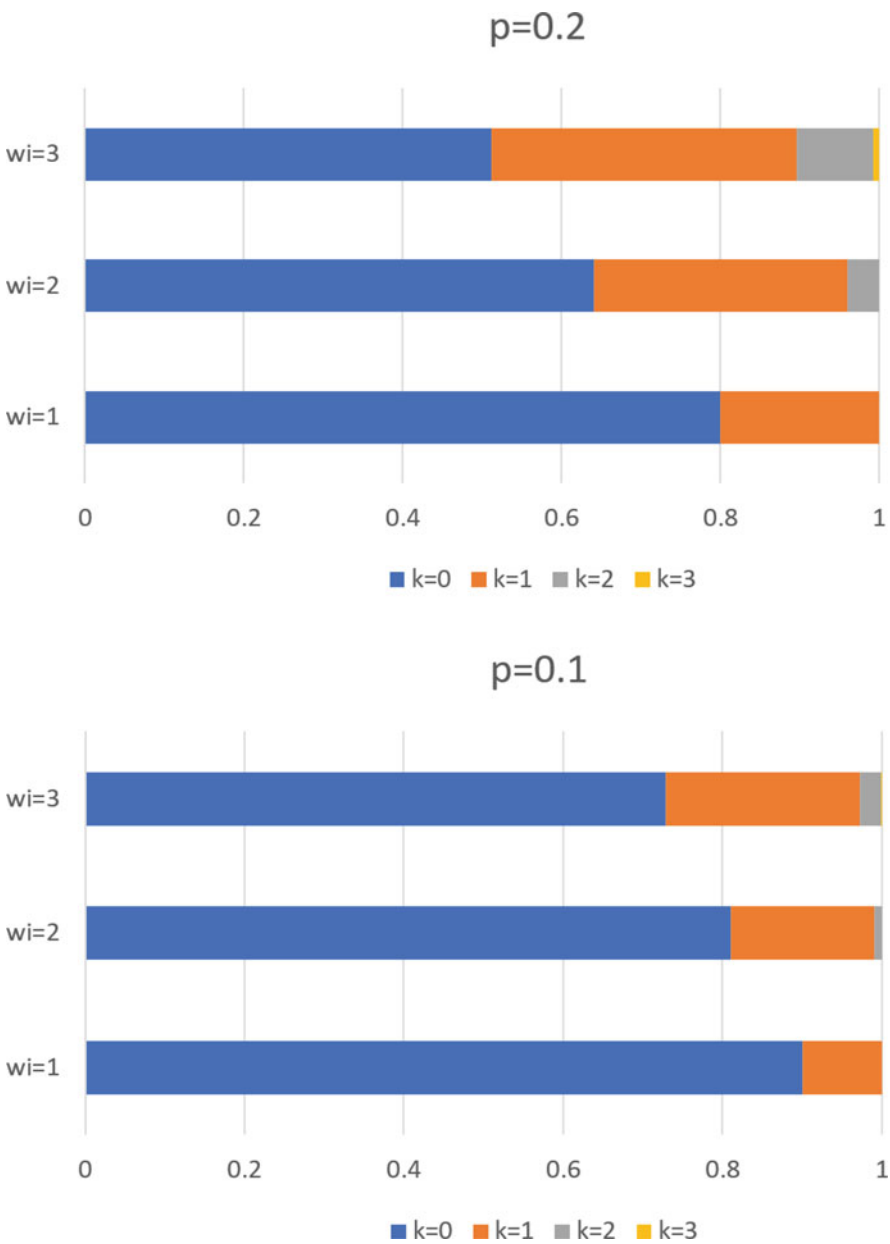


Fig. 2 $B(k; w_i, p)$ for different values of k, w_i, and p

sortition (selecting either block proposer or committee member); and *seed* is a publicly known random number which is updated per round. We will explain the seed update procedure later in this section. Application of *hash* was earlier

explained. The parameter π can be used in VerifySort algorithm (see Algorithm 2) to publicly verify that the number of selected sub-users j, claimed by the user, is in fact correct/valid. To do so, firstly, the function VerifyVRF is performed using the public key of the user to check whether $hash$ and π have been correctly generated using the user's private key and concatenation of $seed$ and user's $role$. If this function outputs 0, it means that the $hash$ and π of the user are not valid and hence VerifySort algorithm also outputs 0 meaning that the user has not been selected at all. Otherwise, similar to sortition algorithm, there is a while loop using which the number of selected sub-users j of a particular user is computed. The cryptographic sortition verification algorithm, VerifySort, is shown in Algorithm.2.

Algorithm 2: VerifySort$(pk, hash, \pi, seed, \tau, role, w, W)$

if $\neg$VerifyVRF$_{pk}(hash, \pi, seed \| role)$ **then**
  return 0
end
$p \leftarrow \frac{\tau}{W}$
$j \leftarrow 0$
while $\frac{hash}{2^{hashlen}} \notin [\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p))$ **do**
  $j{+}{+}$
end
return j

The sortition and VerifySort algorithms require a publicly known random called $seed$. Each Algorand round requires a $seed$ that is known to everyone and is updated such that an adversary cannot control its value in her favor. The initial seed, $seed_0$, is randomly selected by initial parties (with known initial public keys) using the distributed random number generation method proposed in [8]. To update the value of $seed$ from the round $r-1$ to the round r, during the block proposal (B. P.) stage for the round $r-1$, each block proposal candidate computes a proposed $\langle seed_r, \pi \rangle \leftarrow \mathsf{VRF}_{sk}(seed_{r-1} \| r)$ and puts it in his proposed block. If the proposed seed in the agreed block is invalid, the block is considered as an empty block, and the seed for the next round is computed as $seed_r = H(seed_{r-1} \mid r)$. Although $seed$ is updated per round, for security reasons, sortition and VerifySort algorithms in round r are executed using $seed_{r-1-(r \pmod R)}$, meaning that the $seed$ that is used in these algorithms is refreshed once per R rounds, where R is a parameter of the protocol.

5 The Protocol $BA\star$

In the block proposal stage, users receive different blocks from block proposal candidates and select the highest priority one. However, since the selected blocks might be different among users, it is required that Algorand users reach an

agreement on the block that is going to be added to the blockchain. To do so, they perform a Byzantine agreement protocol called $BA\star$. Usage of $BA\star$ removes the possibility of forks in the Algorand blockchain, and hence transactions get finalized once they are added to the blockchain. In other words, transactions in an Algorand block can be relied once the block appears on the blockchain. Furthermore, $BA\star$ is really fast, and hence Algorand achieves 125× Bitcoin's throughput [17].

Each user initializes $BA\star$ with her observed highest-priority block. $BA\star$ consists of two phases, where the first phase, called GC, contains two steps, and the second phase, called $BBA\star$, includes several repeated steps. Users repeat steps of the second phase until agreement is reached. Each step of GC and $BBA\star$ starts with a cryptographic sortition using which committee members for that particular step are selected. It means that committee members for each step are highly likely to be different from those for the next one. This property is called *player replaceability*. We will provide further explanation regarding importance of this property later in this section. Furthermore, each user can draw the *seed* from the block agreed in the ith round to check her presence in the committee of each step of the $(i+1)$th round.

We firstly explain the second phase, i.e., $BBA\star$, which is the binary Byzantine agreement protocol, and then state how by adding the first phase, i.e., GC, $BBA\star$ transforms to an arbitrary-value Byzantine agreement protocol. As mentioned earlier in Sect. 3, Algorand works under weak synchrony assumption. However, to simplify explaining the protocol, the version of agreement protocol that we explain here (derived from [22]) is slightly different from Algorand agreement protocol and works under assumption that the network is complete and synchronous.[1]

Before explaining $BBA\star$, we provide Algorand's definition of Byzantine agreement protocols.

Definition 1 ([8]) Let n and t be positive integers, such that $n = 3t + 1$. A protocol P is an arbitrary-value (respectively, binary) $(n; t)$ Byzantine agreement protocol with soundness $\sigma \in (0; 1]$, if, for every set of values V (respectively, for $V = \{0, 1\}$), and any t-adversary A,[2] in an execution e of P with A with n players, in which every player i starts with an initial value $v_i \in V$, every honest player j halts with probability 1, outputting a value $out_j \in V$, so as to satisfy, with probability σ, the following two properties:

1. *Agreement*. $out_i = out_j$ for all honest players i and j.
2. *Consistency*. If, for some v, $v_i = v$ for all honest players i, then $out_j = v$ for all honest players j.

Agreement means that at the end of the protocol all honest players agree on a common value, and *consistency* means that if all honest users initiate the protocol with a common value v, the final agreement is also reached on v.

[1] A network is synchronous if all messages sent by an honest node are delivered within a known bound of delay.

[2] Meaning that at most t out of n players in execution e of P are malicious

The protocol $BBA\star$ is performed in several steps, and in each step, each player sends her input message for that step to other players. Then she counts the number of players that have sent her a given message in order to update her input message for the next step. The notation $\#_i^s(v)$ shows the number of messages with value v that have been received by player i in step s. Each player i receives exactly one message from each player j. Before explaining $BBA\star$, some intuition will be provided to help the reader to understand logic of $BBA\star$. To do so, we will firstly define a protocol P and then change it step by step. Then, $three$ different variants of P are used to create the $three$-step loop protocol $BBA\star$. For both P and $BBA\star$, each user has a Boolean input and outputs a Boolean value.

Protocol P with magic coin Assume that player i has an initial bit b_i and updates this variable through different steps of the protocol. This bit represents the current opinion of user i about her final output out_i. Protocol P with magic coin is as follows:

1. Every player i sends the bit b_i to all players, including himself.
2. A magically random bit c (called magic coin) appears in the sky that can be observed by everyone.
3. Every player i updates b_i as follows:

   3.1 If $\#_i^1(0) \geq 2t+1$, $b_i = 0$.
   3.2 Else, if $\#_i^1(1) \geq 2t+1$, $b_i = 1$.
   3.3 Else, $b_i = c$

It can be easily seen that if at the start of the protocol all honest players are in agreement on a bit b (i.e., $b_i = b$ for all honest players), then they will remain in agreement on b by the end of protocol. Otherwise, they will be in agreement at the end of the protocol with probability $\frac{1}{2}$. For example, assume that for two players i and j with $i \neq j$, we have $\#_i^1(0) \geq 2t+1$ and $\#_j^1(0) < 2t+1$. Then, player i sets b_i to 0, and player j sets b_j to 0 with probability $\frac{1}{2}$ (since c equals 0 with probability $\frac{1}{2}$). It is impossible that we have $\#_i^1(0) \geq 2t+1$ and $\#_j^1(1) \geq 2t+1$.

Protocol P with frequently magic coin Now, assume that rather than having a magic coin which is random and common to everyone, we have a frequently magic coin which is random with probability of $\frac{2}{3}$. Then, it can be shown that players that are not in agreement reach a consensus with probability of $\frac{1}{3}$. If we have $\#_i^1(0) \geq 2t+1$ and $\#_j^1(0) < 2t+1$, with $i \neq j$, player i sets b_i to 0, and player j sets b_j to 0 with probability of $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$ (If the frequently magic coin is random, which occurs with probability of $\frac{2}{3}$, its value would be 0 with probability of $\frac{1}{2}$).

Protocol P with concrete coin Now, assume that no magic or frequently magic coin appears in the sky and players can create a frequently magic coin by themselves. Assume that $seed$ is a random string known to all players, each player has a key pair whose public key is known to all players, and $H(.)$ and $Sign_i(.)$ show a secure hash function and a secure digital signature using the private key of

player i, respectively. Let $SIG_i(x) \triangleq (i, x, Sign_i(x))$. Then, player i can create the concrete coin through the following protocol:

- Each player i sends the value $v_i = SIG_i(seed)$ to other players.
- Each player i finds v_m such that $H(v_m) \leq H(v_j)$ for all players j.
- Player i sets the concrete magic coin to least significant bit of $H(v_m)$.

v_m belongs to an honest player with probability $\frac{2}{3}$, and hence $lsb(H(v_m))$ is random with the same probability. Thus, $lsb(H(v_m))$ can be considered as a concrete coin. However, there is still a problem with P with concrete coin which is the fact that at the end of the protocol players cannot ensure if they are in agreement. To solve this problem, players can repeat executing the protocol, each time using the values that players outputted during the previous execution of the protocol. However, since reaching the agreement in each round is probabilistic, players cannot still determine the round at which they reach an agreement. One method to solve this issue is repeating the protocol for k times where k is a fixed large number (e.g., $k = 300$). However, this would be time-consuming because, on average, players reach an agreement by repeating the protocol 3 times, but they must practically repeat the protocol k times to guarantee the agreement.

Protocol $BBA\star$ and its analysis Another solution to the previously mentioned problem, which is used by $BBA\star$, is repeating a $three$-step protocol consisting of three correlated executions of the protocol P with concrete coin. In the first and second steps of the loop, the concrete coin is set to 0 and 1, respectively. In the third step, the genuine concrete coin will be used. The γ (starting from 0) represents the number of times the loop has been executed. Each player has a Boolean value as input. At each step of the protocol, each user broadcasts her value as a vote and counts the received votes, and if the number of counted votes for a particular value is higher than a predetermined threshold, the user changes her value to that particular value. If at a particular step a player wants to output 0 (or 1) and exit the protocol, he sends a $0\star$ (or respectively $1\star$) to all players including himself to tell every player that his vote for the next steps does not change and must be counted as 0 (or, respectively, 1). $BBA\star$ is as follows:

1. Each player i broadcasts b_i.
   - 1.1 If $\#_i^1(0) \geq 2t + 1$, then i sets b_i to 0, broadcasts $0\star$, outputs $out_i = 0$, and HALTS.
   - 1.2 If $\#_i^1(1) \geq 2t + 1$, then i sets b_i to 1.
   - 1.3 Else, i sets b_i to 0.
2. Each player i broadcasts b_i.
   - 2.1 If $\#_i^2(1) \geq 2t + 1$, then i sets b_i to 1, broadcasts $1\star$, outputs $out_i = 1$, and HALTS.
   - 2.2 If $\#_i^2(0) \geq 2t + 1$, then i sets b_i to 0.
   - 2.3 Else, i sets b_i to 1.
3. Each player i broadcasts b_i and $SIG_i(seed, \gamma)$.

3.1 If $\#_i^3(0) \geq 2t+1$, then i sets b_i to 0.
3.2 If $\#_i^3(1) \geq 2t+1$, then i sets b_i to 1.
3.3 Else, letting $S_i = \{j \in N$ who sent i a valid message in step 3$\}$, i sets b_i to $c_i^\gamma = lsb(\min_{j \in S_i} H(SIG_j(seed, \gamma)))$, increases γ by 1, and returns to step 1.

Note that at the start of the first step of $BBA\star$, each honest player i has an input b_i which can broadcast if she is selected as a member of the committee for the first step. However, for next steps, if the honest player i is selected as a committee member, she must compute b_i based on the number of votes she has received during the previous completed steps.

The following theorem states that $BBA\star$ is a binary Byzantine agreement protocol that meets the requirements stated in Definition 1 with soundness 1.

Theorem 1 ([8]) *Whenever $n \geq 3t+1$, $BBA\star$ is a binary $(n; t)$ BA protocol with soundness 1.*

This theorem can be simply proven using the following properties: (1) Based on our discussion on protocol P with concrete coin, we know that if at the start of step 3 agreement has not been reached and none of the players have still halted, they will be in agreement at the end of this step with probability $\frac{1}{3}$. (2) If at some point agreement is reached on a bit b, then agreement on b continues to hold. (3) If at step 1 (or similarly step 2) player i outputs $0\star$ and halts, agreement will be reached at the end of that step. (4) If at the start of step 1 (or similarly step 2) all players are in agreement on 0 (or, respectively, 1), all players output 0 (or, respectively, 1) and halt in that step. The reason for the second property is that if at the start of a step there is an agreement on b, all honest players broadcast b and hence all players observe at least $2t+1$ votes with value of b and thus sets b_i to b at the end of the step. To understand the reason of correctness of property 3, assume that the mentioned scenario is the case and at step 1.1 the honest player i outputs $0\star$ and halts. It means that player i has observed at least $2t+1$ votes with a value of 0 at the communication phase of step 1. Since we have at most t dishonest parties, it means that at least $t+1$ honest players have broadcast 0 and no party observes $2t+1$ votes with a value of 1. Thus, all parties set b_j to 0 either in step 1.1 or 1.3.

Protocol GC and its analysis Protocol GC starts after the end of the block proposal stage. This protocol has two steps where each step has its own committee members selected by the cryptographic sortition algorithm executed at the start of the step. Input value of user i is shown by v_i which is the block with the highest priority that user has observed during the block proposal stage. Protocol GC is as follows:

1. Each player i broadcasts v_i.
2. If for some x $\#_i^1(x) \geq 2t+1$, player i broadcasts x.

   2.1 Player i outputs the pair (v_i', g_i) computed as follows:

- If for some x $\#_i^2(x) \geq 2t+1$, then $v_i' = x$ and $g_i = 2$.
- Else, if for some x $\#_i^2(x) \geq t+1$, then $v_i' = x$ and $g_i = 1$.
- Else, $v_i' = \perp$ and $g_i = 0$.

It can be easily seen that GC provides the following properties:

1. It is impossible that two honest players i and j with $i \neq j$ output g_i and g_j such that $|g_i - g_j| = 2$.
2. For all honest players i and j with both $g_i, g_j > 0$, we have $v_i' = v_j'$.
3. If all honest players start the protocol with the common value v, the output for all of them will be the pair $(v, 2)$.

To show the correctness of the first property, assume that player i outputs $g_i = 2$. It means that player i has observed $2t+1$ votes with a value of x where we know at least $t+1$ of them has been cast by honest users. So any honest player j with $j \neq i$ has also received at least $t+1$ votes with a value of x, and hence it is impossible that he outputs $g_j = 0$. The second and third properties can also be shown to be correct similarly.

$BA\star$ and its analysis As mentioned earlier, $BA\star$ is the combination of the protocols GC and $BBA\star$. Assuming that each user has an initial value v_i, the protocol $BA\star$ will be as follows:

1. Each player i performs the protocol GC to compute the pair (v_i', g_i).
2. Each player i performs the protocol $BBA\star$ with $b_i = 0$ if $g_i = 2$ and with $b_i = 1$ otherwise and computes out_i.
3. Each player i outputs v_i', if $out_i = 0$, and $\perp$ otherwise.

The following theorem states that $BA\star$ is an arbitrary-value Byzantine agreement protocol that meets the requirements stated in Definition 1 with soundness 1.

Theorem 2 ([8]) *Whenever $n \geq 3t+1$, $BA\star$ is a (n, t) BA protocol with soundness 1.*

Proof We start with consistency. Assume that all honest parties start the protocol $BA\star$ with a common value v. Then, according to the third property of GC, their output for the protocol GC would be $(v, 2)$. Thus, all the honest players initiate the protocol $BBA\star$ with $b_i = 0$, and hence based on Theorem 1, their output for $BBA\star$ and $BA\star$ would be 0 and v, respectively. This shows that $BA\star$ provides consistency. According to Theorem 1, we know that at the end of step 2, honest players i and j with $i \neq j$ compute the same value $out_i = out_j = out$. If $out = 1$, then both player i and player j output $\perp$ at the end of the protocol meaning that they are in agreement. Otherwise, party i and party j, respectively, output v_i' and v_j' that they computed at the end of step 1. Since $out = 0$, according to Theorem 1, there is at least one honest party k whose input to $BBA\star$ b_k equals 0. It means that g_k for this honest party equals 2. Following the first property of GC, it is impossible that $g_i = 0$ or $g_j = 0$. Thus, since $g_i, g_j > 0$ is satisfied, according to the second

property of GC, we have $v'_i = v'_j$, and hence $out_i = out_j$. This shows that $BA\star$ also provides agreement. □

Under strong synchrony, $BA\star$ reaches consensus in four steps (two steps for GC and two steps for $BBA\star$) if the proposer of the highest-priority block is honest. If the highest-priority block proposer is dishonest, $BA\star$ is expected to reach consensus in 13 steps (2 steps for GC and 11 steps for $BBA\star$) [17].

Player Replaceability We know that $BA\star$ is performed in several steps and due to scalability issues it is executed by a limited number of users. Also, we assumed that the adversary can make any party malicious at any time. Thus, if committee members are fixed during a round, the adversary would be able to identify the committee members during the first step of that round and then control all of them through the next steps. In other words, although the adversary is not capable of corrupting $\frac{1}{3}$ of all the users, she can make all the committee members malicious. To prevent this from happening, Algorand requires execution of the cryptographic sortition per step. It can be proven that $BA\star$ can also reach an agreement under such a condition [9]. Furthermore, only the party himself can determine if he is a committee member for a particular upcoming round step, and hence the adversary cannot predict the users she must manipulate in next steps of the round.

6 Deployment

In this section, we discuss some issues regarding to Algorand deployment [17].

6.1 Block and Transaction Format

Each Algorand block consists of the following data:

- The list of transactions
- The round number
- The seed
- The previous block's hash
- The timestamp

After the block proposal phase, once a user observes the highest-priority block, it uses the mentioned data inside the proposed block to verify its validity. To do so, the user checks if

- All transactions are valid (e.g., transactions do not lead to any account with negative balance and signatures are valid).
- The round number is correct.
- The seed is correct.

– The previous block's hash is correct.
– The timestamp is correct.

If the highest-priority block does not meet all the mentioned criteria, the user uses an empty block as the input to the agreement protocol.

The following six transaction types exist in the Algorand system:

– Payment
– Key registration
– Asset configuration
– Asset freeze
– Asset transfer
– Application call

However, fields and values in a transaction type might be set in such a way that we have more granular transaction types. For example, a regular payment transaction from one account to another account contains the following compulsory fields:

– "amt": The total amount to be sent to the receiver account
– "fee": The fee paid by the sender
– "fv": The first round for which the transaction is valid
– "gh": The hash of the genesis block of the network for which the transaction is valid
– "lv": The last round for which the transaction is valid
– "rcv": The address of the account that the amount is sent to
– "snd": The address of the account that sends the amount
– "type": The transaction type which is "pay" for a payment transaction

but there is an optional field, called "close," which when set it means that the sender account specified in "snd" should be closed and its remaining funds, after paying the fee and amount, must be transferred to the address specified by "close." Although such a transaction is of type "Payment," it is used to close an account. More details about other transaction types can be found in [3].

6.2 Bootstrapping

Before bootstrapping the system, there is an initial set of participants with known weights and public keys. To bootstrap the system, an initial seed is required. As stated earlier, the distributed random number generation algorithm [8] is used to create the initial seed.

There is a certificate corresponding to each Algorand round, where the certificate contains at least $2t + 1$ votes from the last step of the protocol $BA\star$ for that round. When users want to join the system, they start from the genesis block and verify validity of the corresponding certificates in order. Since only the certificate of the last step is verified, the adversary might find a step number for which all the committee members are controlled by the adversary. It can be shown that [17] if

the committee is large enough (i.e., it contains at least 1000 members), then the probability of finding such a step number is negligible, and hence the attack is impractical. Users who are synchronized with Algorand ledger are not required to verify the certificates.

Furthermore, a dedicated study [20] has been done on Algorand bootstrapping that suggests some techniques to avoid the storage and communication bandwidth requirements of other blockchain protocols. These techniques and their corresponding advantages are [19, 20] the following:

- Transaction expiration leading to reduction in the storage cost of the Algorand blockchain
- Sharding to distribute state across participants without weakening security guarantees
- Stamping certificate to allow users to join the network without checking every single block from the genesis block

6.3 Communication

Since the sortition algorithm is noninteractive and is privately run by each Algorand node, it seems that it is impossible for the adversary to identify block proposers before they broadcast their proposed block throughout the network. Otherwise, the adversary would be able to mount DoS attack against block proposers. However, contents of a block do not fit into one packet. Thus, theoretically, an adversary would be able to identify block proposers before they complete sending their block within a TCP connection and then mount DoS attack against them. To make the attack feasible, the adversary must react within a few seconds. It seems that this attack can be neglected as such a fast attacker can disrupt any peer-to-peer network including Bitcoin network. Communication over Tor network [14] is an approach that can be adopted to make the attack even less negligible as it makes it more difficult for the adversary to mount DoS attack. The adversary might also disturb the network through sending invalid messages over the network. To mitigate this attack, Algorand nodes check validity of the messages that they are supposed to relay.

6.4 Scalability

Users relay messages they receive from committee members or block proposers of each step of the agreement protocol. Therefore, as long as the expected committee size or the number of block proposers is fixed, increase in the number of users does not increase the volume of data that each user must relay. Increase in the number of users only affects the time that is required for a message to be broadcast over

the network. It can be shown that the growth in broadcast time is logarithmic in the number of users [26]. Furthermore, since block proposers change from one round to another, it is not an important matter if a small fraction of users get temporarily disconnected from the rest of the network.

Furthermore, [17] reports results of the Algorand simulation to show that Algorand scales well with the number of users. They have deployed a prototype of Algorand in C++ on 1000 large m4.2xlarge virtual machines, each with 8 cores, up to 1 Gbps network bandwidth and 50 users where users propose 1 MByte blocks. In their latency model, virtual machines are assigned to 20 different major cities across the world where latency between virtual machines in the same city is modeled to be negligible. The results show that by varying the number of running virtual machines from 100 to 1000, the latency is near constant (around 20 seconds). Even by varying the number of users on 1000 virtual machines from 50 to 500, the latency is still almost flat showing that Algorand scales well. Their results for 50,000 users (1000 virtual machines, each with 50 users) show that by increasing the block size to 10 MByte, each transaction is still confirmed within a minute and 750 MBytes of transactions are committed per hour which is 125 times the Bitcoin throughput.

7 Security Discussion

In this section, we discuss how Algorand might be resist against different attacks.

7.1 Nothing-at-Stake Attacks

If there is a fork in a PoS-based blockchain, it is financially beneficial to consensus participants to participate in all existing chains. The reason is that (1) unlike PoW-based blockchains where mining on more than one chain negatively impacts the chance of the miner to mine, the consensus participant in a PoS system has nothing to risk and (2) consensus participant will be rewarded no matter which fork will eventually become part of the ledger. Nothing-at-stake [21, 28] attack can be avoided by penalizing consensus participants who cause it, i.e., a portion or all of their stake is forfeited. In Algorand, forks occur with negligible probability, and hence nothing-at-stake attacks are practically avoided.

7.2 Grinding Attacks

In PoS-based blockchains, there is an election mechanism that elects a leader for each round of the system. Grinding attacks [16] target the election mechanism. In other words, the adversary exploits a vulnerability in the election mechanism in

order to be elected as the leader more often than expected. For Algorand, the seed that is used in the sortition algorithm of round r is $seed_{r-1-(r \pmod R)}$. Therefore, if private keys that are used in the sortition algorithm of round r can be updated after the time when $seed_{r-1-(r \pmod R)}$ becomes known, the adversary u might try different values of sk_u to find a value that gives her a higher chance to be selected as the block proposer for round r. To prevent this attack, sk_u must be selected by u prior to $seed_{r-1-(r \pmod R)}$ being known. However, as we will see, this requirement does not suffice under the weak synchrony assumption.

Under weak synchrony assumption, the adversary can control the network for a long period of time. Therefore, the adversary can control the network to force users to agree on empty blocks. Then, the seed for round r is computed as $H(seed_{r-1} \| r)$. It makes the adversary able to continue manipulating the network, guess the seed for next rounds, and select her private key accordingly. Based on the definition of the weak synchrony assumption, in every time period of length b (of order of several days), there is a time interval of length s (of order of several days) when the network is strongly synchronous. Furthermore, it is proved that under strong synchrony, the probability that the attacker can compute the next k selection seeds decreases exponentially with k [17]. Therefore, since s is large enough, grinding attack is practically avoided in Algorand system given that cryptographic sortition of round r uses users' private keys and their corresponding weight from the last block that was created b-time before block $r - 1 - (r \pmod R)$. This ensures that we have at least a large enough strongly synchronous time period between every two weight updates.

7.3 Long-Range Attacks

In this family of attacks [7, 13], the adversary rewrites the history of the blockchain by forking it from a significantly distant point in the past (even from the genesis block). PoS blockchains are potentially vulnerable against long-range attacks. The reason is that unlike PoW blockchains, where the adversary must have the majority of the network's hashrate to be able to create a fork with a desired number of blocks, in PoS systems, the adversary can create blocks at no significant cost.

Although resistance of Algorand against long-range attacks is not formally proved, due to the following reasons, it seems it is resistant against these attacks:

- In all categories of long-range attacks, the adversary creates a hidden fork, mines on her fork to make it longer than the main chain, and finally publishes it. However, in Algorand, a sufficient fraction of stakeholders must certify the output block of each round. Therefore, given that at least $\frac{2}{3}$ of the total currency is held by honest users, it seems impossible for the adversary to have her blocks certified.
- In "posterior corruption" attacks, the adversary uses the private keys from retired users to sign new blocks for her fork and in this way increases her chances of outpacing the main chain. In Algorand, fork occurs with negligible probability.

However, assuming that the adversary is the leader of round r, she creates a block for the main chain. Then, since she has identified the committee members of round r, she can corrupt them to certify a second block for her hidden fork. To avoid this attack, each honest committee member u for step s of round r uses an ephemeral key pair to sign the message where the public key is specific to the user u for step s and round r. He also destroys the private key once the signature is created. Therefore, it is impossible for the adversary to force a committee member to sign another message because the required private key does not exist anymore. Moreover, it is impossible for the adversary to create another ephemeral key pair for the committee member u for step s and round r such that it is accepted by other honest users.

8 Use Cases

This section briefly introduces different use cases of the Algorand system.

Payment The native token of Algorand is called $ALGO$ which is used as currency. The most obvious use case of Algorand system is transferring $ALGO$s from one user account to another. To do so, the account holder can create a transaction and send some funds from her account to another user's address, sign it, and transmit it in the network. This valid pending transaction might be put in a block by the next round block proposer and added to the blockchain.

Asset Algorand users can create on-chain assets with the same security and functionalities of $ALGO$. Such assets are called *Algorand Standard Assets* (ASA). Algorand assets can represent real-world assets including the following:

- Fungible tokens such as gaming points, stablecoins and loyalty points, system credits and cryptocurrencies, securities, and government-issued fiat
- Non-fungible tokens (NFT) such as gaming items, supply chain, real estate, identity, certification, collectable items (e.g., tickets for an event or votes), patents, and regulatory certifications

Assets might be transferred among users like the Algorand native token. Assets are configurable meaning that they can be minted, revoked, or frizzed. A recently deployed asset is COVID vaccine passports, outcome of a joint project between Algorand and Koibanx, introducing blockchain-based vaccine passport in Colombia [2]. The digital passport which is called VitalPass benefits from features of Algorand blockchain (i.e., security and transparency) to track COVID vaccination.

Another form of assets is stablecoin which can be used for real-time payments. Recently, a fintech platform called "Six Clovers" used Algorand blockchain as its infrastructure to provide a payment services to merchants and banks to transact in

digital currency [2]. It plays a role similar to SWIFT[3] but with faster speed, lower fee, and higher transparency. Moreover, in early 2021, a worldwide major copyright management company, called SIAE, launched 4 million NFTs on Algorand blockchain for over 100,000 authors to digitize their rights for their works [2]. The mentioned applications are just some recently deployed samples of endless list of ASA use cases.

Atomic transfer In the real world, if two or more parties want to trade their assets, they usually use a third party to guarantee the exchange. Usage of the third party comes at the cost of higher time and fee particularly for international trades. Algorand uses a method called *atomic transfer* to ease such exchanges. Atomic transfer makes it possible for Algorand users to group up to 16 transactions together and send them on the network. Algorand system guarantees that all transactions either are completed or all fail. Atomic transfer can be completed in a few seconds with negligible fee.

Stateless smart contracts Assume that you want to pay some funds to one or more payees if some conditions are met. Then, for such situations, Algorand stateless smart contracts could be used. Let's describe the concept using a few examples. Assume that Alice wants to borrow some funds from Bob, and Bob is satisfied to lend her under the condition that she locks some assets as the collateral in a particular account called the *contract account*. Assets in the contract account can be released by Alice if she pays back the loan before a predetermined deadline or by Bob otherwise. If these conditions are not met, the funds in the contract account cannot be spent. As an another example, assume that Alice wants to allow her utility company to withdraw up to 200 $ALGO$s every 1000 rounds. To do so, she creates a contract account with the mentioned logic. Such smart contracts are called stateless, because they only rely on data within the transaction itself.

Stateful smart contracts Contrary to stateless smart contracts, stateful smart contracts use some values on the blockchain as part of the contract state. These values can be stored locally or globally. Let's explain the concept using an example. Assume that we want to have a voting application on the Algorand blockchain. The smart contract corresponding with the voting application is on-chain where it contains the list of candidates and number of votes for each one. These values are stored globally in the stateful smart contract. Each vote is actually a transaction based on which the smart contract increment the vote count for the candidate and mark the voter to prevent him from double voting. To mark the voter, a value is stored in the user's local state. Actually, each user who participates in the voting process accepts that its local account is marked by the stateful smart contract. As an another example, a donation application can maintain the total donations and each individual account donation in its global state and each participant's state, respectively. Each user's donation value can be referred to for tax purposes.

[3] Society for Worldwide Interbank Financial Telecommunication

It is also possible to combine the mentioned Algorand functionalities to solve more sophisticated problems. For instance, to have a permissioned voting application, we can have the previously mentioned stateful smart contract where authorization to vote is granted to the authorized users through assets (voting tokens) which are exclusively created for this voting application. In order to vote, the voter must spend this granted token.

9 Conclusion

Algorand is a new cryptocurrency that combines a cryptographic sortition algorithm with a Byzantine agreement protocol, called $BA\star$, to (1) achieve low computation amount, (2) remove the possibility of forks, and (3) provide high throughput. In this chapter, we fully covered the cryptographic sortition and $BA\star$ protocol. We also discussed security of the Algorand blockchain and explained its different use cases. According to the importance of the Algorand blockchain and its wide range of applications, it is worthwhile to perform some practical experiments with realistic number of users to obtain maximum achievable throughput of Algorand on its main blockchain.

References

1. Hashrate Distribution. Available at https://www.blockchain.com/pools
2. Algorand. Blockchain use cases in action: 5 examples of businesses building on algorand. Available at https://www.algorand.com/ecosystem/use-cases/siae
3. Algorand. Transaction reference. Available at https://developer.algorand.org/docs/get-details/transactions/transactions/
4. C. Badertscher, P. Gaži, A. Kiayias, A. Russell, V. Zikas, Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability, in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018), pp. 913–930
5. BitcoinWiki. Proof of stake. Available at https://en.bitcoin.it/wiki/Proof_of_Stake
6. E. Buchman, Tendermint: Byzantine fault tolerance in the age of blockchains. Ph.D. thesis, 2016
7. V. Buterin, Long-range attacks: the serious problem with adaptive proof of work. Ethereum Blog, May, 2014
8. C. Cachin, K. Kursawe, F. Petzold, V. Shoup, Secure and efficient asynchronous broadcast protocols, in *Annual International Cryptology Conference* (Springer, 2001), pp. 524–541
9. J. Chen, S. Micali, Algorand (2016) arXiv preprint arXiv:1607.01341
10. K. Croman, C. Decker, I. Eyal, A.E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E.G. Sirer et al., On scaling decentralized blockchains, in *International Conference on Financial Cryptography and Data Security* (Springer, 2016), pp. 106–125
11. B. David, P. Gaži, A. Kiayias, A. Russell, Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2018), pp. 66–98
12. C. Decker, R. Wattenhofer, Information propagation in the bitcoin network, in *IEEE P2P 2013 Proceedings* (IEEE, 2013), pp. 1–10

13. E. Deirmentzoglou, G. Papakyriakopoulos, C. Patsakis, A survey on long-range attacks for proof of stake protocols. IEEE Access **7**, 28712–28725 (2019)
14. R. Dingledine, N. Mathewson, P. Syverson, Tor: the second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004
15. J.R. Douceur, The Sybil attack, in *International workshop on peer-to-peer systems* (Springer, 2002), pp. 251–260
16. Ethereum. Proof-of-stake faq – Ethereum. 2017. Available at https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ
17. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: scaling Byzantine agreements for cryptocurrencies, in *Proceedings of the 26th Symposium on Operating Systems Principles* (2017), pp. 51–68
18. A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: a provably secure proof-of-stake blockchain protocol, in *Annual International Cryptology Conference* (Springer, 2017), pp. 357–388
19. D. Leung, Algorand's vault: fast bootstrapping for the algorand blockchain (2019). Available at https://www.algorand.com/resources/algorand-announcements/algorands-vault-fast-bootstrapping-for-the-algorand-cryptocurrency
20. D. Leung, A. Suhl, Y. Gilad, N. Zeldovich, Vault: fast bootstrapping for the algorand cryptocurrency (2018). Cryptology ePrint Archive
21. W. Li, S. Andreina, J.M. Bohli, G. Karame, Securing proof-of-stake blockchain protocols, in *Data Privacy Management, Cryptocurrencies and Blockchain Technology. DPM CBT 2017*, ed. by J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, J. Herrera-Joancomartí. Lecture Notes in Computer Science, vol. 10436 (Springer, Cham, 2017). https://doi.org/10.1007/978-3-319-67816-0_17
22. S. Micali, Byzantine agreement, made trivial (2018). Available at https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Distributed%20Computation/BYZANTYNE%20AGREEMENT%20MADE%20TRIVIAL.pdf
23. S. Micali, M. Rabin, S. Vadhan, Verifiable random functions, in *40th Annual Symposium on Foundations of Computer Science (cat. No. 99CB37039)* (IEEE, 1999), pp. 120–130
24. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system. Decentralized Bus. Rev. 21260 (2008)
25. S. Noether, Ring signature confidential transactions for monero. IACR Cryptology ePrint Archieve. Report 2015/1098 (2015)
26. O. Riordan, N. Wormald, The diameter of sparse random graphs. Comb. Probab. Comput. **19**(5–6), 835–926 (2010)
27. C. Stoll, L. Klaaßen, U. Gallersdörfer, The carbon footprint of bitcoin. Joule **3**(7), 1647–1661 (2019)
28. E. Wiki, Problems (2017). Available at https://github.com/ethereum/wiki/wiki/Problems
29. G. Wood et al., Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**(2014), 1–32 (2014)
30. Y. Xiao, N. Zhang, W. Lou, Y.T. Hou, A survey of distributed consensus protocols for blockchain networks. IEEE Commun. Surv. Tutorials **22**(2), 1432–1465 (2020)

Tendermint

Yackolley Amoussou-Guenou, Antonella Del Pozzo, and Sara Tucci-Piergiovanni

1 Introduction

In a strict sense, the blockchain is nothing else than the distributed data structure maintained and used by Bitcoin, the first successful peer-to-peer electronic cash system. Bitcoin's white paper [37] introduced in 2008 a very elegant protocol to build and maintain the blockchain using a popular technique called *proof-of-work* [24] to decide who will append a new block in the chain. The block that will be appended to the blockchain is created by the process, also called miner, having enough computing power to solve a cryptographic puzzle first. The best-known way to solve this puzzle is by repeated trials.

The major criticism for the *proof-of-work* approach is the huge energy waste that comes from its use [21]. Since 2008, therefore, many alternatives to the proof-of-work, called *proof-of-**, have been explored to replace the energy consuming mining process due to the proof-of-work. One can cite *proof-of-activity* [7]; *proof-of-burn* [41] – in which processes need to send away ("burn") some of their virtual coins to get the right to add blocks, the more coins burned, the higher the chances to be picked; *proof-of-elapsed time* [18]; *proof-of-space* [25]; and finally *proof-of-stake* [28], where a process with higher stakes in the system has more chances to be picked.

All these alternatives, although reducing the energy consumption problem, did not prove clear security guarantees with respect to typical attacks such as double spending and selfish mining attacks, because they lack a way of controlling *forks*: the temporal coexistence of multiple chains used to realize those attacks. This is maybe the reason why Ethereum [13], the first general-purpose public blockchain

Y. Amoussou-Guenou (✉) · A. Del Pozzo · S. Tucci-Piergiovanni
Université Paris-Saclay, CEA, Palaiseau, France
e-mail: yackolley.amoussou-guenou@cea.fr; antonella.delpozzo@cea.fr; sara.tucci@cea.fr

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_7

whose whitepaper appeared in 2013, still uses the proof-of-work mechanism to build the blockchain.

Interestingly then from 2008 to 2013, blockchain platforms explored alternative consensus mechanisms without much success. The advent of Ethereum in 2013, which referred to state machine replication to execute general-purpose computer programs or smart contracts, attracted however the interest of the distributed computing community. Distributed computing has been concerned with state machine replication and the underlying consensus mechanisms for 40 years, by defining the concept of *consensus* in a precise computational sense and providing a myriad of proven consensus algorithms. Unfortunately, classical consensus algorithms seemed not to apply in the blockchain context because it is defined for closed systems only, where the number of processes is known in advance. Nonetheless, Tendermint's white paper in 2014 [32] proposed for the first time the use of classical consensus, in the distributed computing sense, in the context of blockchains. The Tendermint approach paved the way for an entire class of blockchains, (Algorand [26], HotStuff [50], RedBelly [19], the current version of Tendermint [12], etc.) in which instead of drawing one block creator at random, it is an entire committee that is selected (through proof-of-* techniques) to run variants of classical Byzantine fault-tolerant (BFT) consensus algorithms (e.g., [16, 23, 34, 36, 49]) to produce the block to append next. In these *committee-based blockchains*, forks are never created, as long as less than two thirds of the committee members are malicious or Byzantine. Under these hypotheses, an honest majority of committee members signs the created block, which is then valid by construction. Other participants need only to adopt such block and add it to their local blockchain. The major advantage of these approaches resides in what is called *instant finality*: as soon as a valid block is produced by a committee, all the included transactions are finalized – they cannot never be revoked. Interestingly, instant finality improves the confirmation transaction time to thousands of transactions per second – where Bitcoin can only confirm a dozen of transactions per second.

The number and prototypes of committee-based blockchains are increasing since 2014 as can be seen in the nonexhaustive list in Fig. 1.

Besides offering instant finality, these committee-based blockchains are interesting because they offer proven security guarantees thanks to their algorithms rooted in distributed computing theory. Finally, this class of blockchains is not based on any energy-consuming process, which qualifies them as a class of blockchains suitable to be used whenever ecological concerns come into play.

As an eminent representative of the committee-based blockchain class, we dedicate this chapter to the Tendermint blockchain, introducing briefly its architecture, the current projects based upon and an extensive and detailed description of the consensus algorithm at its core. For the consensus algorithm, we take a pedagogical approach illustrating first a version of the algorithm that suffered from a subtle liveness vulnerability and then the corrected version. The identification of vulnerabilities and the correctness of these algorithms are real issues nowadays, and we hope to convince the reader throughout this chapter.

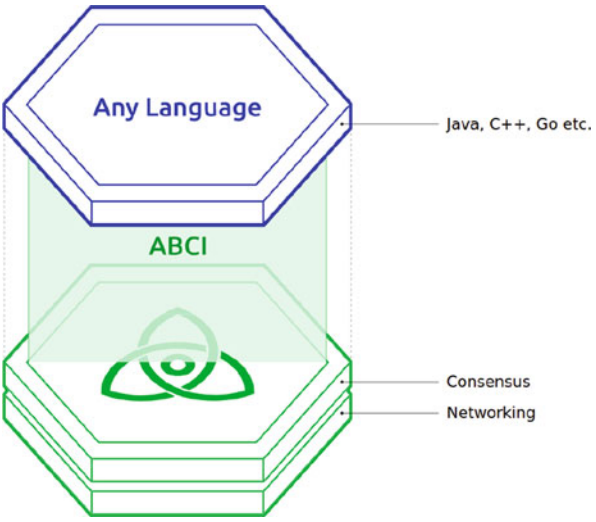


Fig. 1 Increasing number of committee-based blockchains (as of 2020)

2 Tendermint Blockchain Platform

This section focuses on the Tendermint blockchain platform, one of the first blockchain platforms based on Byzantine fault-tolerant (BFT) consensus. We first present its principal characteristics and then an overview on the Tendermint architecture (which consists of different modules). Let us recall that a blockchain is a distributed chain of valid blocks. The first block, the *genesis* block, is predetermined. Each block points back to the parent block (the pointer is the hash of the parent block) and contains a bunch of transactions. The blockchain is maintained by the system participants, also called peers, servers, or processes. Transactions are application-dependent pieces of data sent by the blockchain users, also called clients. In the Bitcoin case, transactions are transfer of cryptomoney assets, while in the case of Ethereum, transactions can be also the creation of smart contracts and the interaction with them. In the Tendermint case, transactions are unspecified, indeed, contrarily to the previous two examples. Tendermint has not a monolithic design. It decouples the blockchain data structure implementation and the particular blockchain application. Finally, we said that the blockchain is a chain of valid blocks. In proof-of-work-based blockchains, as Bitcoin and Ethereum 1.0, a block is valid if (i) its content is valid and (ii) it contains the solution for the proof-of-work. In Tendermint, condition (i) is application-dependent, and condition (ii) is that such block has to be agreed upon by a super majority of peers, and such agreement is carried out thanks to the consensus engine. Let us remark that we are comparing Tendermint with Bitcoin and Ethereum from a flexibility point of view and not on the way new valid blocks are agreed upon. Indeed, proof-of-work-based solutions do not require any knowledge on the number of the system participants and are partition-tolerant. Contrarily, Tendermint relies on the knowledge of the number

and identities of processes participating to the consensus algorithm. Tendermint, being based on a BFT consensus algorithm, inherits its properties of safety and consistency: safety because it can tolerate up to $1/3$ of peers arbitrarily failing and consistency because all the peers see the same transaction log and computes the same states. Notice that in Tendermint not all the peers are involved in the consensus execution. We refer to the validators set as the subset of peers in charge to run the consensus engine. Such set is initially predetermined in the genesis block, but depending on the application logic, it can vary block after block.

Tendermint is composed of two major modules: (i) a blockchain consensus engine called Tendermint core and (ii) a generic application interface called the Application BlockChain Interface (ABCI). The ABCI interface enables the transactions to be processed in any programming language. It is possible to use Tendermint to implement BFT state machine replication for applications written in any programming language. In other words, a state machine is a machine that updates its current state every time it reads an input (or receives it). A replicated state machine is a state machine replicated over different peers that update their state consistently processing the same sequence of inputs.[1] State machine replication and blockchain share the same paradigm, in the sense that their state is dictated by the sequence of transactions. Other two relevant components in Tendermint core are the *p2p* and the *mempool* modules. In particular, p2p manages the peer discovery, the connection, and the communications. The mempool manages the list of potentially valid transactions. Those transactions are relayed to other peers and are potentially placed in a block proposition for the consensus engine, which is presented in the further sections. Notice that the following operations are left to the application specification: (i) the validation of each transaction received against the current state, (ii) the validation of the cryptographic credentials of the transaction and once the transaction is validated, and (iii) the update of the application state. Indeed, Tendermint core needs to explicitly demand to the application level the validation of transaction and the state change. This is done with the ABCI, detailed in the following:

ABCI provides an interface between the application logic (any state machine replication application expressed in any language) and Tendermint core, which is mainly composed of the BFT consensus and networking layers (cf. Fig. 2). The interface consists of a set of methods; each method has associated with a request and response message type. In other words, methods are called by sending the request message, and the return is received with the response message (cf. Fig. 3).

Those methods are split across four separate ABCI connections:

- The consensus connection, driven by the consensus protocol and responsible for the block execution: *InitChain*, *BeginBlock*, *DeliverTx*, *EndBlock*, *Commit*. In particular:

[1] Notice that as necessary condition those machines have to be deterministic, such that given the same sequence of inputs they go through the same sequence of states.

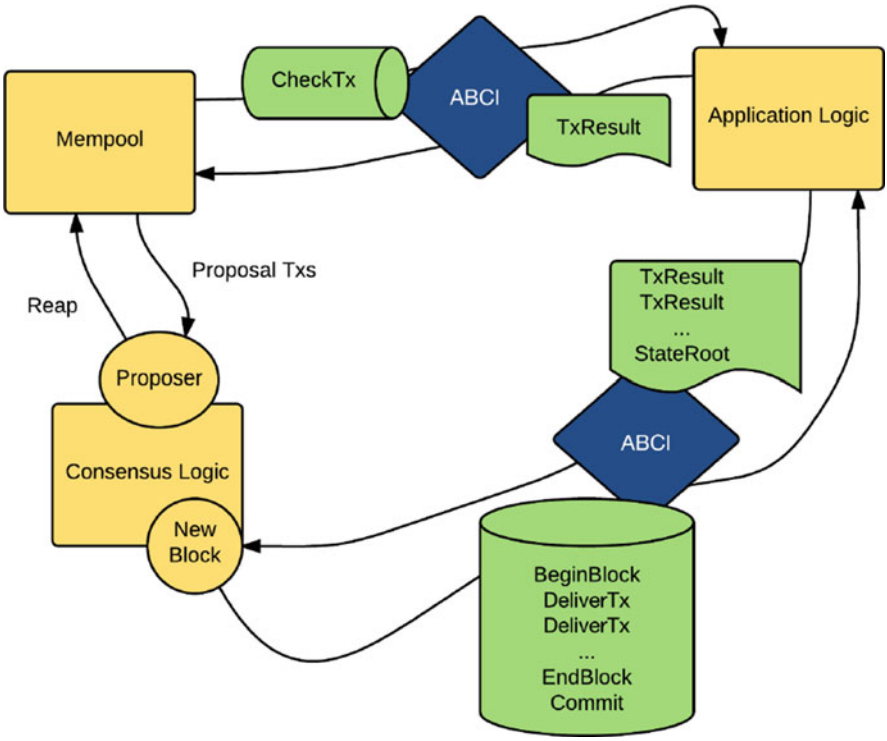


Fig. 2 High-level view of the architecture (https://cosmos.network/intro)

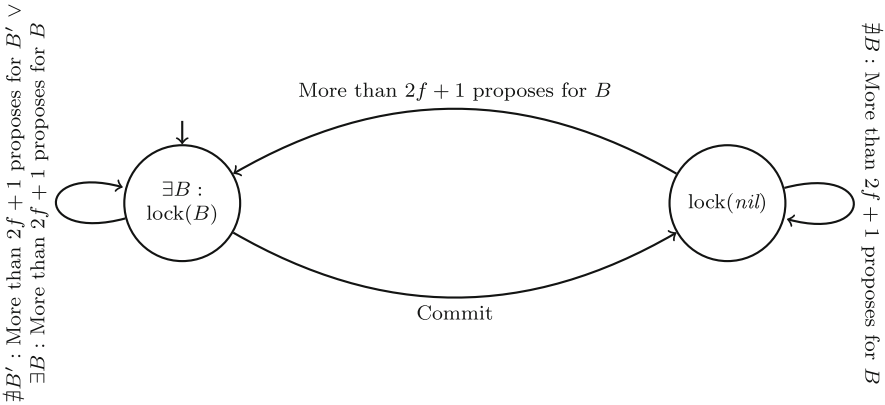


Fig. 3 Relevant ABCI connections with the application logic and the consensus logic (https://docs.tendermint.com/master/introduction/what-is-tendermint.html)

  - The *InitChain* method is called by the Tendermint core when the chain is first started, i.e., at the beginning of the blockchain (the genesis block). Taking into account the information in the genesis block, it initializes the different parameters related to block executions, for instance, block size, the consensus parameters, the validators for the next block, etc.
  - During the consensus execution, validators call the ABCI methods *BeginBlock*, *DeliverTx* (for each transaction in the block), and *EndBlock* in order to validate the transactions carried in a block proposal (more details in the next section). The same is done by peers, not validators, when receiving a new decided block. In particular:
    - *BeginBlock*: signals the beginning of a new block. Called prior to any DeliverTxs
    - *DeliverTx*: to fully execute the carried transaction and returns information about the validity of such transaction with respect to the application state and the previous delivery transactions in that block
    - *EndBlock*: signals the end of a block and can be associated with the logic to manage the validator set update
  - *Commit* finally: *Commit* is called to finalize the volatile state produced during the previous *DeliverTx* calls into the application's internal state. The application returns the Merkle root hash of its new state.

- The mempool connection, for validation of new transactions, before they are relayed to other peers or included in a block – *CheckTx*:
  - When a Tendermint core peer receives a transaction tx from some client, it sends a *CheckTx* message to the application layer to check for the validity of the latter. If the response is positive, then tx is added to the peer's mempool and is relayed to other peers. Notice that the validity of transaction is application-dependent.
- The info connection, for initialization and for queries from the user: *Info* and *Query*.
- The snapshot connection, for serving and restoring state sync snapshots. Used, for instance, when a new peer joins the system to fetch and apply a snapshot of the state machine: *ListSnapshots*, *LoadSnapshotChunk*, *OfferSnapshot*, and *ApplySnapshotChunk*.

The complete details of the interactions between the Tendermint core and the ABCI can be found on the Tendermint's website [43–46].

2.1 Blockchain Projects Using Tendermint

Tendermint is now used as the blockchain platform of many applications. These applications are often built using the Cosmos SDK [42]. The Cosmos SDK is a

modular framework developed by the Tendermint foundation providing a common ground for developing different kinds of applications, possible in most of the programming languages; at the core of Cosmos SDK, there is the Tendermint consensus engine.

On their website, Tendermint/Cosmos presents few different use cases where the framework can be applied: Open Finance, Games, Video Streaming, Healthcare, Prediction Markets, Cross-Border Payments, etc.

Currently in the ecosystem, one can count more than 100 projects, among them we can mention:

- Binance Chain [9]. This platform is developed by Binance, one of the largest companies offering trading opportunity on crytpocurrency exchanges. Binance chain is a platform providing users means to build their own decentralized applications and digital assets on one blockchain while taking advantage of the trading performance offered on the Binance decentralized exchange (DEX).
- Agoric [2]. Agoric is a platform for building distributed applications to establish new markets, therefore without a centralized control. They have a focus on decentralized finance.
- Terra [47]. Terra offers to its users a platform for making cross-border exchanges by means of using stablecoins (cryptocurrencies that overall have a fixed rate with "classical" currencies like Dollar or Euro).
- ISISnet [27]. ISISnet provides the platform for the Blockchain-based Service Network [10], which is a cross-cloud, cross-portal, and cross-framework global public infrastructure network used to deploy and operate on different types of blockchain distributed applications.
- Aragon [35]. The goal of Aragon is to have a platform that helps creating tools for decentralized organizations. Among other applications, they provide the Aragon Network [6] which is a digital financial jurisdiction.
- IOV [40]. IOV is a name service application that provides in a distributed fashion a username that is linked to a list of blockchain (or crypto) addresses.
- BigchainDB [8, 38]. BigchainDB is a blockchain-based database that has the characteristics of decentralization and immutability and the ability to treat anything stored in the database as an asset.

3 Tendermint Blockchain Agreement Procedure

In this section, we dive into the Tendermint core consensus algorithm. Before that, we first detail the system model, i.e., the set of assumptions that abstract the system: the processes, the kind of failures that can occur, and the communication primitive and its semantic. Alongside, we define the consensus problem as a conjunction of liveness and safety properties. Finally, we present the Tendermint solution, which solves consensus in the specified system model. In particular, for the pedagogical purpose, we give two versions of Tendermint: the first implemented algorithm

containing a design – very subtle – bug-hindering liveness, and the current and corrected algorithm where the bug has been solved.

System Model
The system is composed of a finite set Π of n processes, namely, $\Pi = \{p_1, \ldots\ p_n\}$. We consider that up to f processes can be Byzantine faulty [39]. By Byzantine, we mean a process that behaves arbitrarily. This model allows us to encompass any behavior and thus also malicious attackers. To be able to solve the consensus problem, we assume that $f < n/3$ and more precisely we consider $n = 3f + 1$.

Processes communicate with each other by exchanging messages through an eventually synchronous network [23]. *Eventually Synchronous* means that after a finite unknown time $\tau > 0$ there is an unknown bound δ on the message transfer delay. When $\tau = 0$, the network is *synchronous*.

In the following, we assume the presence of a *best effort broadcast primitive*, broadcast(), which means that when a correct process broadcasts a value, eventually all the correct processes deliver it. To simplify the presentation, it is assumed that a process can send messages to itself. A process p_i receives a message by executing the primitive delivery(). Messages are created with a digital signature, and we assume that digital signatures cannot be forged. When a process p_i delivers a message, it knows the process p_j that created the message.

Consensus
Agreeing in a distributed fashion among a set of processes has been intensively studied in distributed systems with the abstraction of the distributed consensus or simply consensus defined as follows:

Definition 1 (Consensus) An algorithm implements the consensus in the presence of Byzantine processes if and only if it satisfies the following properties:

- *Termination.* Every correct process decides on a value (a block).
- *Agreement.* If two correct processes decide, respectively, on values B and B', then $B = B'$.
- *Validity.* A decided value by any correct process is valid; it satisfies a predefined predicate.

The validity property considered has been introduced by Cachin et al. in [14]. It is important to note that the validity predicate must be known by all processes and depends on the application. External validity was later adapted by Crain et al. in [19] as a well-suited validity concept for blockchains.

In the rest of this section, we discuss a particular Consensus solution designed to work in the Blockchain setting, the Tendermint core algorithm. In the following, we refer to it as Tendermint. Interestingly, in the blockchain realm, already exist Consensus based blockchain proposals (e.g., [1, 5, 20, 22], and [30]). As a common point, the majority of those solutions share *Practical Byzantine Fault Tolerant* (PBFT)[17] as consensus algorithm. PBFT is a leader-based protocol that made the consensus solution practical, contrarily to the DLS solution [23], the so far only solution for Consensus in an eventually synchronous system but having a message

complexity of $O(n^4)$. In the PBFT case, if the leader is correct, the complexity boils down to $O(n^2)$, while if the leader is suspected to be faulty, then a view change mechanism takes place to change the leader and resume the computation. The view-change is used to avoid that, in case of faulty leader, if some correct process decides on a value v, the other correct processes cannot decide on a value $v' \neq v$ when the new leader proposes a new value. In the case of a view change, all processes exchange their view of the current system state to proceed with the new leader (view-change) safely. Without getting into the details, this operation has a message complexity of $O(n^3)$. The protocol terminates when there is a correct leader, which may happen for the first time in epoch $f + 1$, then in the worst-case scenario, PBFT has a message complexity of $O(n^4)$. In the following, we will see that Tendermint, leader-based as PBFT, uses the locking mechanism, an alternative mechanism to preserve safety when the leader changes that drops the overall message complexity to $O(n^3)$. Intuitively, in this case, processes instead of exchanging all the messages they delivered (as with view change), they locally keep track of potentially decided values (locking on a value) and use this information to manage further proposed value. Further details on the Tendermint's complexity are discussed on page 216.

3.1 Tendermint Intra-committee Consensus Algorithm

We note that we focus on a single committee at a time. Tendermint follows the rotating coordinator paradigm, i.e., for each new block to be appended there is a proposer, chosen among the committee members that proposes the block.If the block is not decided, then a new proposer is chosen (in a round-robin fashion) and so on until a block is decided by correct committee members. In the following, we present two versions of Tendermint: first, the initial version containing a bug that is deeply discussed in [3], and the latter (currently implemented) version resolving the bug of the first version, discussed in [4]. Our presentation focuses on the important and key aspects of the algorithm; for clarity, we did not include some optimizations (such as skipping some rounds if sufficiently many processes are in a higher round, etc.).

Message Syntax
A committee member i broadcasts a message $\langle TAG, h, e, m\rangle$, where m contains a value v; we say that i pre-proposes, proposes, or votes v if TAG=`PRE − PROPOSE`, TAG=`PROPOSE`, and TAG=`VOTE`, respectively. h represents the height or the consensus instance, and e the *epoch* intuitively represents the number of rounds for that height since the beginning of the consensus instance.

Basic Principles of the Protocol
Recall that each block in the blockchain is characterized by its height h, which is the distance in terms of blocks from the genesis block, which is at height 0. For each new height, the two algorithms share a common algorithmic structure, they proceed in *epoch*s, and each epoch e consists in three rounds: the *pre-propose* round, the

Algorithm 1 Message management for process i

```
1: upon reception of ⟨TYPE, h, e, message⟩ from process j do
2:   if ∄c : (⟨TYPE, h, e, c⟩, j) ∈ messagesSet then
3:     messagesSet ← messagesSet ∪ (⟨TYPE, h, e, message⟩, j)
```

propose round, and the *vote* round. During the PRE-PROPOSE round, the proposer of the round pre-proposes a value v to all other committee members. During the PROPOSE round, if a committee member accepts v, then it proposes such value. If a committee member receives *enough* proposals for the same value v, then it votes for v during the VOTE round. Finally, if a committee member receives *enough* votes for v, it decides on v.

In this case, *enough* means at least $2f + 1$ occurrences of the same value from $2f + 1$ different committee members; from each committee member, only the first value delivered for each round is considered (cf. Algorithm 1).

If the proposer is correct, it pre-proposes the same value to all the correct committee members (at least $2f+1$). If all the correct committee members propose such value, it follows that all the correct committee members will vote for such value and decide for it. If the proposer is Byzantine, it can pre-propose different values to different correct committee members, creating a partition in the pre-proposal value set collected by committee members. Depending on what the remaining Byzantine committee members do, some correct committee members may decide on a value v, and some other may not (since there are $3f+1$ committee members; there cannot be two different values that collect $2f+1$ distinct votes in the same epoch), then a new epoch starts. To not violate the Agreement property of the consensus (Definition 1), correct committee members that have not decided yet in the previous epoch must only decide for v; for this reason, committee members, before they vote for some value v, *lock* on that value, *i.e.*, they will refuse to propose a further pre-proposed value different than v.

Information from One Epoch to the Next

The two variables $lockedValue$ and $validValue$ carry the potentially decided value from one epoch to the next one. Note that $validValue$ was not present in the earlier versions of Tendermint [33], which was suffering from a livelock issue.

The variable $lockedValue$ represents the following: If one correct committee member decides on v, it means that it collected at least $2f+1$ votes for v during the VOTE phase, since there are at most f Byzantine processes among the committee members; thus, there are at least $f + 1$ correct committee members that voted for v; and those committee members must not vote for any other value different than v. For this reason when a committee member delivers at least $2f + 1$ proposals for a value v during the PROPOSE round, it sets its $lockedValue$ to v, and we say that the committee member is locked on the value v for the corresponding epoch. When a correct committee member is locked on a value, it can only vote for that value.

It can eventually change its locked value only if it delivers enough proposes for a new value more recent than its lock. Hence, if $f + 1$ correct committee members are locked on a same value v, they can only propose and vote for v; any other value $v' \neq v$ can only receive at most $2f$ proposes, which is not enough for both locking and voting, *i.e.*, it is not possible to decide any value different than v.

Variables and data structures

h is an integer representing the consensus instance the committee member is currently executing. e_i is an integer representing the epoch where the committee member i is in; we note that for each height a committee member may have multiple epochs. $decision_i$ is the decision of committee member i for the current consensus instance. $proposal_i$ is the value the committee member i proposes. $vote_i$ is the value the committee member i votes. $lockedValue_i$ stores a value that is potentially decided by some other committee member. If committee member i delivers more than $2f + 1$ proposes for the same value v during its PROPOSE round, it sets $lockedValue_i$ to v. Whenever a process i updates its $lockedValue_i$, its updates the value $lockedEpoch_i$ to the current epoch. This variable keeps track of the epoch at which the $lockedValue_i$ was lastly set.

Each committee member manages timeouts, the `timeoutPrePropose`, `timeoutPropose`, and `timeoutVote`, concerning relatively the PRE-PROPOSE, the PROPOSE and VOTE round respectively. Those timeouts are set to $\Delta_{\text{Pre-propose}}$, Δ_{Propose}, and Δ_{Vote}, respectively, and are started at the beginning of the respective round. They are incremented if they expire before the committee member moves to the next round and if the committee member does not deliver all messages before the end of the timeout.

Functions

We denote as $Value$ the set containing all values (blocks). Let $Height = \mathbb{N}^*$ be the set of all heights, and let $Epoch = \mathbb{N}$ be the set of all epochs for a given height:

- proposer : $Height \times Epoch \rightarrow C \subseteq \Pi_\rho$ is a deterministic function which gives the proposer out of the set of committee members for a given epoch at a given height in a round robin fashion.
- `isValid` : $Value \rightarrow \{\texttt{false}, \texttt{true}\}$ is an application-dependent predicate that is satisfied if the given value is valid with respect to the blockchain. If there is a value v such that $\texttt{isValid}(v) = \texttt{true}$, we say that v is valid. Note that we set $\texttt{isValid}(nil) = \texttt{false}$.
- getValue() $\in Value$ return a valid value.
- sentByProposer : $Height \times Epoch \times Value \rightarrow \{\texttt{false}, \texttt{true}\}$ is a predicate that gives `true` if the given value has been pre-proposed by the proposer of the given height during the given epoch.
- Let $\mathcal{X}$ be the set either of possible proposals or of possible votes. The function $\mathsf{2f+1} : 2^{\mathcal{X}} \rightarrow \{\texttt{false}, \texttt{true}\}$: checks if there are at least $2f + 1$ proposals (resp. votes) in the given set.

Algorithm 2 (part 1) Initial implemented version eventual synchronous Tendermint for height h executed by p_i

```
1:  Initialisation:
2:    e_i := 0                                          /* Current epoch number */
3:    decision_i := nil        /* This variable stocks the decision of the committee member p_i */
4:    lockedValue_i := nil
5:    lockedEpoch_i := −1
6:    proposal_i := getValue()     /* This variable stocks the value the committee member will
      (pre-)propose */
7:    v_i := nil                        /* Local variable stocking the pre-proposal if delivered */
8:    lockedEpoch_j := nil             /* Local variable stocking the proposer's lockedEpoch */
9:    vote_i := nil          /* This variable stock the value the committee member will vote for */
10:   timeoutPrePropose := Δ_Pre-propose; timeoutPropose := Δ_Propose;
      timeoutVote := Δ_Vote

11: Round PRE-PROPOSE:
12:   Send phase:
13:     if decision_i ≠ nil then
14:       ∀v, j : (⟨VOTE, h, e_i, v⟩, j) ∈ messagesSet_i, broadcast⟨VOTE, h, e_i, v⟩
15:       return
16:     if proposer(h, e_i) = i then
17:       broadcast ⟨PRE − PROPOSE, h, e_i, proposal_i, lockedEpoch_i⟩
18:   Delivery phase:
19:     set timerPrePropose to timeoutPrePropose
20:     while (timerPrePropose not expired) ∧ ¬(∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j))
        do
21:       if ∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j) then
22:         v_i ← v_j                                  /* v_j is the value sent by the proposer */
23:         lockedEpoch_j ← e_j                /* e_j is the lockedEpoch sent by the proposer */
24:     if ¬(∃v, epochProp : sentByProposer(h, e_i, v, epochProp)) then
25:       timeoutPrePropose ← timeoutPrePropose + 1
26:   Compute phase:
27:     if 2f+1(⟨PROPOSE, h, lockedEpoch_j, v_i⟩) ∧ lockedEpoch_i ≤ lockedEpoch_j <
        e_i ∧ isValid(v_i) then
28:       lockedValue_i ← nil
29:       lockedEpoch_i ← −1                                          /* p_i unlocks */
30:       proposal_i ← v_i
31:     else
32:       if ¬isValid(v_i) ∨ (lockedEpoch_i > lockedEpoch_j ∧ lockedValue_i ≠ v_i) then
33:         proposal_i ← nil
                                          /* Note that isValid(nil) is set to false */
34:       if isValid(v_i) ∧ (lockedEpoch_i = −1 ∨ lockedValue_i = v_i) then
35:         proposal_i ← v_i
```

3.1.1 First Version

In this section, we present the previous Tendermint consensus algorithm which had a liveness bug. We provide its pseudo-code in Algorithms 1–3.

The algorithm proceeds in three rounds for any given epoch e at height h.

Algorithm 3 (part 2) Initial implemented version eventual synchronous Tendermint for height h executed by p_i

```
1:  Round PROPOSE:
2:    Send phase:
3:      if proposal_i ≠ nil then
4:        broadcast ⟨PROPOSE, h, e_i, proposal_i⟩
5:      broadcast ⟨HeartBeat, PROPOSE, h, e_i⟩
6:    Delivery phase:
7:      set timerPropose to timeoutPropose
8:      while (timerPropose not expires) ∧ ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) do{}  /*
        Note that the HeartBeat messages should be from different committee members */
9:      if ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) then
10:       timeoutPropose ← timeoutPropose + 1
11:   Compute phase:
12:     if ∃v' : 2f+1(⟨PROPOSE, h, e_i, v'⟩) ∧ isValid(v') ∧ sentByProposer(h, e_i, v')
        then
13:       lockedValue_i ← v'
14:       lockedEpoch_i ← e_i
15:       vote_i ← v'
16:     else
17:       vote_i ← nil

18: Round VOTE:
19:   Send phase:
20:     ∀v ∈ Value, ∀j ∈ Π : (⟨PROPOSE, h, e_i, v⟩, j) ∈
        messagesSet_i, broadcast⟨PROPOSE, h, e_i, v⟩
21:     if vote_i ≠ nil then
22:       broadcast ⟨VOTE, h, e_i, vote_i⟩
23:     broadcast ⟨HeartBeat, VOTE, h, e_i⟩
24:   Delivery phase:
25:     set timerVote to timeoutVote
26:     while (timerVote not expires) ∧ ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) do{}
27:     if ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) then
28:       timeoutVote ← timeoutVote + 1
29:   Compute phase:
30:     if ∃v_d, e_d : 2f+1(⟨VOTE, h, e_d, v_d⟩) ∧ isValid(v_d) ∧ decision_i = nil then
31:       decision_i ← v_d
32:     else
33:       e_i ← e_i + 1
34:       v_i ← nil
35:       if lockedValue_i ≠ nil then
36:         proposal_i ← lockedValue_i
37:       else
38:         proposal_i ← getValue()
```

- Round PRE-PROPOSE (Lines 11– 35 of Algorithm 2): If the committee member p_i is the proposer of the epoch, it pre-proposes its proposal value; otherwise, it waits for the proposal from the proposer. The proposal value of the proposer is its $lockedValue_i$ if $lockedValue_i \neq nil$, *i.e.*, if the proposer is already locked. If

a committee member j delivers the pre-proposal from the proposer of the epoch, j checks the validity of the pre-proposal and if valid to accept it with respect to the value of $lockedValue_i$. If the pre-proposal is accepted and valid, j sets its proposal $proposal_j$ to the pre-proposal; otherwise, it sets it to nil.

- Round PROPOSE (Lines 1– 17 of Algorithm 3): During the PROPOSE round, each committee member broadcasts its proposal and collects the proposals sent by the other committee members. After the delivery phase, committee member p_i has a set of proposals and checks if v, pre-proposed by the proposer, was proposed by at least $2f+1$ different committee members; if it is the case and the value is valid, then p_i sets $vote_i$ and $lockedValue_i$ to v; otherwise, it sets $vote_i$ to nil and does not update $lockedValue_i$. If p_i updates $lockedValue_i$ during the PROPOSE round of an epoch e, it sets $lockedValue_i$ to e.

  The PROPOSE round is depicted in Fig. 4.
- Round VOTE (Lines 18– 38 of Algorithm 3): In the round VOTE, an obedient committee member p_i votes $vote_i$ and broadcasts all the proposals it delivered during the current epoch and then p_i collects all the messages that were broadcasted. First, p_i checks if it has delivered at least $2f+1$ of proposal for a value v' pre-proposed by the proposer of the epoch; in that case, it sets $lockedValue_i$ to that value. Then it checks if a value v' pre-proposed by the proposer of the current epoch is valid and has at least $2f+1$ votes. If it is the case, then p_i decides v' and goes to the next height; otherwise, it increases the epoch number and updates the value of $proposal_i$ with respect to $lockedValue_i$.

  The VOTE round is depicted in Fig. 5.

We recall that each committee member has a timeout for each round. If during a round committee member p_i does not deliver at least $2f+1$ messages sent during that round (or the pre-proposal for the PRE-PROPOSE round), the corresponding timeout is increased. Those messages can be values or heartbeats, in the case in which a correct committee member has not a value to propose or vote.

A normal execution, with four correct processes is depicted in Fig. 6. The proposer p_1 sends the pre-proposal B to all other committee members, each delivers the pre-proposal at the beginning of their PROPOSE round, accepts it, and sends a proposal for B. All committee members receive the proposals from all other

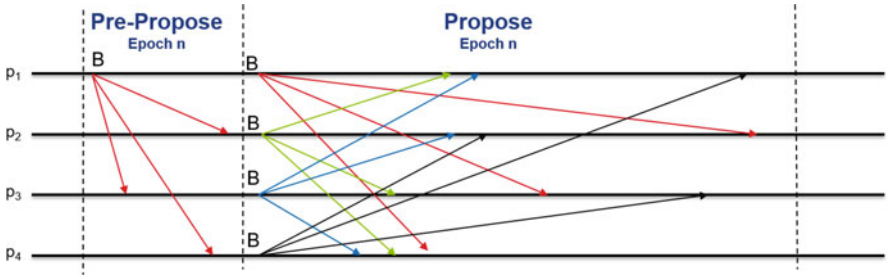


Fig. 4 Propose phase

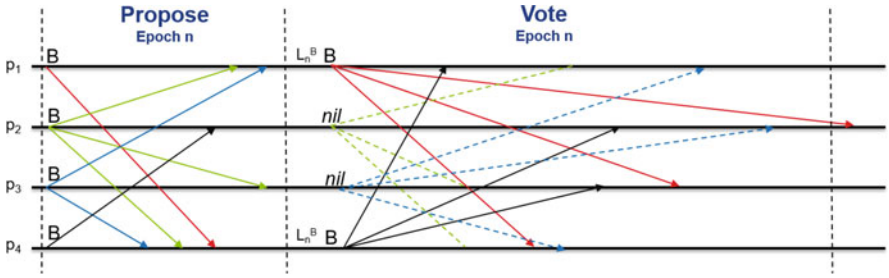


Fig. 5 VOTE round of the Tendermint algorithm with four correct processes

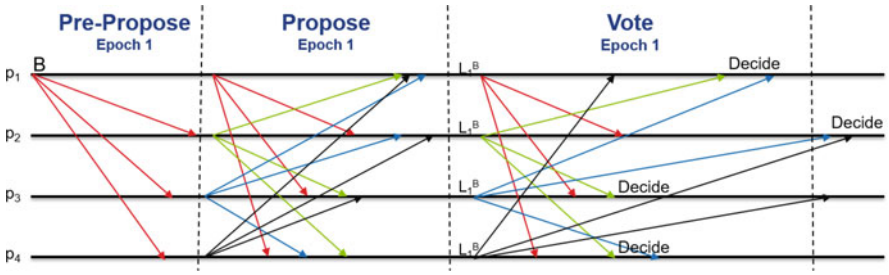


Fig. 6 Normal execution of the Tendermint algorithm with four correct processes

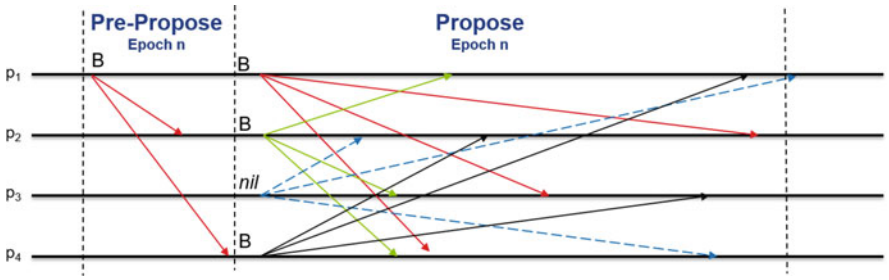


Fig. 7 Propose phase: when a process did not deliver the block or finds it invalid, it proposes *nil*

committee members before the VOTE round (so at least $2f + 1$); therefore, they lock on B during the current epoch 1 and vote for the block. All committee members deliver vote for the block B corresponding to epoch 1, so they decide.

Specific executions of the PROPOSE round are sketched in Fig. 7, 8, and 9.

3.1.2 The Livelock Bug

In [15], the authors advocate without providing any evidence that there is a livelock problem in the Tendermint's description proposed in the Buchman's master manuscript [11].

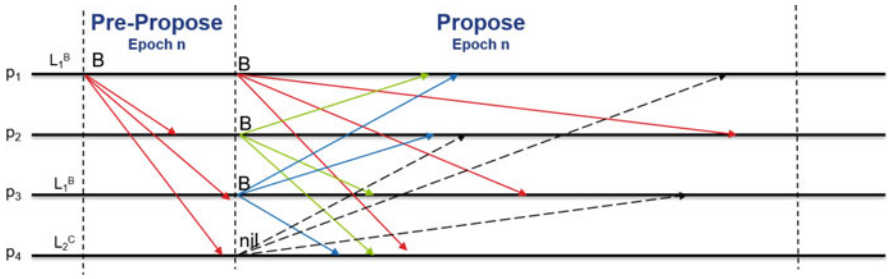


Fig. 8 Propose phase: when a process is locked on a "Newer" block

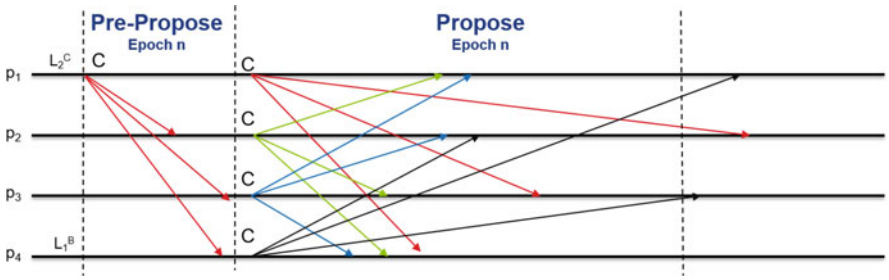


Fig. 9 Propose phase: when a process delivers a "Newer" block than its locked value

Hereafter, we exhibit this bug in details. To do so, let us consider the example where the system consists of $four$ processes $\{p_1, p_2, p_3, p_4\}$, such that , p_1, p_2, and p_3 are correct and p_4 is Byzantine.

We will mostly focus on the variables $lockedValue$ and $lockedEpoch$ of the processes. Since p_4 is Byzantine, we will not talk nor discuss about its variables. Let us also assume that the order of the proposer (in a round robin fashion) is p_1, then p_2, then p_3, then p_4, then p_1, etc.

Let us assume when entering the synchronous period p_1 is locked on value B and no other process is locked. Without loss of generality, we start the synchronous period by epoch 2, and we assume that p_1 locked during epoch 1 on the block B. At the beginning of epoch 2, we have the following state: $lockedValue_1 = B$ and $lockedEpoch_1 = 1$; $lockedValue_2 = \perp$ and $lockedEpoch_2 = -1$; $lockedValue_3 = \perp$ and $lockedEpoch_3 = -1$. During the synchronous period, communication between correct processes is reliable and timely, *i.e.*, all correct processes deliver messages sent from all correct processes in the same round. That assumption does not hold for Byzantine processes. In fact, a Byzantine process can send a message to a correct process at the end on the round, such that it is delivered by the other correct processes not in the same but in a subsequent round.

The live lock scenario is exhibited in Fig. 10, and the description is presented in the following:

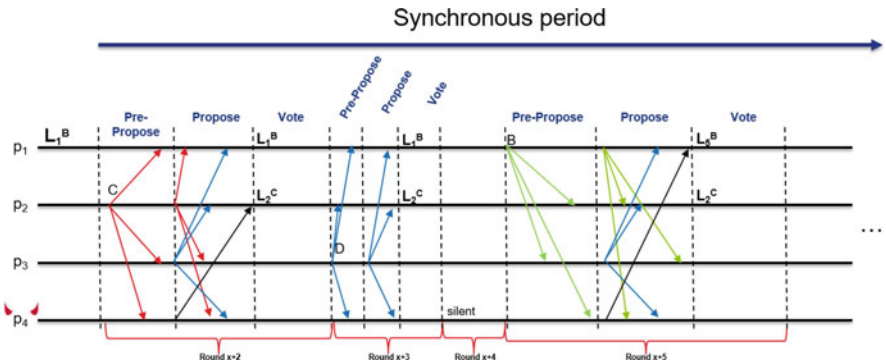


Fig. 10 Livelock scenario preventing liveness

Epoch 2 – p_2 is the proposer. Since p_2 is not locked, it can pre-propose any value, for instance, value C, which is valid with respect to the application.
p_1 being locked on B at round 1 does not accept the pre-proposal for C (line 32 of Algorithm 2). p_2 and p_3 accept it and propose C (line 34 of Algorithm 2). Therefore, all correct processes will deliver at the end of the PROPOSE round at least two propose messages for C. Now, assume that p_4, the Byzantine process, sends at the very end of the PROPOSE round, a propose message for C only to process p_2 such that no other process has the time to deliver the message before the end of the round. p_2 then delivers three propose messages (which is $2f + 1$ propose messages) for C, and p_1 and p_3 deliver exactly two propose messages for C (strictly less than $2f + 1$ propose messages) during the PROPOSE round.
During the VOTE round of epoch 2, p_2 then locks on C and sends a vote message for C (Lines 12–14 of Algorithm 2), while p_1 and p_3 do not have a message to vote for.
The state of the processes at the end of epoch 2 and beginning of epoch 3 is $lockedValue_1 = B$ and $lockedEpoch_1 = 1$, $lockedValue_2 = C$ and $lockedEpoch_2 = 2$, and $lockedValue_3 = \perp$ and $lockedEpoch_3 = -1$.

Epoch 3 – p_3 is the proposer. Since p_3 is not locked, it can pre-propose any value, for instance, value D, which is valid with respect to the application.
p_1 being locked on B at round 1 and p_2 being locked on C at round 2 do not accept the pre-proposal for D (line 32 of Algorithm 2).
Only p_3 accepts and sends a propose message for D (line 34 of Algorithm 2); no process gets the minimal $three$ messages for vote, so none of the processes votes for D.
The state of the processes at the end of epoch 3 and beginning of epoch 4 is $lockedValue_1 = B$ and $lockedEpoch_1 = 1$, $lockedValue_2 = C$ and $lockedEpoch_2 = 2$, and $lockedValue_3 = \perp$ and $lockedEpoch_3 = -1$.

Epoch 4 – p_4 is the proposer. Recall that p_4 is Byzantine. It stays silent and does not send a pre-proposal. Nothing changes with respect to the previous

epoch, so the state of the correct committee members is as follows: $lockedValue_1 = B$ and $lockedEpoch_1 = 1$, $lockedValue_2 = C$ and $lockedEpoch_2 = 2$, and $lockedValue_3 = \perp$ and $lockedEpoch_3 = -1$.

Epoch 5 – p_1 is the proposer. Since p_1 is locked on B, it pre-proposes B, with the epoch being locked on (line 17 of Algorithm 2). p_3 is not locked, so it accepts and sends a proposal for B (line 34 of Algorithm 2). p_2 in the other hand is locked on C on epoch 2, which is bigger than 1, the locked epoch of p_1, so p_2 does not accept B (line 32 of Algorithm 2).
Therefore, p_1 and p_3 will deliver at the end of the PROPOSE round at least two propose messages for B. Now, assume that p_4, the Byzantine process, sends at the very end of the PROPOSE round a propose message for B only to process p_1 such that no other process has the time to deliver the message before the end of the round. p_1 then delivers three propose messages (which is $2f+1$ propose messages) for B, and p_2 and p_3 deliver exactly two propose messages for B (strictly less than $2f + 1$ propose messages) during the PROPOSE round.
During the VOTE round of epoch 5, p_1 then locks again on B, hence updates it locked epoch to 5, and sends a vote message for B (Lines 12–14 of Algorithm 2), while p_2 and p_3 do not have a message to vote for.

Epoch 6 – p_2 is the proposer. Since p_2 is locked on C, it pre-proposes C, with the epoch (epoch 2) being locked on (line 17 of Algorithm 2). p_3 is not locked, so it accepts and sends a proposal for B (line 34 of Algorithm 2). p_1 on the other hand is locked on B, now on epoch 5, which is bigger than 2, the locked epoch of p_2, so p_1 does not accept C (line 32 of Algorithm 2).
Therefore, p_2 and p_3 will deliver at the end of the PROPOSE round at least two propose messages for C. Now, assume that p_4, the Byzantine process, sends at the very end of the PROPOSE round, a propose message for C only to process p_2 such that no other process has the time to deliver the message before the end of the round. p_2 then delivers three propose messages (which is $2f+1$ propose messages) for C, and p_2 and p_3 deliver exactly two propose messages for C (strictly less than $2f + 1$ propose messages) during the PROPOSE round.
During the VOTE round of epoch 6, p_2 then locks again on C, hence updates it locked epoch to 6, and sends a vote message for C (Lines 12–14 of Algorithm 2), while p_1 and p_3 do not have a message to vote for.

...

This scenario can continue forever, and therefore, no correct committee member will be able to decide.

This bug however requires a strong collaboration between Byzantine processes and needs Byzantine processes to send messages at the right time such that it does not diffuse too fast to other processes.

3.1.3 Corrected Version

Algorithm 4 (part 1) Eventual synchronous Tendermint for height h executed by p_i

```
1:  Initialisation:
2:    e_i := 0                                                    /* Current epoch number */
3:    decision_i := nil        /* This variable stocks the decision of the committee member p_i */
4:    lockedValue_i := nil
5:    lockedEpoch_i := −1;
6:    validValue_i := nil; validEpoch_i := −1
7:    proposal_i := getValue()     /* This variable stocks the value the committee member will
      (pre-)propose */
8:    v_i := nil                          /* Local variable stocking the pre-proposal if delivered */
9:    validEpoch_j := nil                   /* Local variable stocking the proposer's validEpoch */
10:   vote_i := nil          /* This variable stock the value the committee member will vote for */
11:   timeoutPrePropose := Δ_Pre-propose; timeoutPropose := Δ_Propose;
      timeoutVote := Δ_Vote

12: Round PRE-PROPOSE:
13:   Send phase:
14:     if decision_i ≠ nil then
15:       ∀v, j : (⟨VOTE, h, e_i, v⟩, j) ∈ messagesSet_i, broadcast⟨VOTE, h, e_i, v⟩
16:       return
17:     if proposer(h, e_i) = i then
18:       broadcast ⟨PRE − PROPOSE, h, e_i, proposal_i, validEpoch_i⟩
19:   Delivery phase:
20:     set timerPrePropose to timeoutPrePropose
21:     while (timerPrePropose not expired) ∧ ¬(∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j))
        do
22:       if ∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j) then
23:         v_i ← v_j                                          /* v_j is the value sent by the proposer */
24:         validEpoch_j ← e_j                        /* e_j is the validEpoch sent by the proposer */
25:     if ¬(∃v, epochProp : sentByProposer(h, e_i, v, epochProp)) then
26:       timeoutPrePropose ← timeoutPrePropose + 1
27:   Compute phase:
28:     if 2f+1(⟨PROPOSE, h, validEpoch_j, v_i⟩) ∧ validEpoch_j ≥ lockedEpoch_i ∧
        validEpoch_j < e_i ∧ isValid(v_i) then
29:       proposal_i ← v_i
30:     else
31:       if ¬isValid(v_i) ∨ (lockedEpoch_i > validEpoch_j ∧ lockedValue_i ≠ v_i) then
32:         proposal_i ← nil
                                                  /* Note that isValid(nil) is set to false */
33:       if isValid(v_i) ∧ (lockedEpoch_i = −1 ∨ lockedValue_i = v_i) then
34:         proposal_i ← v_i
```

In this section, we describe the corrected version of the algorithm focusing on the modifications that address the previously described bug. One can note that the modification of the algorithm seems to be minor. In fact, the main difference is the introduction of the variables $validValue$ and $validEpoch$ (line 6 of Algorithm 4).

Algorithm 5 (part 2) Eventual synchronous Tendermint for height h executed by p_i

```
Round PROPOSE:
  Send phase:
    if proposal_i ≠ nil then
      broadcast ⟨PROPOSE, h, e_i, proposal_i⟩
    broadcast ⟨HeartBeat, PROPOSE, h, e_i⟩
  Delivery phase:
    set timerPropose to timeoutPropose
    while (timerPropose not expires) ∧ ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) do{}  /*
    Note that the HeartBeat messages should be from different committee members */
    if ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) then
      timeoutPropose ← timeoutPropose + 1
  Compute phase:
    if ∃v' : 2f+1(⟨PROPOSE, h, e_i, v'⟩) ∧ isValid(v') ∧ sentByProposer(h, e_i, v')
    then
      lockedValue_i ← v'
      lockedEpoch_i ← e_i
      validValue_i ← v'
      validEpoch_i ← e_i
      vote_i ← v'
    else
      vote_i ← nil

Round VOTE:
  Send phase:
    ∀v ∈ Value, ∀j ∈ Π : (⟨PROPOSE, h, e_i, v⟩, j) ∈
    messagesSet_i, broadcast⟨PROPOSE, h, e_i, v⟩
    if vote_i ≠ nil then
      broadcast ⟨VOTE, h, e_i, vote_i⟩
    broadcast ⟨HeartBeat, VOTE, h, e_i⟩
  Delivery phase:
    set timerVote to timeoutVote
    while (timerVote not expires) ∧ ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) do{}
    if ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) then
      timeoutVote ← timeoutVote + 1
  Compute phase:
    if ∃v'' : 2f+1(⟨PROPOSE, h, e_i, v''⟩) ∧ isValid(v'') ∧ sentByProposer(h, e_i, v'')
    then
      validValue_i ← v''
      validEpoch_i ← e_i
    if ∃v_d, e_d : 2f+1(⟨VOTE, h, e_d, v_d⟩) ∧ isValid(v_d) ∧ decision_i = nil then
      decision_i ← v_d

    else
      e_i ← e_i + 1
      v_i ← nil
      if validValue_i ≠ nil then
        proposal_i ← validValue_i
      else
        proposal_i ← getValue()
```

However, these variables and how they are handled together with $lockedValue$ and $lockedEpoch$ are the key ingredient of solving the aforementioned live lock issue.

More technically, the variable $validValue_i$ stores a value that is potentially decided by some other committee members. If the committee member p_i delivers at least $2f + 1$ proposes for the same value v (from different committee members) whether during its PROPOSE round or its VOTE round, it sets $validValue_i$ to v. $validValue_i$ is the last proposed value that a committee member delivered at least $2f + 1$ times; it can be different than $lockedValue_i$. The latter two variables are used as follows: if p_i is the proposer, then p_i pre-proposes $validValue_i$ if different from nil. Otherwise, if p_i is a committee member, it checks the new pre-proposal against $lockedValue_i$ and $validValue_i$ if those are different from nil.

Intuitively, it solves the livelock bug as follows: When new pre-proposals were checked only against the $lockedValue$ (as in the previous version), we had that a correct validator locked on a value v refuses (and does not propose) values different from v; it means that when some correct validator is locked, the proposer needs to pre-propose some of the value on which the correct validators are locked on. However, such value, in order to be accepted, cannot be checked only against the $lockedValue$ because we may never have the correct validator proposing such value (the livelock bug). In this case, the $lockedValue$ serves two purposes: (i) being pre-proposed when different than nil at the proposer and (ii) used to decide when a new pre-propose is received if to propose it or not. In the second version, the $validValue$ lightens the task of the $lockedValue$. Indeed, this latter is employed as pre-proposal (when set) then, when received by validators, is checked against their $lockedValue$. By construction of the algorithm, all correct validators have the same $validValue$ at the end of the epoch in the synchronous period. This implies that if a correct committee member locks, then all the other correct committee members set their valid value accordingly – the next correct proposer pre-proposes such value that is then proposed by all the correct committee members.

The algorithm proceeds in the same fashion, *i.e.*, in three rounds for any given epoch e at height h. We only highlight the presence of the two new variables:

- Round PRE-PROPOSE (Lines 12–34 of Algorithm 4): The description of this round is mainly the same as before. We highlight the fact that a correct committee member p_i also takes into account $validValue_i$ and $validEpoch_i$ to accept a pre-proposed value. A correct proposer sends not its $lockedValue_i$, but its $validValue_i$ if it is not nil.
- Round PROPOSE (Lines 1– 19 of Algorithm 5): When a correct committee member p_i updates $lockedValue_i$ (resp. $validValue_i$), it also updates $lockedEpoch_i$ (resp. $validEpoch_i$) to the current epoch.
- Round VOTE (Lines 20– 43 of Algorithm 5): If a correct committee member p_i delivered at least $f + 1$ times the same type of messages from an epoch higher than the current one, p_i moves directly to the PRE-PROPOSE round of that epoch and when a correct committee member p_i updates $validValue_i$, it also updates $validEpoch_i$ to the current epoch.

Complexity

There exist many different consensus algorithms, and many more are continuously proposed. They usually have different efficiency or are more or less adapted to some specific applications. To compare different consensus algorithms, many factors can be taken into account; the most popular comparison tool is the "complexity" of the algorithms. The most common for Byzantine agreement algorithms are *bit*, *message*, and *round* complexity in the worst-case scenario during the period of synchrony before reaching agreement. In particular, bit complexity refers to the number of information exchanged by processes in the agreement process; for simplicity here, we refer to it as the number of information each process must store, message complexity is the number of messages exchanged by correct processes, and the round complexity is the maximum number of rounds needed to reach an agreement.

Note that in PBFT in synchronous rounds, when the leader is correct, the complexity of the protocol is $O(n^2)$. Otherwise, a view change mechanism takes place to change the leader and resume the computation. The view change is used to avoid that; in case of a faulty leader, if some correct process decides on a value v, other correct processes cannot decide on a value $v' \neq v$ when the new leader proposes a new value.

The view change mechanism in classical consensus algorithms implies that when a leader is suspected to be faulty, all processes have to collect enough evidences for the view change. That is, the view change message contains at least $2f + 1$ signed messages, and these messages are sent from at least $2f + 1$ processes which yields a message complexity of $O(n^2)$. These messages are then sent to all processes, and the view change has then $O(n^3)$ message complexity. Since the protocol terminates when there is a correct leader, which may happen for the first time in epoch $f + 1$, then in the worst-case scenario, it has a message complexity of $O(n^4)$.

Interestingly, Tendermint and similar recent approaches use an alternative mechanism for leader replacement that allows to drop message complexity to $O(n^3)$ and even $O(n^2)$ when using threshold or collective signatures. Processes, instead of exchanging all the messages they already delivered (used previously to trigger a view change), locally keep track of potentially decided values.

In [31], Kowalski and Mostéfaoui propose an algorithm for consensus in synchronous setting with a bit complexity of $O(n^3 \log n)$ for an optimal round complexity of $f + 1$. There is a trade-off between the bit complexity and the round complexity of the Byzantine agreement.

We present in Table 1 the complexity of some consensus algorithms that are developed for eventual synchronous systems.

Table 1 Comparison of BFT consensus algorithms in the eventual synchronous system

Protocol	Message complexity	Rounds complexity	Bit complexity	Phases per view
DBFT [20]	$O(n^3)$	–	?	–
PBFT [16]	$O(n^4)$	$f + 1$	*poly*	3
HotStuff [50]	$O(n^2)$	$f + 1$	$O(n^2 \log n) *$ threshold signature	4
Tendermint [12]	$O(n^3)$	n	$O(n^2 \log n)$	3

4 An Algorithmic Perspective of Blockchain Consensus

Historically, consensus has been formulated to deal with concurrency and failures arising in distributed computations: input values for consensus can be viewed as concurrent proposals, and then, during the consensus algorithm execution, participants vote for proposals and agree on one of them by reaching a majority quorum. When a consensus algorithm terminates, all the correct processes are synchronized, i.e., they all stick on the agreed value and know that all others do the same. If the agreed value is the state of the blockchain, i.e., a block, then a consensus is reached on the blockchain state at the end of the algorithm execution. To solve consensus in these terms, the designer usually knows the set of n processes involved in the computation, as such n is used in consensus's algorithms to set the size of different data structures and determine quora. In public blockchains on the contrary, n is not known in advance; therefore, classical consensus algorithms do not directly apply. The fact that Bitcoin was designed for an open system where n is not known has been probably one of the reasons why Bitcoin algorithmic techniques to achieve agreement on the state of the chain depart completely from those (nontrivial) techniques used in classical consensus algorithms. More precisely, we can say that from an algorithmic viewpoint, Bitcoin employs a simple reliable diffusion technique: as soon as a block is created through proof-of-work, it is broadcast to all participants in the network. When multiple chains arise, because of concurrency or malicious behavior, the longest-chain rule is locally used by each correct (honest) participant to converge to a single state. In a nutshell, Bitcoin reaches consensus on the state of the chain through a simple algorithm, without relying on consensus. This can be possible because Bitcoin algorithmic techniques have been designed to *prevent* concurrent proposals as much as possible to *in fine* avoid to resort to classical consensus algorithmic techniques.

The lack of consensus in Bitcoin, however, implies that processes do not explicitly synchronize on the state of the chain, whereas consensus's explicit synchronization implies that there exists a point in time in which a participant knows that agreement has been reached in the whole system. This consensus awareness is not achievable in Bitcoin: a process never knows if a valid block in the chain could be revoked, being orphaned, at a later time.

Blockchains using Byzantine fault-tolerant (BFT) consensus aim at marrying the best of both worlds: achieving agreement awareness, in a network of an unknown number of participants. More in detail, those that first synchronize on the state of the chain are committees that create and agree on a new block through BFT consensus. The new block is then diffused by the committee to all other participants. Let us note that the diffusion-and-validation scheme between committees and participants is the same as for Bitcoin, with the only difference that, thanks to the intra-committee BFT consensus, exactly one valid block is created for each height of the blockchain and that validation consists solely of verifying the quorum of the committee that voted for the block. When a participant validates and adopts the block, she also knows that the block will be irrevocably adopted by all honest participants in the network,

for that height of the chain. This last feature, which is called in blockchain jargon *instant finality*, comes from the use of consensus and its explicit synchronization of (a part of) the network.

5 Future Directions

Being able to solve consensus in the blockchain context is only the beginning. Indeed, this opens to numerous future research directions that need to be addressed to put in place a long-lasting blockchain system. The main challenge to deal with is the necessary consensus condition on the fraction of tolerated Byzantine participants. Such a challenge requires investigating how to select the committee and how to behave correctly or to propose incentive processes such that the threshold holds. This demands both an incentive system (a rewarding mechanism) and the capability to account for processes that misbehave (and their removal from the system). However, blockchain solutions also have to be able to take countermeasures when the assumption of the fraction of Byzantine processes does not hold. In that case, we have either no block production or forks, which demands to featuring either a committee change or a fork recovery procedure.

6 Conclusion

In this chapter, we went through the Tendermint blockchain platform. We discussed the Tendermint blockchain architecture, and later we focused on the Tendermint core algorithm. We detailed why the Tendermint core algorithm is relevant in message complexity, with respect to already existing consensus algorithms. Finally, we gave two versions of the solution: the incorrect and the correct version. We believe that going through the two versions of the algorithm is of pedagogical interest to understand all the algorithm mechanisms better.

References

1. I. Abraham, D. Malkhi, K. Nayak, L. Ren, A. Spiegelman, Solida: a blockchain protocol based on reconfigurable Byzantine consensus, in *21st International Conference on Principles of Distributed Systems, OPODIS 2017*, Lisbon, Portugal, 18–20 Dec 2017 (2017), pp. 25:1–25:19
2. Agoric, (2020) [Online; Accessed 20 Oct 2020]
3. Y. Amoussou-Guenou, A. Del Pozzo, M. Potop-Butucaru, S. Tucci-Piergiovanni, Correctness of tendermint-core blockchains, in *22nd International Conference on Principles of Distributed Systems (OPODIS 2018), Leibniz International Proceedings in Informatics (LIPIcs)* (2018), pp. 16:1–16:16

4. Y. Amoussou-Guenou, A. Del Pozzo, M. Potop-Butucaru, S. Tucci-Piergiovanni, Dissecting tendermint, in *Networked Systems – 7th International Conference, NETYS 2019*, Marrakech, Morocco, 19–21 June 2019. Revised Selected Papers, Lecture Notes in Computer Science, vol. 11704 (Springer, 2019), pp. 166–182
5. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A.D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S.W. Cocco, J. Yellick, Hyperledger fabric: a distributed operating system for permissioned blockchains, in *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018*, Porto, Portugal, 23–26 Apr 2018 (ACM, 2018), pp. 30:1–30:15
6. Aragon Association. Aragon network (2020). https://aragon.org/token/ant, [Online; Accessed 20 Oct 2020]
7. I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, Proof of activity: extending bitcoin's proof of work via proof of stake [extended abstract] y. ACM SIGMETRICS Perform. Eval. Rev. **42**(3), 34–37 (2014)
8. BigchainDB GmbH. BigchainDB (2020). https://www.bigchaindb.com/, [Online; Accessed 20 Oct 2020]
9. Binance. Binance chain (2020). [Online; Accessed 20 Oct 2020]
10. BSN. The blockchain-based service network (2020). https://bsnbase.io/g/main/index, [Online; Accessed 20 Oc 2020]
11. E. Buchman, Tendermint: Byzantine Fault tolerance in the age of blockchains. Master thesis, Guelph University, 2016
12. E. Buchman, J. Kwon, Z. Milosevic, The latest gossip on BFT consensus. Technical report, Tendermint, 2018
13. V. Buterin et al., Ethereum white paper. GitHub repository, vol. 1 (2013), pp. 22–23
14. C. Cachin, K. Kursawe, F. Petzold, V. Shoup, Secure and efficient asynchronous broadcast protocols, in *Advances in Cryptology – CRYPTO 2001, 21st Annual International Cryptology Conference*, Santa Barbara, California, USA, 19–23 Aug 2001. Proceedings, Lecture Notes in Computer Science, vol. 2139 (Springer, 2001), pp. 524–541
15. C. Cachin, M. Vukolic, Blockchain consensus protocols in the wild (keynote talk), in *31st International Symposium on Distributed Computing, DISC 2017*, 16–20 Oct 2017, Vienna, Austria. LIPIcs, vol. 91. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2017), pp. 1:1–1:16.
16. M. Castro, B. Liskov, Practical byzantine fault tolerance, in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, USA, 22–25 Feb 1999. (USENIX Association, 1999), pp. 173–186
17. M. Castro, B. Liskov, Practical Byzantine fault tolerance, in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)* (1999)
18. L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, W. Shi, On security analysis of proof-of-elapsed-time (poet), in *International Symposium on Stabilization, Safety, and Security of Distributed Systems* (Springer, 2017), pp. 282–297
19. T. Crain, V. Gramoli, M. Larrea, M. Raynal, (leader/randomization/signature)-free Byzantine consensus for consortium blockchains (2017) CoRR abs/1702.03068v2
20. T. Crain, V. Gramoli, M. Larrea, M. Raynal, DBFT: efficient leaderless Byzantine consensus and its application to blockchains, in *17th IEEE International Symposium on Network Computing and Applications, NCA 2018*, Cambridge, MA, USA, 1–3 Nov 2018 (IEEE, 2018), pp. 1–8
21. A. de Vries, Bitcoin's energy consumption is underestimated: a market dynamics approach. Energy Res. Soc. Sci. **70**, 101721 (2020)
22. C. Decker, J. Seidel, R. Wattenhofer, Bitcoin meets strong consistency, in *Proceedings of the 17th International Conference on Distributed Computing and Networking Conference (ICDCN)* (2016)
23. C. Dwork, N.A. Lynch, L.J. Stockmeyer, Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988)

24. C. Dwork, M. Naor, Pricing via processing or combatting junk mail, in *Advances in Cryptology – CRYPTO '92, 12th Annual International Cryptology Conference*, Santa Barbara, California, USA, 16–20 Aug 1992. Proceedings, Lecture Notes in Computer Science, vol. 740 (Springer, 1992), pp. 139–147
25. S. Dziembowski, S. Faust, V. Kolmogorov, K. Pietrzak, Proofs of space, in *Annual Cryptology Conference* (Springer, 2015), pp. 585–605
26. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: scaling Byzantine agreements for cryptocurrencies, in *Proceedings of the 26th Symposium on Operating Systems Principles*, Shanghai, China, 28–31 Oct 2017 (ACM, 2017), pp. 51–68
27. ISISnet. Isisnet (2020). https://www.irisnet.org/ [Online; Accessed 20 Oct 2020]
28. S. King, S. Nadal, PPCoin: peer-to-peer crypto-currency with proof-of-stake (2012). https://www.peercoin.net/whitepapers/peercoin-paper.pdf, [Online; Accessed 10 July 2020]
29. E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing bitcoin security and performance with strong consistency via collective signing, in *25th USENIX Security Symposium, USENIX Security 16*, Austin, TX, USA, 10–12 Aug 2016 (USENIX Association, 2016), pp. 279–296
30. E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing Bitcoin security and performance with strong consistency via collective signing, in *Proceedings of the 25th USENIX Security Symposium* (2016)
31. D.R. Kowalski, A. Mostéfaoui, Synchronous Byzantine agreement with nearly a cubic number of communication bits, in *ACM Symposium on Principles of Distributed Computing, PODC'13*, Montreal, QC, Canada, 22–24 July 2013 (ACM, 2013), pp. 84–91
32. J. Kwon, Tendermint: consensus without mining (2014). https://tendermint.com/static/docs/tendermint.pdf, [Online; Accessed 10 July 2020]
33. J. Kwon, E. Buchman, Tendermint. https://tendermint.readthedocs.io/en/master/specification.html (Visited on 22 May 2018)
34. L. Lamport, R.E. Shostak, M.C. Pease, The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
35. G. Markou, C. Schwarz, Aragon chain (2020). https://github.com/aragon/aragon-chain-spec/, [Online; Accessed 02 Aug 2021]
36. J. Martin, L. Alvisi, Fast Byzantine consensus. IEEE Trans. Dependable Secur. Comput. **3**(3), 202–215 (2006)
37. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system (2008) https://bitcoin.org/bitcoin.pdf, [Online; Accessed 10 July 2020]
38. M. Nugent, R.G. Lennon, Blockchain for decentralized data analysis, in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (IEEE, 2019), pp. 1059–1060
39. M.C. Pease, R.E. Shostak, L. Lamport, Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
40. Starname. IOV Name Service (2020). [Online; Accessed 20 Oct 2020]
41. I. Stewart, Proof of burn–bitcoin wiki (2012). [Online; Accessed 30 July 2021]
42. Tendermint. Cosmos SDK (2020). https://cosmos.network/ecosystem/apps, [Online; Accessed 02 Aug 2021]
43. Tendermint and Cosmos. Anatomy of an SDK Application | Core Concepts (2020). https://docs.cosmos.network/master/basics/app-anatomy.html, [Online; Accessed 12 Nov 2020]
44. Tendermint and Cosmos. Baseapp | Core concepts (2020). https://docs.cosmos.network/master/core/baseapp.html, [Online; Accessed 12 Nov 2020]
45. Tendermint Core. Applications | Tendermint spec (2020). https://docs.tendermint.com/master/spec/abci/apps.html, [Online; Accessed 12 Nov 2020]
46. Tendermint Core. Methods and types | Tendermint spec (2020). https://docs.tendermint.com/master/spec/abci/abci.html, [Online; Accessed 12 Nov 2020]
47. Terra. Terra (2020). https://terra.money/, [Online; Accessed 20 Oct 2020]

48. The DiemBFT Team, M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, A. Sonnino, State machine replication in the diem blockchain. Technical report, The Libra Association (2019). Version 26 May 2020, [Online; Accessed 02 Aug 2021]
49. S. Toueg, Randomized Byzantine agreements, in *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, BC, Canada, 27–29 Aug 1984 (ACM, 1984), pp. 163–178
50. M. Yin, D. Malkhi, M.K. Reiter, G. Golan-Gueta, I. Abraham, HotStuff: BFT consensus with linearity and responsiveness, in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019*, Toronto, ON, Canada, 29 July–2 August 2019 (ACM, 2019), pp. 347–356

Part III
Security and Scalability

The Security of Delegated Proof-of-Stake Wallet and Stake Pools

Mario Larangeira and Dimitris Karakostas

1 Introduction

In order to better explain the advantages of proof of stake (PoS), it is convenient to review the mechanism which has similar motivation and is widely used in blockchain-based systems: proof of work (PoW). Hence, it is convenient to start by briefly describing PoW-based systems first, since it works, at the same time, as introduction to the blockchain systems, describing its basic structure, and the security guarantees that it can provide. The PoW design has shown it can provide security guarantees at the cost of significant energy consumption rates. This approach, highly based on the computational power of the parties, was initially proposed to fight e-mail spam [11].

In the blockchain scenario, it is used in a rather different manner. It plays a crucial role to prevent the disruption of the distributed system by an adversary who can control more than half of the combined, i.e., the sum of all parties of the network, computational power. This design was implemented in the real world firstly by the Bitcoin protocol [24]. Arguably, PoS systems are considered more advanced than PoW; however, typically, they also rely on the same basic structure for the blockchain data structure while offering similar security guarantees efficiently, i.e., without the high energy consumption drawback.

The final goal of the arrangement between PoS (as it is also for the case of PoW) and blockchain data structure is to construct a *decentralized ledger* with very

M. Larangeira (✉)
Tokyo Institute of Technology/IOHK, Department of Mathematical and Computing Science/Research Division, Meguro, Tokyo, Japan
e-mail: mario@c.titech.ac.jp; mario.larangeira@iohk.io

D. Karakostas
University of Edinburgh, Informatics, Edinburgh, Scotland, United Kingdom
e-mail: d.karakostas@ed.ac.uk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_8

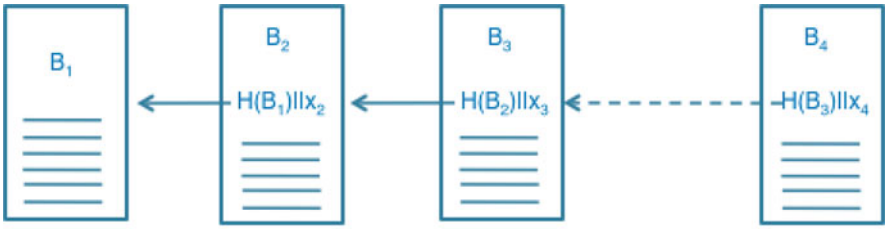


Fig. 1 The blocks contain a hash value, relying on a hash function H and a *nonce* value x chosen for each block. Since each block i contains the value of the previous block B_{i-1} concatenated with the nonce x_i, it makes hard to tamper the contents of each block

well-defined security properties, namely, *liveness* and *robustness*. Informally, these properties guarantee that contents can be added and not changed or reordered while preventing any reasonable set of malicious participants to disrupt the system.

1.1 The Blockchain Data Structure

The term *blockchain* became synonymous of a class systems. Technically, it is a very specific *data structure*, which in its simplest form is composed of *blocks*, i.e., pieces of information, concatenated to each other linearly. Crucially, the order of the blocks, or even their contents, should not be easily altered in order to implement the core property of a blockchain-based system: *immutability*. In other words, it is an *add-only* database where the entries cannot be reordered.

A blockchain-based system, as it was first conceived by Satoshi Nakamoto [24], does not allow changes in the contents once they are persisted. This property is intrinsically related to the relation between the blocks of the *chain*. Each block relates to each other via a *hash link*. That is, each block contains the *digest*, i.e., a compact representation, of the previous block. Any change in the contents would affect the digest. Moreover, a malicious participant aiming to make changes for a given digest value, that is, without affecting the next block, would have to break fundamental computational properties of one of the most useful primitives in computer science: the hash function. In other words, we can assume such a feat is unlikely to happen.

The digest of the previous block prevents that a small change is applied to the original data set, since if it is changed the digest is largely affected, allowing the easy verification of the integrity of the data and affecting the next block. The consequence that the next block digest would also have to be recomputed. The common term used to identify the specialized nodes that generate the blocks is "miner." The terminology dates back early days of the Bitcoin protocol and the analogy of creating Bitcoins and the mining of physical gold. Each miner performs the proof of wok, in order to add a new block to the whole data structure, e.g., the new block B_4 in tracing lines of Fig. 1.

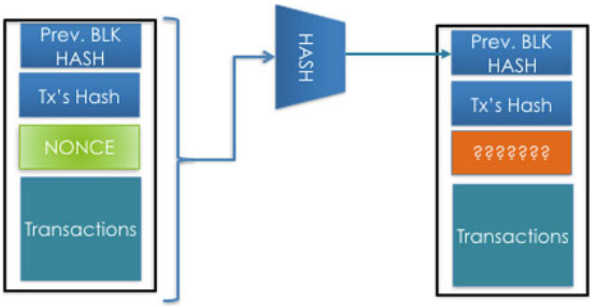


Fig. 2 As mentioned earlier, each block contains the nonce concatenated with the digest of the contents of the previous block. The set of transactions is contained as a *digest* Tx's hash of all the transactions of the block

1.2 The Block Structure

Each block is linked to its predecessor in the chain by the hash link. This value is one of the components of the block. Typically, the block can be divided into four main parts:

1. **Hash link**: The hash link is the value that connects the current block with its predecessor. This value is computed by hashing the concatenation of the contents of the previous block and the *nonce* value.
2. **Nonce**: This value is the "missing piece" to be found in order to generate a new block. Proof-of-work protocol participant performs trial and error in order to find this value.
3. **Set of transactions**: Each block contains several ordered transactions, which the block issuer collected and decided to include.
4. **Root of Merkle tree of transactions**: The root of the Merkle tree of all transactions of the block are organized in a Merkle tree.

Figure 2 illustrates these four main sections of the block in the Bitcoin protocol [24].

In the case of the PoW, another component is required, namely, the **nonce**, which denotes actual *proof of work*. Each participant of the protocol puts forth a trial-and-error approach to find a suitable value for **nonce** under the PoW protocol specification. The validity criteria is based on the value of the hash found. It should be less than a given value; in the Bitcoin protocol, it is denoted as the difficulty parameter. In PoS setting, there is no need of such parameter. This value is used in the hash link to connect two blocks, in such a way that each new block can only be issued by finding this value.

1.3 The Desirable Blockchain Properties

The blockchain data structure combined with the participants of the protocol should provide three properties as they were first formalized in [14]. Informally, these properties can be described as follows:

- **Common prefix**: The initial portion of two blockchains are the same; in other words, for every chain known to the participants of the protocol, if enough (newest) blocks are excluded, all the parties would find a *common* set of blocks.
- **Chain growth**: This property means that the data would be continued to be added to the chain; therefore, the data structure continues to grow.
- **Chain quality**: The basic assumption of a distributed system is that some of the participants of the protocol are controlled by a single adversary, which can influence the protocol by issuing malicious blocks, i.e., containing information that deviates from the protocol honest behavior. This property defines what is the safe ratio between honest and adversarial blocks.

A detailed discussion about the robust ledger definition in the context of the Bitcoin protocol can be found in [14]. For our purposes, we can summarize its basic properties as the following:

- **Liveness:** Once a transaction is included in the collection of entries, it is permanent and has its order fixed among others.
- **Robustness:** New transactions suggested by the participants are eventually included.

The robust ledger definition and the blockchain main properties in the context proof of stake differ little from the proof-of-work setting. However, the way to prove that protocol participants achieve such properties differ greatly since both paradigms rely on fundamentally different assumptions based, respectively, on "stake" and "work." In order to further investigate these differences and in preparation to introduce the proof-of-stake model in Sect. 3, we now review the common security concerns of relying solely in "work."

2 Security Considerations

The security of the protocol is crucially dependent on the proof of work produced by the players in a couple of ways. It certainly depends on the rate of block production and also the "quality" of its creation, i.e., honestly created, as one would expect given the properties outlined in Sect. 1.3. Here, we describe more concretely a few security, centralization, and environmental concerns common to PoW systems.

The Hash Race and Rewards: The protocol performed by the "miners" is critical for the maintenance of the security properties associated more directly with the blockchain data structure. However, the protocol characteristics are not enough to convince the nodes to lend their respective computational resources, aiming to keep the whole system. In combination with PoW, the nodes are incentivized by the distribution of rewards in the form of newly produced tokens registered on each new block. The PoW and reward combination builds a setting that the nodes perform a "hash race," when, on each new addition to the blockchain, the nodes dispute the rewards.

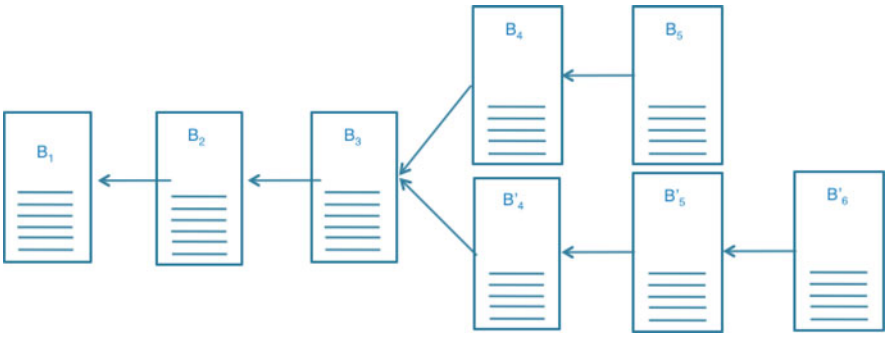


Fig. 3 Different sets of participants in the network may keep two different versions of the blockchain data structure. While one part may recognize the structure $B_1||\ldots||B_5$, others would rely on the $B_1||B_2||B_3||B'_4||\ldots||B'_5$

The Eventual Consistency and Fork Situation: The race between players in a distributed setting can naturally trigger the situation that a set of players agrees on a correct view of the blockchain, whereas a different set agrees on another view. This situation is commonly known as a "fork," as illustrated in Fig. 3. During this situation, the system is in a state of inconsistency with two competing blockchain views supported by two different sets of players.

This inconsistent state is designed to be temporary. The reason is that each set of participants will work and expand its own data structure, i.e., its view of the blockchain; however, the expansion rate would have different rates. Therefore, when the new block is produced, first by one of the sets of participants, as it is naturally expected, the fork collapses, i.e., all participants agree to either one of the two conflicting views as given by Fig. 4. The expectation of eventually solving the inconsistency state is based on the difference in block production, and this property is named *eventual consistency*.

The Sybil and 51% Attacks: So far, we have covered honest behavior of participants. In a distributed system, malicious participants can trigger inconsistent state of the overall system. In a known attack named Sybil attack, first formalized by Doucer [10], the adversary controls a high number of participants created artificially, the Sybil attack, aiming to disrupt the regular outcome of the system, which in our case, is the view of the blockchain. Previous classical Byzantine fault-tolerant systems, e.g., [5], have their fault-tolerant resistance based on the maximum number of *byzantine* participants. Critically, for a *permissionless* distributed system, i.e., participants can freely join the protocol, this is a practical attack. However, PoW-based systems were shown to fundamentally differ from this approach by not relying on the number of malicious players, but on their combined hash power. Intuitively, an adversary performing this attack would disperse its computational power while creating malicious nodes, rendering it not successful.

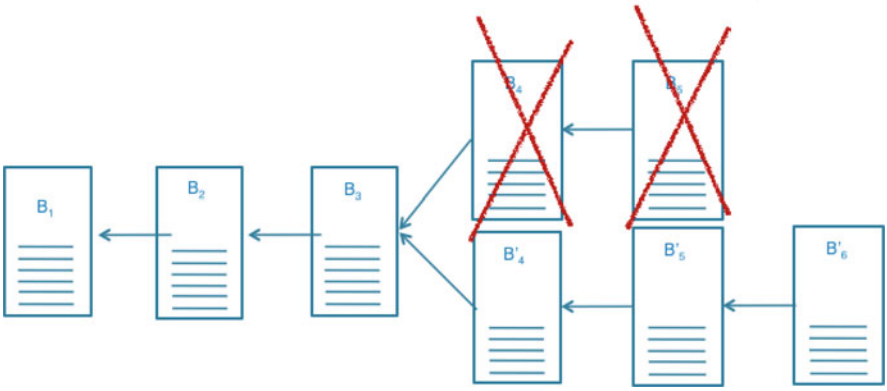


Fig. 4 By the time B'_6 is issued, the eventual consistency takes place, since all honest participants would converge to the $B_1||B_2||B_3||B'_4||\ldots||B'_5||B'_6$ structure. In other words, one of the branches of the fork will be discarded, and its contents will be lost

Creating new artificial participants is not a viable approach; however, somehow, gathering more hash power is a viable alternative option. The PoW/reward setting drives the participants to keep increasing the computational power in order to win the race among all the players as often as possible. The ratio of successful block production is directly proportional to the hash power of player with respect to the whole set of players. A known threshold in the adversarial hash power, as shown by Nakamoto [24], is half. An adversary who has more than half of the hash power can execute what is known as the 51% attack. Briefly, the attack is successful because even the combined hash power of all the honest participants is smaller than the malicious one. Therefore, eventually, the maliciously generated blockchain will surpass the honest one. More technically, the *chain quality property* of Sect. 1.3 cannot be preserved. In a fully matured PoW-based system, i.e., where the total hash power is large, the 51% is a high threshold which may be hard to achieve, although big mining pools could collude to surpass it. A more alarming case is for system with low combined computational power. In such systems, even a more modest adversary could join and single handedly overtake the whole system, given its lower threshold.

Hardware, Centralization, and Environmental Concerns: The side effect of relying on the computational power of the participants and establish the race between them, as described earlier, is the promotion of more efficient equipment to perform the PoW. Since the PoW introduction, there has been a line of improvements based on the development of hardware. The first nodes naively performed PoW on regular desktop computers and software. Given the financial gains of the rewards offered, more efficient methods were introduced. Nowadays, "mining" is performed in big "mining farms," where in a single facility thousands of ASICs, i.e., purposely designed chips, are employed to perform PoW:

- **First Generation:** CPU and regular desktop computers.
- **Second Generation:** Graphic processing units (GPU) were the next step, in terms of efficient and an off-the-shelf alternative for PoW.
- **Third Generation:** FPGA provided more tailored hardware in addition to efficient methods of cooling.
- **Fourth Generation:** ASICs introduced purposely designed chips, a cost-effective alternative, to mining.

The most efficient alternative for mining is based on ASIC, and in order to stay competitive, a miner should employ thousands of chips and build big facilities to host them, therefore limiting the number of player that can join the protocol. The high expenses required to perform the protocol have been leading to a greater concentration of players. In comparison to the early generations, fewer players afford to perform PoW efficiently. Thus, the protocol became more centralized significantly close to the threshold of the 51% Attack.

Yet another undesirable consequence of such intensive computation is the increasing environmental footprint left by the PoW design. The most widely known blockchain-based system consumes the equivalent of a country in terms on electrical energy, which raises concerns for future wide adoption of such systems.

Note that PoW design promotes the computation-intensive competition between nodes. As an intrinsic feature of the system, it is unavoidable that nodes pursue even higher computational rates, since it will allow them to perform ahead in terms of rewards. The more intensive the computation is, the more likely the node will be issuing a new block. The development of more efficient hardware is one strategy that leads to more centralization; consequently, more and more only well-funded player can afford to invest in more sophisticated hardware. As a summary, note that one approach is to depart of the computation-intensive routine that PoW relies on. This is the approach of PoS.

3 The Proof-of-Stake Model

The limitations of PoW, such as high energy consumption, increased interest in alternative designs to avoid Sybil attacks. The main idea is to avoid the "hash race" of the PoW paradigm, as well as any similar intensive computation.

The most prominent alternative to PoW is proof of stake (PoS). PoS relies on a mechanism akin to a lottery. Specifically, each user of the system holds some amount of tickets, called their "stake." Although stake can be interpreted in various ways, for simplicity, we define it as the amount of tokens a user owns.[1] Crucially, selecting a "lottery winner" at each turn is significantly less intensive and more efficient, compared to the repeated hashing of PoW.

[1] An alternative interpretation of stake is the amount of tokens multiplied by the time period during which a user owns the tokens.

The idea of PoS goes back to at least 2011.[2] The first system to employ stake instead of computational power was PPCoin, proposed by King and Nadal [20]. In this "pure" PoS system, the more stake, i.e., the more coins, a user has, the higher the probability that they will be selected to update the ledger, by issuing a new block. Given the distribution of stake, each user that holds at least one coin is eventually, i.e., after a large enough amount of time, selected. Therefore, decentralization follows naturally, as long as the user base is diverse and the majority of stake is not under control of a single user.

A crucial difference between PoW and PoS is the set of parties that participate in the consensus mechanism. In PoW, this set comprises of miners, i.e., parties with hashing power; in PoS, it comprises of "stakeholders", i.e., users of the system. Interestingly, in PoW, miners are not necessarily users, while also users are not required to control any hashing power. In contrast, in PoS, the two sets of parties, users and consensus participants, are identical. However, this design choice affects the security assumptions.

Intuitively, in order for the PoS system to be secure, each user should perform the required actions when needed. Abstaining, i.e., lack of action, is considered adversarial behavior, as it may damage the properties outlined in Sect. 1.3 and render the system insecure. For instance, if a user does not issue a block when needed, both *chain growth* and *chain quality* are hurt; in the extreme scenario, when no user participates, the system suffers from *denial of service*, as no blocks and no transactions are published.

Various proposals attempt to mitigate the abstaining problem. First, a reward scheme could incentivize users to participate via rewards, often financial in the form of tokens in the system. However, incentives still do not necessarily guarantee security, since a user may even act against their interest. Therefore, the main cause of concern is that the assumption that every token holder is available to act timely is not necessarily viable.

The Choice of the Block Issuer: Until this point, it is clear that the block issuer is chosen probabilistically, depending on their stake in the system. Now, we will describe concretely how a participant is selected from the whole set of users. In academic literature, such procedure is often denoted as a *lottery protocol*. In our case, a more accurate term is "weighted lottery protocol," since the probability of a user winning the lottery is weighted by their stake.

The whole set of participants execute the lottery protocol in rounds, i.e., *time slots*. Additionally, rounds are often grouped in *epochs*. Thus, in this general model, for every epoch, a well-defined number of time slots exist. For each round, the players perform a lottery protocol, which elects a "winning" token (Fig. 5). The user that owns this token is the *block issuer* and is eligible to produce the next

[2] We refer to the post by the user "QuantumMechanic" on the Bitcoin Forum: https://bitcointalk.org/index.php?topic=27787.0 [retrieved: December 2020].

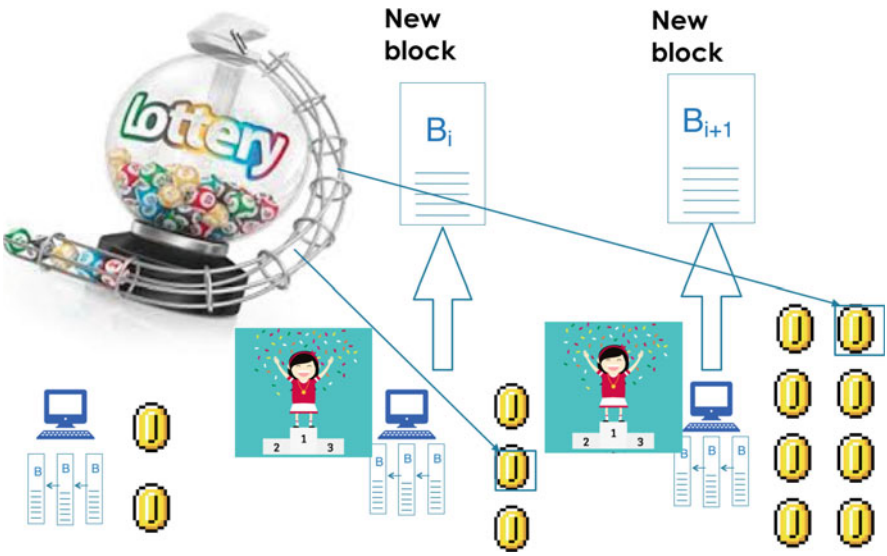


Fig. 5 The lottery protocol selects among all the tokens available in the system which one is the winner on the round. The chosen node is expected to issue the new block. Naturally, the user that holds more "stake" is expected to be chosen more frequently

block. Note that the probability of winning such lottery derives naturally from the amount of tokens each party owns.

We emphasize that the most critical part of this process is how to perform the lottery protocol securely. In other words, assuming the existence of malicious parties, under some well-defined network, the ledger that employs such lottery mechanism must preserve the properties outlined in Sect. 1.3.

Implementing the lottery protocol takes many forms, with varying computational and communication complexities. A common performance limitation stems from the number of participants. Given that, in PoS systems, every stakeholder should participate in the lottery protocol, the communication and computational burden among all stakeholders may prove impractical.

An alternative design, which aims to sidestep these efficiency concerns, employs *subselection*. Following this technique, each stakeholder can "vote" a representative, each vote weighted by the user's stake. The elected representative form a committee, which in turn performs the lottery protocol. Since the committee is typically smaller than the entire set of stakeholders, the lottery protocol is now performed more efficiently. In literature, this approach is often referred to as "delegated PoS" [22], where a user enable their chosen *delegate* to participate in consensus and produce blocks on their behalf. Consequently, the user can remain offline, and as long as the delegates are honest, the system's security may be ensured.

Real-World PoS Implementations: In practice, various systems implement different flavors of PoS. Pure PoS, being computationally intensive, is not a design of choice for any major blockchain system; instead, different types of subselection are employed. In their initial stages, systems like NEO[3] and Cardano[4] restricted the execution to a small set of servers, without the ability of stakeholders to participate, either directly or via representation. This is still the case for Algorand,[5] where the list of relay nodes is controlled by the Algorand Foundation and stakeholders cannot influence it. In some systems, this restriction was gradually eased, e.g., Cardano now allows each stakeholder to choose a delegate (called "stake pool"), which participates in PoS on their behalf. Although in Cardano there is no formal restriction on how many stake pools can participate, this is not the case for other systems. For instance, stakeholders of EOS[6] can choose any delegate of their choice, but only 21 such delegates participate at any point in time. Additionally, Cardano imposes no stake restriction on who can become a delegate; this is not the case in systems like Tezos[7] and Harmony,[8] where to become a delegate one needs to control a minimum amount of stake.

Additionally, different systems employ different reward schemes for stakeholders and participants/delegates. In the simplest case, as in Algorand, stakeholders are rewarded a fixed rate per unit of stake they own, while relay operators do not receive rewards. In other cases, e.g., Tezos, the delegate's reward depends on the amount of blocks they contribute to the chain, while the stakeholder's reward depends on their chosen delegate's performance and the operation fees charged by the delegate. Finally, some systems impose further restrictions on rewards to incentivize specific behavior; for example, Cardano employs progressive rewards with an upper cap, such that stakeholders are incentivized to produce a diverse set of representatives, which is of a specific size chosen by the protocol's designers.

4 Attacks on Proof-of-Stake Consensus

Given the earlier description of the PoS model implementing a consensus protocol and its fundamental difference from PoW design, it is natural that different attack strategies appear in order to disrupt the protocol. Three major strategies are as follows:

[3] https://neo.org.

[4] https://cardano.org.

[5] https://algorand.foundation.

[6] https://eos.io.

[7] https://tezos.com.

[8] https://www.harmony.one.

- **Grinding Attack**: The adversary targets the lottery protocol, from Sect. 3, in order to obtain extra advantage, in comparison to what its stake amount allows, in being selected as the block issuer.
- **Nothing-at-Stake Attack**: The attacker exploits a situation when it does not have real risk for maliciously performing the protocol. While the owner of the stake may worry in losing value of the stake, since the adversary does not have any stake, it is free to perform such attack.
- **Long-Range Attacks**: A few strategies are known for this attack; in general, the goal for the adversary is to somehow substitute the real chain, kept by honest users of the system, by one that it controls.

The above attacks are particular to PoS system. On the other hand, some of the attacks have similar equivalent in both paradigms. We start by describing them.

4.1 Similarities with PoW: 51% and Sybil Attacks

The PoS paradigm has similarities with PoW in the sense that similar attacks are known. A more concrete example is the 51% attack described in Sect. 2. The equivalent attack exists in the PoS setting. Whereas the main resource in PoW is the computational power, in PoS, it is the stake within the system. A participant who accumulates more than half of all the stake would, consequently, be chosen as block issuer more often than all other participants combined. In such case, this particular scenario means that a single user controls the majority of a currency, in the case of a cryptocurrency, which can be clearly seen in the ledger and also infeasible in the regular PoS. However, in the case of delegated PoS, entities as stake pools and exchanges, i.e., nodes that convert tokens between different chains, may get close to this threshold. The security in the presence of these entities is discussed in Sect. 7.

The stake of PoS also works as a deterrent against the Sybil attack in a natural way. Likewise, as already described in Sect. 2 for PoW, creating new identities within the system would not bring advantage to the adversary that aims to create more blocks or change the already created chain. By design, the identities are associated with the stake in the system; therefore, the more identities created by the adversary, the more spread the stake become. However, the combined amount of stake does not change; thus, the probability of being chosen as the block issuer remains unchanged.

4.2 Grinding Attack

In this class of attack, the adversary targets the lottery protocol in order to influence the choice of block issuer. More concretely, a malicious user aims to obtain an unfair advantage in order to generate blocks more often than its stake in the system

allows. Several PoS protocols introduce different designs to implement the lottery protocol as, for example, Ouroboros family of protocols [2, 6, 19] and Algorand [16] are based on verifiable random functions, key-evolving signatures, and committee selection.

4.3 Nothing-at-Stake Attack

An additional and equally important concern in the PoS setting is the "nothing-at-stake" problem [12, 23]. As opposed to PoW, in PoS, a player can easily create blocks that extend multiple parallel chains, an adversarial behavior which diverges from every PoS protocol's rules. Thus, an adversary may profit by attacking the system in this manner, even as they own a large amount of the very assets which they attack. This problem becomes more apparent in the cases of custodian services, which manage assets on behalf of their clients, but assume no financial risk themselves.

The work in [18] sidesteps this problem by introducing "exile" addresses, i.e., addresses excluded from the execution of the protocol. Section 5 describes the general framework of [18].

4.4 Long-Range Attacks

As shown earlier, the PoS paradigm fundamentally relies more heavily on cryptography tools in order to implement the lottery protocol in comparison to PoW-based protocols which relies in a scarce resource, i.e., computational power. Despite the dependence on the "stake," PoS-based protocols lack the guarantee that the accumulation of hash power provides in the PoW paradigm for the issuing of each block. The consequence is that the adversary can compute a malicious version of the blockchain data structure with ease. Thus, an attack vector is to present an "alternative history" to honest users, a maliciously computed blockchain. Such attacks [8] can be categorized as follows:

- **Simple Attack:** The malicious participant creates a competing chains that contains only its own blocks, i.e., the blocks it generated alone. The chain is composed only of its blocks and therefore can be easily manipulated.
- **Posterior Corruption/Bribery:** This attack is more sophisticated than the previous one, because instead of a one malicious participant adding only its own blocks the alternative chain contains also blocks generated by previously honest players that left their keys unprotected or turn them away when offered a bribery.
- **Stake Bleeding Attack:** Once again, the adversary keeps a malicious chain containing only its own blocks; however, this time, the attacker keeps an eye on the main chain and the transaction pool in order to select transactions that

would provide better fee sums. This allows the private chain to balance the stake share more favorably to the malicious participant, who eventually can overtake the real chain.

Each of them have countermeasures used in real systems that we describe later. For now, let us take a closer look on each of the earlier cited class of attacks.

Simple Attack: Recall that on PoS, on each run of the lottery protocol, a participant is chosen and therefore is expected to issue the new blocks. The malicious participant can create a blockchain data structure with a different sequence of blocks. Instead of honestly generated blocks, the maliciously generated chain is composed only of the adversary. Note that the adversary can create blocks on the same pace of honest users.

The adversary's strategy results in a chain composed only of malicious blocks, which competes with the real chain. This strategy works when nodes do not verify timestamps for each block. Therefore, in order to compute a chain with the same length, the adversary computes blocks in advance by forging timestamps. A longer chain would overtake the honest one, and since only the adversary adds blocks, it can manipulate the contents freely.

Posterior Corruption/Bribery: In this attack, a malicious participant will once again "rewrite history" by presenting an alternative chain. This time the adversary will start from an honestly generated chain where some of the previous (honest) participants no longer have stake in the system; however, the generated blocks are now part of the real chain.

It is fair to assume that when the participant exits the system , its previously safe keys may now be more easily exposed to threats. In particular, an adversary aiming to disrupt system can collect these now unused (and potentially unprotected) keys and use them to rewrite history. Note that each block is signed at the moment of its regeneration, and they replace the original block at will in their places in the chain, while skipping the blocks of the missing keys, i.e., the ones the adversary does not have.

In fact, the adversary may obtain the unused keys via corruption of the node (which may be unprotected), or the participant, who already has no stake in the system, could be compelled to sell them through a bribery offer.

Stake Bleeding Attack: This attack has the common feature that the attacker keeps a private chain composed only of its blocks, i.e., the blocks generated by itself, likewise the *simple attack*. The difference in this case is that in the public chain, the one being attacked, the adversary skips publishing blocks. The idea is that in the blocks of the private chain, the adversary will manipulate the contents in order to accumulate more stake.

From the start of the attack, the adversary would have a shorter chain in comparison to the public one. However, the adversary can choose freely the transactions to be added into its private blocks as long as they are valid, i.e., they are published in the honest chain, and are in the transaction pool, e.g., available to all participants. Both are easily observed by the attacker. In particular, the choice would

favor transactions that give the attacker a bigger fee sum per block. Despite of being a private chain, note that in this stage the share of the stake may already balance more favorably to the attacker. In the long term, the competing chain would indicate a faster growth of stake to the adversary than to the honest players. Eventually, the adversary would achieve most of the stake in its alternative history. Despite its feasibility, the attack is rather slow and requires an unusual high amount of stake to be performed as observed in [15].

Countermeasures: The early mentioned attacks are well known, and practical design decisions are already implemented in several protocols in addition to the *longest-chain rule* (which in the PoW paradigm is related to how much work a chain accumulated):

- **Checkpoint**: In this design decision, the nodes allow only the reorganization of the blockchain, i.e., change to a competing chain, up to only a limited number of blocks. Needless to point out that some sort of "long"-range attack are still feasible but not from the genesis block; thus, we could call them "short"-range attacks.
- **Key-Evolving Signature (KES)**: KES [17] is a public key primitive that the owner keeps the same public key, and while the secret key changes, it *evolves*. This feature comes at hand in the case of leakage of the evolved key, since it cannot be used to resign old blocks. An adversary that collects used keys from honest users could not be used much further in the past. More concretely, they cannot be used before it had evolved. This is the strategy introduced by the Ouroboros family of protocols [6];
- **Context-Aware Transaction**: The main idea [21] is to make the transactions of a block particular that block, and that block only. One way of adding such context, and therefore making it particular to a block, is to introduce the hash of the previous block into the transaction, similar to what is already done with the blockchain data structure itself. Another alternative is to add hash values of *time* when the information is available. This newly introduced types of transaction, once added, cannot be easily reused in a competing chain, since the hash values would be different.
- **Plenitude Rule**: This countermeasure was introduced by the Ouroboros Genesis protocol [2], and it is designed to detect a malicious chain competing with the honest one. A common feature of the early illustrated malicious chains, e.g., Figs. 6, 7, 8, and 9, is that in the initial stages of the chain the frequency of blocks is low. That is, the block production is sparse. Thus, the goal is that through the verification of the density of blocks, especially in the early stages, the honest users can identify the competing chain. However, even with this measure, it is necessary to consider the natural assumption that the attacker would have the minority of the stake.

Other approaches put forth strategies that are fundamentally different and do not closely affect the consensus protocol design. The PoS version of Ethereum [25] suggests to implement punishment to participants in the form of *financial slashing*.

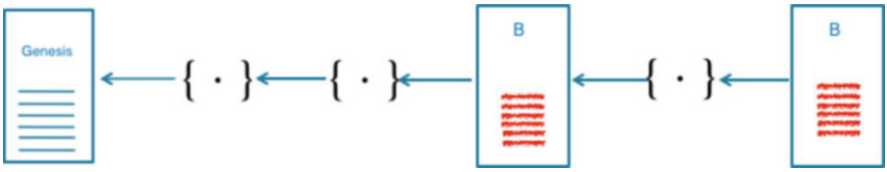


Fig. 6 Simple attack: the adversary keeps a hidden chain which contains only the blocks produced by the attacker, i.e., blocks in red, skipping the blocks from honest users. That is, the produced blocks contain only the hash value of the previous, i.e., malicious, block

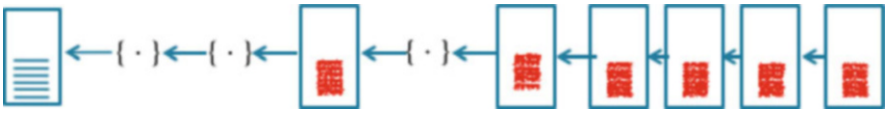


Fig. 7 Simple attack: eventually, the malicious chain becomes longer than the real one. At this moment, the adversary releases the chain to the public in order to compete with the real one

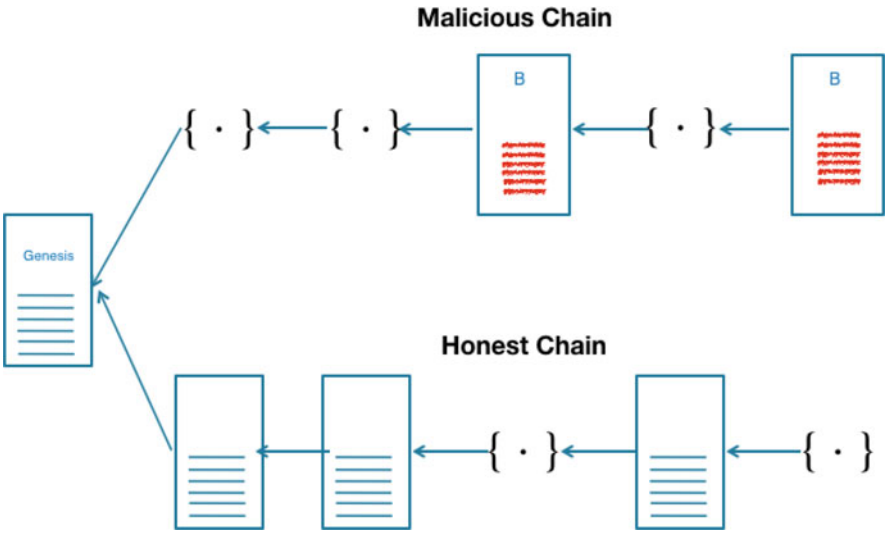


Fig. 8 Stake bleeding attack: the attacker, while keeping a chain solely with its blocks, manipulates the transactions within the blocks to maximize the rewards and fees collected on each block. This long performed attack aims to make the attacker to be selected more often by taking advantage of the accumulated stake to be chosen as block issuer more often

An equally novel approach is to rely on secure hardware solutions as suggested by Li et al. [23]. Secure platforms, i.e., trusted execution environment (TEE), like Software Guard Extensions (SGX) allow the generation and handling of the keys to be kept secret.

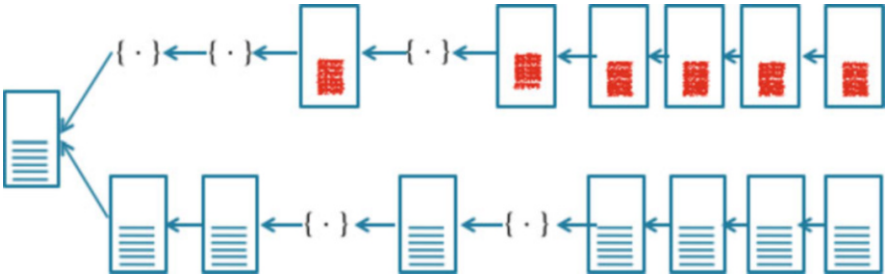


Fig. 9 Stake bleeding attack: in the long term, given the artificial accumulation of stake in the malicious chain, the adversary can release its branch in order to overtake the honest chain

So far, we covered the potential security issues and design decisions on PoS protocols in the level of the consensus protocol. However, a recent vector of attacks were observed in a higher level: the PoS wallet. We will study these issues next.

5 The General PoS Wallet

A generic definition of a PoS wallet was introduced by Karakostas et al. [18] and covers the basic requirements for a "good" PoS wallet and introduces the overall mechanism for *delegation* between players of the PoS consensus protocol. This general framework allows different modes of operation for wallets.

5.1 The Basic Requirements

The PoS wallet manages an account, which comprises of a set of addresses. Each address owns a nonnegative amount of cryptocurrency assets, and the account performs two basic operations:

- **Payment**: the act of issuing transactions and transferring assets between users
- **Staking**: the act of participating in the consensus protocol, either directly (e.g., issuing a block) or indirectly (e.g., delegating stake rights)

To perform each operation, an account holds some private pieces of information, e.g., private keys. Let $\mathcal{I}_p$ denote the information which is needed to perform a payment, i.e., to transfer assets owned by an account to a different account, while $\mathcal{I}_s$ denote the information needed to perform staking actions. If $\mathcal{I}_p$ and $\mathcal{I}_s$ are distinct, then participating in consensus protocol does not require the use of $\mathcal{I}_p$, which is reserved *only* for transferring funds (and vice versa), thus reducing the attack surface on each information piece.

A core difference of PoS, compared to PoW, is the ability to *delegate* staking rights. Specifically, a user A can give the rights over their stake to another user B; via this delegation process, B is eligible to participate in consensus protocol on behalf of A. In turn, B may delegate the staking rights of A's assets to a third party C, and so on, thus forming a "delegation chain." The work in [18] formalizes this delegation framework and provides a detailed discussion on PoS wallets, including the list of requirements for a general PoS system; a shortened version of this list is presented in Table 1.

5.2 Address Attributes for an Account

Each address is associated with a list of g *attributes*, i.e., objects that identify a property of the address. These attributes are categorized as follows:

- *Public* attributes are publicly identifiable and can used without any interaction with the account's owner.
- *Semipublic* attributes become public only when a transaction that spends some assets controlled by the address is issued.
- *Private* attributes never become public.

Examples of such attributes are: (i) Public: the hash of the payment information $\mathcal{I}_p$ (ii) Semipublic: a public payment key, e.g., if $\mathcal{I}_p$ is a public/private key set (iii) Private: a private payment key

More concretely, we assume that $\mathcal{I}_p$ and $\mathcal{I}_s$ are implemented as two key sets. Specifically, $\mathcal{I}_p$ is the payment key set $(\mathsf{vkp}, \mathsf{skp})$, where vkp is the public key and skp is the private key, whereas similarly $\mathcal{I}_s$ is the staking key set $(\mathsf{vks}, \mathsf{sks})$.

Address Identification: Each address also contains the recovery tag ht, a public parameter which allows the identification of addresses. Recovery is a process that relies only on the master key of the wallet, so the wallet should be able to recover an address by *only knowing its payment key*. Therefore, the tag links the address to the attribute list, without revealing the semipublic attributes, e.g., the payment key. During recovery in the simplest setting, the wallet computes all keys that it can control and uses them to compute the recovery tags.

An alternative design is searchable recovery. This design builds on the premise of searchable encryption [3]. Specifically, a user who possesses the wallet's master key can use it to search all addresses that appear in the ledger and recognize the ones that belong to the wallet. This scheme uses an instance of a semantically secure symmetric encryption scheme, which is comprised of the algorithms $\langle \mathsf{Enc}, \mathsf{Dec} \rangle$. To compute a recovery tag, the wallet computes the output of Enc, using the master key as the encryption key and the address's payment key as the plaintext. During the recovery phase, the wallet parses all addresses in the blockchain and decrypts the tag in each one. If the output is a well-formed key, then the address belongs to the wallet. Interestingly, since the output of the encryption function Enc is by

Table 1 High-level requirements for a general proof-of-stake wallet [18]

Requirement	Description
Address nonmalleability	Given an address α (and possibly also transactions issued by that address), an attacker should not be able to construct a different address α' that shares the same payment information $\mathcal{I}_p$ as α
Address uniqueness	For any two addresses α, α', it should hold $\alpha = \alpha'$ only if they are associated with the same information $\mathcal{I}_p$ and $\mathcal{I}_s$
Short addresses	An address should be as short as possible, under the restriction that it satisfies also all other requirements
Multiple types of addresses	It should be possible to categorize each address by different types, depending on the subset of basic operations that it supports; for example, an address of a specific type may not be allowed to participate in consensus (as briefly mentioned in Sect. 4.3)
Multiple device support	An account has the capacity to exist on multiple devices that share *no joint private state*
Address recovery	Given the payment information $\mathcal{I}_p$ of every address that it manages, for every address α that exists in the public ledger, the account should have the capacity to identify whether it can issue transactions from α
Privacy and unlinkability	Given any two distinct addresses $\alpha \neq \alpha'$, it should be infeasible to identify whether α and α' are managed by the same account
Account master key	Given an information piece $\mathcal{I}$, which is called the account's "master key" or "seed," a user can generate $\mathcal{I}_p$ and $\mathcal{I}_s$ for every address managed by the account
Staking and payment separation	If an attacker obtains $\mathcal{I}_s$ of an address α, then it should not be able to issue payments for α (and vice versa for $\mathcal{I}_p$ and staking actions)
Payment key information safety	Given an address α, if no payment is issued for α, then no information about α's $\mathcal{I}_p$ should be public, except (possibly) its hash
Key Exposure mitigation	If $\mathcal{I}_s$ is compromised, then the account's rightful owner should be able to transfer the staking rights of all assets it manages to a new staking information $\mathcal{I}_s\prime$
Cost-effective delegation	Delegating the staking rights of an account's assets to a third party should be as cost-effective as possible, under the restriction that all other requirements are also satisfied
Chain delegation restriction	There should be an upper limit on the size of the chain delegation of each asset, i.e., the number of times that the staking rights of an asset can be re-delegated
Delegation verification	Delegation should be publicly verifiable

definition random in the domain of Enc, then the recovery tag generation acts as a pseudorandom function.

On a high level, a wallet should enable a stakeholder to issue payments, participate in the PoS protocol, or delegate their stake. Therefore, from now, we continue looking into these main actions. How security of the overall PoS system is affected by delegation will be discussed, next, in Sects. 6 and 7.

5.3 Payment Transaction

Before describing the payment action, we should first introduce a general, and more detailed, model for a PoS system. Such system comprises of various parameters and functions, which abstract the system's inner workings and allow to handle it in a black box manner. At any time slot, each of the following properties is parameterized by the execution of the PoS protocol up to that point:

- $F_\theta(\cdot)$: a function that returns the PoS assets θ that a given address owns, i.e., an address's stake
- $F_{\Phi,tx}(\cdot)$: a function that computes the fees Φ that are required to publish a given transaction on the ledger
- Φ_{reg}: an integer which defines the cost of registering a new stake pool on the system
- α_{reg}: a special address, which is used as the recipient in transactions that register a stake pool
- $F_{PoS,player}(\cdot)$: a function that outputs the address α_{PoS}, the owner of which is eligible to participate in the PoS protocol

Payment is the act of transferring a set Θ of PoS assets from a sender's address α_s, i.e., the payment's "issuer," to a receiver's address α_r.

A wallet initiates a payment by first constructing the object $tx = (\Theta, \alpha_s, \alpha_r, m)$, where m is a special field that contains the transaction's metadata, e.g., the number of fees or, as we see next, a staking-related certificate. In the case of *UTxO* systems, like Bitcoin and Cardano/Ouroboros [2, 6], the metadata also contains a "change" address α_c, which receives the assets owned by α_s and which are not in Θ. In the case of *account-based* systems, like Ethereum, the metadata contain the number of transactions that were previously issued by α_s; with hindsight, this will serve as a countermeasure to replay attacks (discussed in Sect. 7.2).

Next, the wallet retrieves the balance Θ' of α_s via the function $F_\theta(\alpha_s)$. Additionally, it calculates the payment transaction's fees $\Phi = F_{\phi,tx}(tx)$. If the address α_s has enough balance to cover both the payment and the fees, i.e., if $(\Theta \cup \Phi) \subseteq \Theta'$, the wallet generates the object (tx, σ), where σ is the signature of tx generated by the key skp of α_s, and publishes it on the ledger.

5.4 Stake Pool Registration

Any party that wishes to participate in consensus protocol needs to create a stake pool. Stake pools are the entities among which consensus is run and which are eligible to extend the chain by producing new blocks. Therefore, each stakeholder can either create a self-managed stake pool or delegate their stake to an existing pool, as will be shown in the following section.

Every stake pool is identified by a staking key, $(\mathsf{vks}, \mathsf{sks})$, which needs to be registered in the system. This key pair is created by the wallet following the standard key generation process.

The registration of a stake pool is performed via a certificate. The registration certificate $r = (\mathsf{vks}, m)$ contains the pool's key and metadata m; the pool's metadata may contain information such as the name of the pool's operator, the pool's participation fee, etc. The certificate r is signed by the pool's private key, sks, and is published on the ledger via a special payment transaction.

In order to construct the transaction that publishes r, the wallet retrieves the registration fees Φ_{reg} and the special address α_{reg}. Subsequently, it chooses an address α_s, which it owns, and creates the transaction $tx = (\Phi_{reg}, \alpha_s, \alpha_{reg}, r)$. Therefore, tx is a transaction that transfers Φ_{reg} assets from α_s to α_{reg} and contains the stake pool registration certificate in its metadata. This transaction is then signed and published via the process described in Sect. 5.3.

5.5 Stake Delegation

As described earlier above, a stakeholder who does not wish to directly participate in consensus can delegate their stake rights to a stake pool. Stake delegation is also achieved via certificates, via a process similar to the staking pool registration.

Each address α is associated with a staking key vks_s, which controls the staking rights of all assets owned by α. A certificate that delegates the staking rights of address α is a tuple $d = (\mathsf{vks}_s, \langle \mathsf{vks}_d, m \rangle)$. The first element, vks_s, is the staking key associated with α. The second element, vks_d, is the staking key of the delegate, i.e., the staking pool that receives the staking rights. The third element is the certificate's metadata. As with stake pool registration certificates, d is signed by sks_s and published on the ledger in the metadata of a specially crafted payment transaction.

In line with the general requirements from Table 1, the addresses are divided into three types, depending on the way that they are associated with their staking key vks:

- **Base address**: The address contains, for example, $\mathcal{H}(\mathsf{vks})$ for a hash function $\mathcal{H}$.
- **Pointer address**: The address contains a pointer to a certificate published on the ledger, which is either in the form $(\mathsf{vks}, \cdot, \cdot, \cdot)$, in case of delegation certificate, or $(\mathsf{vks}, \cdot, \cdot)$, in case of stake pool registration certificate.

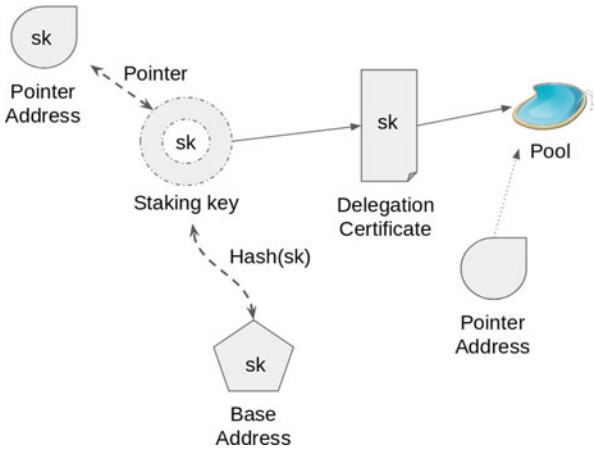


Fig. 10 A representation of the delegation mechanism. Base addresses are linked to a key via a hash values, whereas pointers may point to either a staking key sk or a pool directly. Note the tracing lines mean that the addresses are related to the staking key

- **Exile address**: The address is not associated with any staking key.

The implementation of a pointer depends on the system's characteristics, e.g., it might be an index (of a transaction in the ledger), the hash of a transaction, etc.

When a delegation certificate d is published by some key vks_s, *all* associated addresses are re-delegated accordingly. This applies to both base addresses, which contain the hash value of the key, and pointer addresses, which point to some old certificate issued by the key. For example, suppose a pointer address α_p that points to delegation certificate $(\mathsf{vks}_s, \mathsf{vks}_d, m, \sigma)$. Initially, α_p is associated with vks_s. Following, vks_s publishes a newer certificate $(\mathsf{vks}_s, \mathsf{vks}'_d, m', \sigma')$. Now, all addresses associated with vks_s, including α_p, are delegated to vks'_d. Figure 10 illustrates delegation for various types of addresses.

This design mechanism for delegation of stake allows two types of procedures, depending on when the delegation certificate is published. Moreover, it allows "consecutive" staking, or "delegation chain." From now, we look into these two topics.

5.5.1 Heavy and Light Delegation

Delegation certificates come in two forms, depending on how they are published:

Heavyweight certificates are published directly on the ledger via a special transaction, according to the process described in the previous section. A heavyweight certificate is identifiable by an *index*, which represents the position of the certificate in the ledger, and is constructed as the tuple (b, x, c). Specifically, b identifies the position in the ledger of the block B which contains the certificate, i.e., how many blocks occur in the blockchain between the genesis block and B. The value x identifies the transaction τ which contains the certificate, i.e., the position of τ in the list of all transactions included in B. Finally, c identifies the delegation certificate in

the metadata of τ, i.e., its position in the list of (possibly many) certificates that τ publishes. As described above, the index also serves as the pointer, which identifies the key that controls the staking rights of a pointer address.

Lightweight certificates are constructed locally and are not published by default. Their utility is to quickly and cost-effectively re-delegate the staking rights from a key vks to a newly created key vks'; with hindsight, lightweight will prove useful to implement the "hot/cold" mode of wallet operation. Specifically, a lightweight delegation certificate is an object similar to a heavyweight certificate, with one additional parameter; the metadata of a lightweight certificate also contain a (integer) "counter." A lightweight certificate becomes public when the key vks, which issued the certificate, is required to participate in the PoS protocol (more concretely described in Sect. 5.6). At that point, the new key vks' can participate on its behalf, by publishing the lightweight certificate in the new block. As will be described next, if multiple conflicting blocks are created from different (but valid) lightweight certificates, the integer counter in the certificate is used to choose a chain.

5.5.2 Delegation Chains

Chain delegation occurs when a staking key, which has been delegated the staking rights of some assets, issues a delegation certificate, transferring the staking rights to a new key. Whether chain delegation is acceptable and to what degree, i.e., how many links can exist in a delegation chain, is a protocol-specific rule. Additionally, if chain delegation is permitted, a certificate may explicitly prohibit re-delegation, e.g., by setting a flag "allowChain" in its metadata.

To provide further intuition around delegation chains, consider the following example. Suppose two certificates Σ_0 and Σ_1. Now, assume that Σ_0 delegates from the key vks_0 to a key vks_2 and states "allowChain" $= true$, while Σ_1 delegates from vks_1 to the same delegate key vks_2 and states "allowChain" $= false$. Following, a third certificate Σ_3 is published, from key vks_2 to (delegate) key vks_3. If chain delegation is permitted, then it is valid, but it only acquires the staking rights of all addresses associated with vks_0.

More concretely, a *delegation chain* is a list of certificates $[\Sigma_1, \ldots, \Sigma_i]$ such that, for each certificate Σ_j, $1 < j \leq i$, it holds that $\Sigma_{j-1}[\mathsf{vks}_{delegate}] = \Sigma_j[\mathsf{vks}_s]$. We say that vks *is delegated via a chain* $[\Sigma_1, \ldots, \Sigma]$ if $\Sigma[\mathsf{vks}_{delegate}] = \mathsf{vks}$.

Evidently, the validity of a delegation certificate is computed following a number of rules: First, its signature should be valid. Second, stake pools are typically not allowed to re-delegate their stake, so no registration certificate should exist for the key that issues the delegation certificate. Third, a certificate that re-delegates stake is valid only if chain delegation is permitted by the system. Observe that although a delegation certificate may be valid at the time of publishing, it can become invalid at a later point. For example, if a registration certificate is published for some key vks, all delegation certificates issued by vks are invalidated. Therefore, a registration

certificate acts as both a registration mechanism for vks and a revocation mechanism for all delegation certificates issued by vks.

5.6 PoS Protocol Participation

Participation in the PoS protocol consists of publishing specially crafted, protocol-related signed messages. Specifically, participation comprises of *block generation*, i.e., the act of extending an existing chain with a newly created block. At any point in the execution, the function $F_{PoS,player}$ outputs an address α_{PoS}, which is responsible for issuing a block. A block is a tuple (vks, m); $(\mathsf{vks}, \mathsf{sks})$ is the key which signs the block, while m is the block's payload, i.e., tree of transactions, hash values as described in Fig. 1. As with delegation and stake pool registration, the PoS wallet issues a signature on a new block using the staking key of a corresponding address.

In the plain case, a block is valid only when it is signed by the staking key that corresponds to α_{PoS}, the address whose participant is eligible to join the protocol. However, this is not necessarily the case under delegation, which requires more complicated validity rules. Next, we describe these rules, regarding the validity of blocks and chains, as well as the rules that pertain to chain delegation, i.e., the re-delegation of delegated stake.

Block Validity: A block B which is valid for a chain C can be appended to it. At any point in the execution, a user holds a local chain, C, and retrieves the address α_{PoS}, which is elected by the PoS protocol to extend C; α_{PoS} is associated with a staking key vks_{PoS}. Therefore, the user that controls α_{PoS} is eligible to produce a valid block, either directly or indirectly (via delegation).

Assume a block B, which is signed by a key $(\mathsf{vks}, \mathsf{sks})$. If $\mathsf{vks} = \mathsf{vks}_{PoS}$, then B is valid for C. Otherwise, if a heavyweight delegation certificate, from vks_{PoS} to vks, is published in C, then B is valid for C. Otherwise, if B contains a lightweight delegation certificate from vks_{PoS} to vks, then $\mathcal{B}$ is valid. Finally, if vks is neither of the above holds, i.e., if the block's signature is not produced by the elected or a delegate key, then B is invalid for C.

Chain Validity: Consider that $C||B$ denotes the concatenation of a chain and a block and $head(C)$ denotes the latest block in a chain C. The validity of a chain is computed iteratively. On the base step, the empty chain $C = \epsilon$ is valid. Following, given a chain $C = C'||B$, if C' is valid and B is valid for C' (given the previous block validity rules), then C is valid.

Chain Decision: At each point in time, a user chooses a single chain among the set $\mathbb{C}$ of all valid chains that are available on the network following a list of rules: First, the user computes the subset of $\mathbb{C}$ which consists of the longest valid chains. If a single longest chain exists, then it is chosen. Otherwise, if multiple, equally

long (with the largest length) chains exist, then the user employs the following tie-breaking rules.

Assume two chains C_1, C_2 in the subset of longest chains, such that $head(C_1)$ and $head(C_2)$ are signed by the keys $(\mathsf{vks}_1, \mathsf{sks}_1)$ and $(\mathsf{vks}_2, \mathsf{sks}_2)$, respectively. First, if vks_1 and vks_2 are both delegated via heavyweight certificates, then the latest delegation takes effect; e.g., if the delegation to vks_1 occurred before the delegation to vks_2, then C_2 is chosen. Second, if only one of the two keys is delegated via a heavyweight certificate (while the other is delegated via a lightweight certificate), then the chain corresponding to the heavyweight delegation is chosen. Finally, if both keys are delegated via lightweight certificates, then the certificate with the higher counter takes effect; if both have the same counter, then the user chooses the chain that was first observed on the network. This comparison is performed between all chains in the subset, so the longest chain that satisfies all tie-breaking rules is eventually chosen. We remark that the above rules allow only a single level of delegation, although this can be easily extended to any bounded size of delegation chains.

5.7 Wallet Modes of Operation

A PoS wallet can be executed in different modes, depending on the level of safety, security, and privacy the users want to ensure.

5.7.1 The Regular Mode of Operation

The most straightforward way to run a wallet is in the *regular mode*, illustrated in Fig. 11. This wallet is bootstrapped using a base address α_0, the stake of which is managed by a staking key $(\mathsf{vks}, \mathsf{sks})$. After the first transaction is received, which pays some amount of funds to α_0, the user performs staking actions by using $(\mathsf{vks}, \mathsf{sks})$. Therefore, it creates a delegation certificate Σ, which delegates to $(\mathsf{vks}, \mathsf{sks})$. As an enhancement, Σ can be given to the sender of that first transaction, so that it is published along with the first assets that are sent to α_0. The subsequent addresses are pointer addresses that point to Σ; thus, all addresses are managed by $(\mathsf{vks}, \mathsf{sks})$. When the user wishes to delegate to a staking pool identified by the key vks_P, the wallet creates and publishes a delegation certificate Σ_d which delegates from vks to vks_P.

5.7.2 Offline Wallet with Cold Staking

The cold/hot wallet technique is a well-known procedure to improve the security by keeping the payment information offline. An offline wallet is stored in an offline device, e.g., on paper. This wallet is rarely accessed for payments, so its funds are

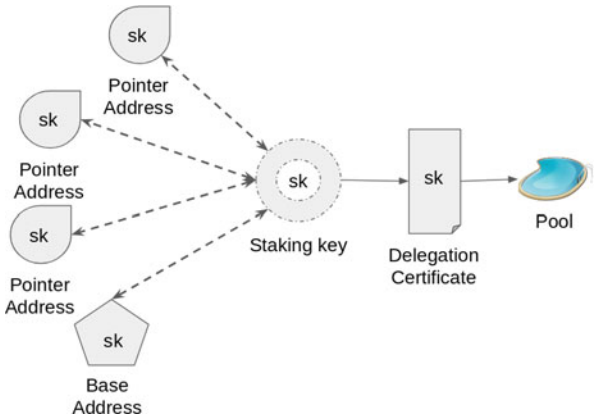


Fig. 11 The regular wallet mode manages the pointer addresses via the staking key and creates a delegation certificate Σ_d to the stake pool

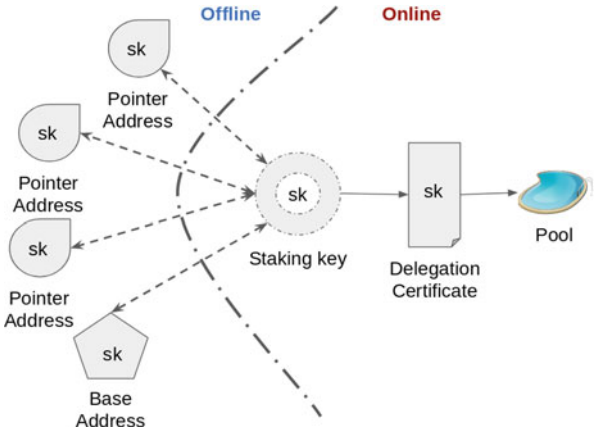


Fig. 12 The regular wallet mode with cold staking. The staking key is kept online, while the payment key in the base address is kept offline

rarely spent. However, it regularly performs staking actions, like delegating stake. It is, thus, bootstrapped similarly to the regular user wallet, i.e., by creating a base address α_0, a certificate Σ, and pointer addresses as above. The payment keys are stored offline, in a secure device, whereas the staking keys are managed as follows:

- For *basic security*, the staking key $(\mathsf{vks}, \mathsf{sks})$ that manages all addresses is stored online, so the user issues staking actions while the payment keys remain offline, although in the case sks is compromised, the user needs to access the payment keys and move the funds to new addresses with a new staking key.
- For *enhanced security*, the wallet creates a certificate Σ' that delegates from vks to vks_h, which is a "hot" key. After Σ' is published, the user stores $(\mathsf{vks}, \mathsf{sks})$ offline and $(\mathsf{vks}, \mathsf{sks})_h$ online, in order to stake for the wallet's addresses. In case $(\mathsf{vks}, \mathsf{sks})_h$ is compromised, the $(\mathsf{vks}, \mathsf{sks})$ can be used to issue Σ'', which delegates to a new "hot" key.

Figure 12 illustrates the regular mode of operation with cold staking.

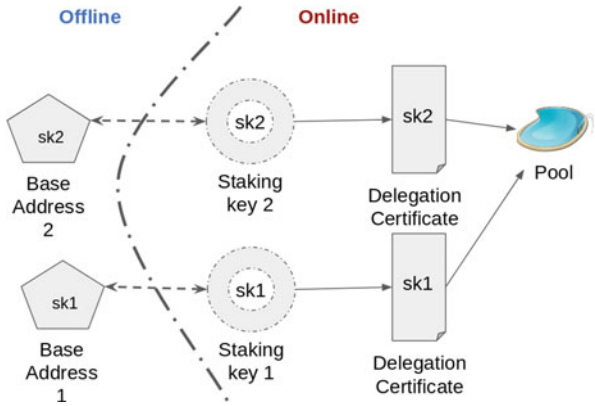


Fig. 13 Enhanced unlinkability with cold staking. For each new re-delegation, the wallet creates a new certificate

5.7.3 Enhanced Unlinkability

A wallet with enhanced unlinkability maintains the best level of privacy possible. Each address's stake is managed by a unique staking key, so, when changing the delegation of all addresses, the wallet creates a certificate for each. Depending on the level of security, this is achieved as follows:

- **Online:** The wallet is stored online and creates only pointer addresses, which point directly to a stake pool's key. In order to re-delegate, the wallet creates new pointer addresses and moves the funds from the old to the new addresses.
- **Offline:** The wallet's payment keys are stored offline, so the user cannot move funds as before. Now, the wallet creates only base addresses, each managed by a unique staking key, and keeps the staking keys online. In order to re-delegate, the wallet publishes a new certificate for each address.

Note that the same wallet manages all the staking keys which are part of the same account. Figure 13 illustrates the enhanced unlinkability modes with offline payment keys.

5.7.4 The Stake Pool's Wallet

A stake pool's wallet completes only staking actions; therefore, it creates the staking key pair $(\mathsf{vks}, \mathsf{sks})_P$ which is managed as follows, depending on the desired level of security:

- For *basic security*, $(\mathsf{vks}, \mathsf{sks})_P$ is online. It is used directly for staking, and in case of compromise, the wallet creates a new staking key and alerts the users to re-delegate to the new key.
- For *enhanced security*, $(\mathsf{vks}, \mathsf{sks})_P$ creates a lightweight certificate Σ_l, which delegates to a "hot" key vks_{Ph}, and is then stored offline, while the pool uses $(\mathsf{vks}, \mathsf{sks})_{Ph}$ and Σ_l for staking. If $(\mathsf{vks}, \mathsf{sks})_{Ph}$ is compromised, the wallet

creates a new key $(\mathsf{vks}, \mathsf{sks})'_{Ph}$ and a new certificate Σ'_l, which delegates from $(\mathsf{vks}, \mathsf{sks})_P$ to vks'_{Ph}, and uses them for staking after.

The general framework for the PoS wallet defines a mechanism to construct the delegation system and also introduces a method to add attributes to the address strings. The list of attributes in an address may open the protocol to a new attack vector. This new potential threat is the topic of the next section.

6 The Address Malleability

The complicated nature of PoS addresses gives rise to novel attacks, such as *address malleability*. This attack is similar to malleability notions that have been explored in existing literature [9]. Address malleability is observed in relation between payment and staking keys and possibly other attributes as described in Sect. 5.2. Specifically, this family of malleability attacks enables an adversary to construct addresses on behalf of honest users, which may correspond to honest payment keys but adversarial staking keys. Such addresses are called *forgeries*. In order to ensure the PoS protocol's security, it is crucial to prevent an adversary from constructing forgeries or manipulating honestly generated addresses.

A forgery is successful if an honest wallet accepts it and can spend its assets. Consequently, it fails if it cannot receive assets or its funds are unspendable. A malleability attack could take place by intercepting honestly generated addresses, which are exchanged between users, and replacing them with forged addresses on the fly. As long as the adversary remains unnoticed, it may be very profitable to mount a broad, network-level attack and artificially inflate their delegated stake. The following sections describes the threat model and different levels of malleability attacks.

6.1 Adversary and Attack Models

The goal of the adversary is to inflate their stake in the system by exploiting address malleability. In other words, the adversary attempts to control the stake of an address without necessarily controlling its payment key, in such way that the honest owner of the address does not easily recognize that such stake "shift" has taken place, unless actively looking for it.

We note that address malleability is adjacent to, though potentially more severe than, transaction malleability in Bitcoin [1, 7]. Using the latter, an attacker could modify a transaction and changes its identifier (i.e., its hash), such that the issuer was led to believe that the original transaction is unconfirmed. Although this attack may lead to loss of funds (e.g., as claimed in the Mt. Gox incident [7]), it does not compromise the ledger's consensus mechanism, due to the decoupling of the mining

and the transaction processes in PoW. In contrast, address malleability directly affects a PoS ledger's security, since an attacker can artificially inflate their stake to the extent of (potentially) violating the honest majority assumption in terms of stake as mentioned in Sect. 4.1 and later more formally described in Sect. 7.2.

There exist numerous use cases of adversaries that might try to exploit this ability. One such adversary may attempt to control the majority of the stake in a cost-effective manner, i.e., without having to actually buy or mint the assets. A more sophisticated scenario that is thoroughly described in Sect. 7.2 is a malicious stake pool that artificially increases the amount of rewards that it receives, under the natural assumption that the reward amount depends on the stake percentage that the pool is delegated. In both cases, the adversary aims to control as much stake as possible without being noticed by the honest user. Figure 14 showcases the malleability attack of a malicious stake pool leader in a Man-in-the-Middle attack and forges an address.

The stake pool use case becomes even more pertinent as the incentives mechanisms for a number of PoS systems mandate that larger stake pools receive greater rewards [4]. In such environment, it might be very profitable for a stake pool leader

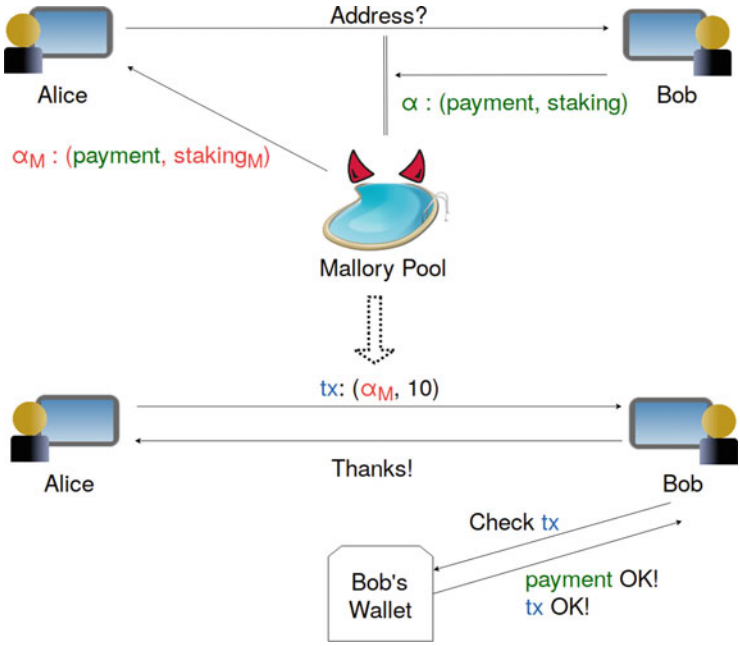


Fig. 14 A malleability attack by a malicious stake pool leader. Mallory intercepts Bob's address and replaces it with a forgery with the same payment key but an adversarial staking key. When Alice makes a payment and if Bob's wallet only checks the payment key, then it fails to identify the forgery

to mount a broad, network-level attack to exploit malleability and artificially inflate the delegated stake.

There can be two types of adversaries, depending on the information access, although, in all cases, the adversary does not control the signing payment key skp:

- **Network Adversary:** Access to the ledger's contents and the addresses produced by an honest party
- **Targeting Adversary:** Access to the same information as the network adversary, as well as the semipublic attributes, as described in Sect. 5.2, of the addresses of the "victim," i.e., the honest user for which it attempts to produce the forgery.

These two classes of adversary are illustrated with an example. Assume a company C that receives regular payments in cryptocurrencies. For security reasons, C stores the private keys of its addresses on an offline server, whereas the public keys are stored online, in order to easily compute and share new addresses with its clients. An adversary that pretends to be a client is a network adversary, i.e., may access some of C's addresses. Instead, an adversary that infiltrates the online server and accesses the public keys is targeting one. If the system does not protect against targeting adversaries, then malleability may be used to mount a covert attack, i.e., to obtain the staking rights of C's assets without raising flags.

6.2 Classification of Malleability Attacks

Malleability attacks range from *full malleability*, i.e., when there is no inherent protection, to *nonmalleability*, i.e., where an adversary cannot forge any address without access to its *private* attributes. Defining these levels allows the construction of different address schemes, which are suitable for the varying needs of real-world projects. For example, a fully malleable address is typically short, so an application focused on performance rather than security would opt for using it. However, a security-oriented project would rather aim for the highest levels of malleability protection.

Malleability levels are distinguished based on the following properties and are described briefly next:

- **Adversarial Types**: whether the adversary is the network or targeting types, as described above
- **Self-Verification**: whether a wallet can recognize a forgery for one of its own addresses (not necessarily for addresses of other wallets)
- **Cross-Verification**: whether a wallet can recognize a forgery for *any* address

Typically, malleability is closely related to the size of addresses. In order to better estimate them and compute their lengths, we assume that the employed hash function is SHA256 and the signature scheme is ECDSA on secp256r1, standard cryptography primitives for signature and hash function.

From now, we present the concrete constructions for the addresses with four levels of malleability as identified in [18].

Level 1, Full Malleability: Fully malleable address schemes enable forgeries from all adversaries. Fully malleable addresses are typically constructed as the concatenation of the hashes of the payment and staking keys: $\alpha = \mathcal{H}(\mathsf{vkp})||\mathcal{H}(\mathsf{vks})$. Evidently, given α, an adversary can replace $\mathcal{H}(\mathsf{vks})$ with $\mathcal{H}(\mathsf{vks}')$, where $\mathcal{H}(\mathsf{vks}')$ is some adversarial key. The length of fully malleable addresses is 64 bytes, i.e., the sum of two hashes. Figure 15 illustrates the instantiation of the malleable address scheme.

Level 2, Ex Post Malleability: Ex post malleable addresses prohibit forgeries from network adversaries, though allowing forgeries from targeting adversaries. Ex post malleable addresses are constructed as follows: $\alpha = \mathcal{H}(\mathsf{vkp}||\mathsf{vks}||ht)||\mathcal{H}(\mathsf{vks})||ht$. The first part of the address is the hashed root of its attribute tree; the attribute tree consists of the payment key vkp, the staking key vks, and a recovery tag ht, which is used by the wallet to identify the addresses it controls. Given only α, an attacker can no longer replace vks since, in doing so, the first part of the address becomes invalidated. However, the attacker can produce forgeries if it also knows the payment key vkp. The length of ex post malleable addresses is 96 bytes, equal to the sum of three hashes.

Level 3, "Sink" Malleability: Sink malleable address schemes protect against both network and targeting adversaries. Therefore, a wallet does not accept a forgery of one of its addresses, and all funds owned by a forgery are "burnt." Intuitively, if transactions form a graph, where the nodes are addresses and the edges are transactions, forgeries are "sinks," which trap funds, and no transaction can spend them. However, although a wallet can identify forgeries for its payment keys, it cannot identify forgeries for an address with a third party's payment key. To construct a sink malleable address, the payment key certifies the staking key as follows: $\alpha = \mathcal{H}(\mathsf{vkp})||\mathsf{vks}||\mathsf{Sign}(\mathsf{skp}, \mathsf{vks})$. Evidently, the only way for an attacker to produce a successful forgery is to forge a signature. The ECDSA signature is 512 bits, so the address is 128 bytes.

Level 4, Nonmalleability: Nonmalleable addresses offer the highest level of protection. Specifically, all forgeries are unsuccessful, and every party can identify whether *any* address is a (attempted) forgery. A nonmalleable scheme both uses a signature, as in the sink malleable scheme, and reveals the public payment key: $\alpha = \mathsf{vkp}||\mathcal{H}(\mathsf{vks})||\mathsf{Sign}(\mathsf{skp}, \mathsf{vks})$. Therefore, a wallet sends funds to an address only if the signature verifies with the provided key. The nonmalleable address's length is about 129 bytes (33 bytes for the payment key, 32 bytes for the hash, and 64 bytes for the signature). Figure 16 illustrates the instantiation of the nonmalleable address scheme.

Table 2 summarizes the above comparison. As expected, higher levels of malleability protection induce a higher-performance cost, in terms of address length. Additionally, the fully nonmalleable scheme reveals the public key, which

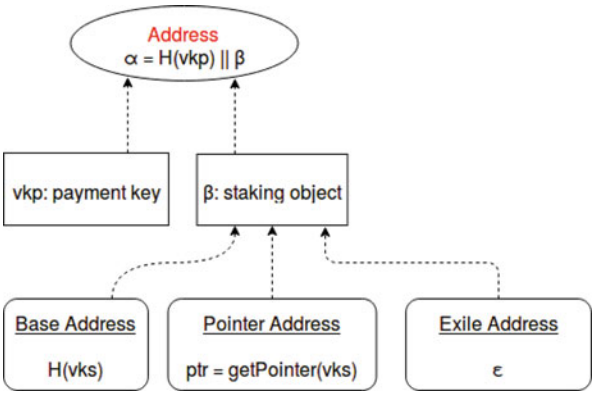


Fig. 15 The construction of a full malleable address of Level 1, for the three types of addresses, **base**, **pointer**, and **exile** as described in Sect. 5.5. The $getPointer(\cdot)$ function outputs the index of the staking key, whereas ϵ denotes the empty string

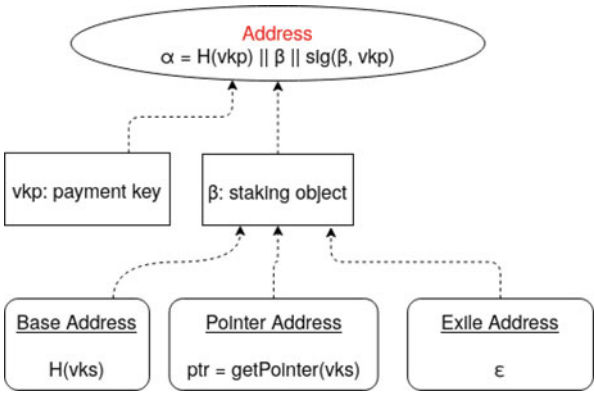


Fig. 16 The construction of a nonmalleable address of Level 4. As before, the $getPointer(\cdot)$ function outputs the index of the staking key, whereas ϵ denotes the empty string

Table 2 Comparison of malleability levels, from malleable to fully nonmalleable, based on the types of adversaries that can successfully forge an address, the ability of a wallet to identify a forgery, and the address length. We note that the nonmalleable scheme reveals the public payment key by default, whereas all other schemes reveal only its hash (and the public key is revealed only upon conducting a transaction)

Malleability level	Network protection	Targeting protection	Self-verification	Cross-verification	Address length (bytes)
1. Full	✗	✗	✗	✗	64
2. Ex post	✓	✗	✗	✗	96
3. Sink	✓	✓	✓	✗	128
4. Nonmalleable	✓	✓	✓	✓	129

may enable quantum attacks if a non-post-quantum secure signature scheme, like ECDSA, is employed. Alternatively, a nonmalleable scheme could use a noninteractive zero-knowledge proof-of-knowledge protocols, which would enable the same level of security without revealing the public key.

As we discussed in the beginning of this section, malleability attacks may alter the distribution of the stake; therefore, they may threat the security of PoS protocol.

In the next section, we investigate more formally the interplay between stake share and the presence of pool participants.

7 Security in the Presence of Stake Pools

The delegation framework described in Sect. 5 introduces new types of players which can be roughly divided in two major groups: First, there exist *stakeholders*, i.e., users who own assets and can delegate their respective stake, in order to go offline. Second, there exist *delegates*, i.e., parties which receive the staking rights over the assets that they do not own and participate in consensus on behalf of stakeholders. Thus, the rights over the stake of the consensus protocol are concentrated on the delegates set, i.e., the stake pools.

Here, we will take a closer look into this concentration of the stake rights and the overall security of system.

7.1 Theoretical Limit on Malicious Stake Share

As discussed in Sect. 2, the amount of stake that the adversary controls is a critical variable. Specifically, following the well-known theoretical limits, a PoS system's security cannot be guaranteed if more than half of the available stake is malicious, i.e., it is controlled by the adversary.

It is unavoidable that such upper bound exists; therefore, a security concern of any delegation-based system is to prevent that the accumulated stake of a participant, either via having the token and via receiving the delegation from others, is higher than 50% of the total stake. The ability of dynamically change the rights over the stake naturally leads to participants that possibly do not own any assets but act as delegates, namely, *stake pools*. Stake pools are purposely well-equipped, in terms of infrastructure, to perform the lottery protocol from Sect. 3, also named the *consensus protocol*, hence being attractive delegate choices for stakeholders.

Similarly, there exist *custodian* participants, i.e., parties that are well-equipped to manage assets. Such custodians, e.g., cryptocurrency exchanges, attract assets from several users who, although *legally* owning the assets, do not wish to manage the payment keys. Evidently, custodians also act as a form of delegates, since they entirely control assets on behalf of stakeholders. Thus, in addition to stake pools, custodians also naturally attract stake.

In the following section, we explore how the delegation mechanism affects the theoretical limit of security in a delegated PoS system.

7.2 Security in Stake-Pooled PoS

Here, we assume an underlying PoS protocol as described in Sect. 5. The honest stake threshold assumption for this protocol is τ, which is the minimum percentage of honest stake needed for the protocol to be secure. Typically, τ is set to $1/2 + \epsilon$ or $2/3 + \epsilon$ for some small positive number ϵ.

In a simplified model, all stake is delegated to P pools, each of potentially different size. Although this may strike as a strong assumption, this model allows a player to act as a pool and participate on their own. That is, among the P pools, there may be a single participant who manages its own stake and joins the consensus protocol on its own. Additionally, P_h out of the P pools are honest; these pools control, either directly or via delegation, ρ_h percent of the total stake. Similarly, let ρ_a denote the adversarial stake share.

As mentioned in Sect. 4.1, in order to analyze the security of a stake-pooled PoS variant, we need to focus on adversarial control over pools, rather than stake itself. This shift is necessary since, when a player delegates their stake, they effectively relinquish their staking rights. For example, if an honest stakeholder delegates their stake to an adversarial pool, then this stake becomes malicious in the stake-pooled setting, since it is controlled by the attacker. Thus, in this setting, the adversary compromises security by corrupting an appropriate amount of pools, such that the percentage of stake that honest pools control is less than τ, the "honest minimum threshold."

It is important to note that the above assumption does not imply that the adversary is required to corrupt a large number of pools. For instance, if a single pool controls $\rho_a \geq 1 - \tau$ of the total stake, an attacker can compromise security by only corrupting this single, albeit large pool. Corollary 1 formalizes this argument, while the following paragraphs discuss attacks that become prominent due to the introduction of stake pools and the certificates described above.

Corollary 1 ([18]) *The stake-pooled variant of a PoS protocol π is secure if $\rho_h \geq \tau$, where ρ_h is the percentage of the total stake controlled by pools which are managed by honest leaders and τ is the honest stake threshold assumption of π.*

Given the theoretical limits, the above argument holds and renders the system secure. However, it is still worth to discuss a couple of attack strategies.

7.2.1 Stake Pool and Sybil Attack

When stake pools participate in the PoS protocol, rather than the stakeholders themselves, the possibility of *Sybil attacks* [10] arises. Sybil attacks enable an attacker to create multiple, fake identities, in order to repeatedly participate in a protocol. In a stake-pooled PoS system, suppose that the adversary creates a large number of stake pools. In these systems, it is typically hard for honest players to identify an adversarial pool, as it could appear legitimate. Therefore, honest

stakeholders might be convinced to delegate to these pools, thus increasing the adversarial stake ratio. This is an inherent problem, since no form of external identification exists and an adversary can create a large number of staking keys and registration certificates (as described in Sect. 5.4).

A potential countermeasure is to require that the stake pool leaders own some minimum amount of assets. In the above model, this method is facilitated by introducing an extra field in the delegation certificate's metadata. This field identifies the addresses that the leader controls and thus their funds. These funds are committed to the pool so, as long as the funds are locked in these addresses, the pool's leader cannot use them in multiple pools. While this does not directly prevent a Sybil attack, it does bound the attacker's identity production capability.

7.2.2 Replay Attack

Another important consideration is *replay protection*. Replay attacks are prominent in account-based ledgers, where an adversary may republish an old transaction. For instance, suppose *Alice* sends x assets from her account A to *Bob*. After the payment is published, A controls $y = z - x$ assets, where z are the funds A held before the payment. In a replay scenario, the adversary republishes this payment; thus, a further amount x of funds is sent from A to Bob. The same vulnerability exists against certificates. Specifically, an attacker can republish an old certificate, in order to forcefully change a user's delegation choice.

One mitigation for replay attacks is based on an *address whitelist*. Specifically, the certificate defines an extra field in its metadata, which contains the addresses that are allowed to publish it. Naturally, this solution assumes that, upon creating the certificate, the stakeholder has already generated these addresses. Following, to verify a certificate, a node checks whether it is published in a transaction issued by a whitelisted address. In this case, the adversary can replay the certificate only if it accesses the private payment key of one of the whitelisted addresses. Notably, the information which counters the replay attack is in the certificate, so the transaction verifiers do not need to parse the entire ledger or maintain extra local state, as is the case of the counter-based mechanisms used in Ethereum [13].

An alternative mitigation mechanism is to use deadlines. In this case, a certificate defines a time limit, i.e., the latest block in which the transaction can be published. It is important to choose this limit carefully. If the limit is too low, the block producers might not receive it in time or might not prioritize it, and thus not publish it at within the necessary time period. On the other hand, if it is too high, the user needs to wait for a large amount of time until they are sure that the delegation is finalized, which hurts the usability of the system.

8 Final Remarks

The PoS paradigm is considered the major contender to the wide substitution for the energy-inefficient PoW with equivalent security guarantees. The former sidesteps the disadvantages of the latter by deploying significant more sophisticated cryptographic machinery. Despite the well-known cryptographic building blocks, e.g., *lottery protocol*, verifiable random functions, and committee selections, several types of attacks were proposed. Arguably, the most alarming ones are the *long-range attacks*.

Observe that these threats focused on the execution of the consensus protocol and the substitution of honest chairs by malicious one. A possibly more subtle class of attacks target the wallet definition for PoS. Only recently a general framework was proposed, and more concretely these attacks could be formulated, i.e., the *malleability attacks* of the address strings. The PoS wallet definition also introduces concrete mechanisms for the delegation of the rights over the stakes of the participants.

These mechanisms roughly split the participants into regular users, the ones that transfer the rights over the stake, and *delegators* or *stake pools*, who actually jointly perform the PoS consensus protocol and to whom the overall security of the system relies on. Thereby, despite the existence of more direct attacks to the consensus protocol, the PoS wallet and the delegation mechanism definition play a crucial role in the security of the delegated PoS consensus protocol.

References

1. M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, On the malleability of bitcoin transactions, in *FC 2015 Workshops*, ed. by M. Brenner, N. Christin, B. Johnson, K. Rohloff. *LNCS*, vol. 8976 (Springer, Heidelberg, 2015), pp. 1–18. https://doi.org/10.1007/978-3-662-48051-9_1
2. C. Badertscher, P. Gazi, A. Kiayias, A. Russell, V. Zikas, Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability, in *ACM CCS 2018*, ed. by D. Lie, M. Mannan, M. Backes, X.F. Wang (ACM Press, New York, 2018), pp. 913–930. https://doi.org/10.1145/3243734.3243848
3. M. Bellare, A. Boldyreva, A. O'Neill, Deterministic and efficiently searchable encryption, in *Annual International Cryptology Conference* (Springer, Berlin, 2007), pp. 535–552
4. L. Brünjes, A. Kiayias, E. Koutsoupias, A.-P. Stouka, Reward sharing schemes for stake pools, in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7–11, 2020* (IEEE, New York, 2020), pp. 256–275. https://doi.org/10.1109/EuroSP48549.2020.00024
5. M. Castro, B. Liskov, Practical byzantine fault tolerance, in *Proceedings of the Third Symposium on Operating Systems Design and Implementation* OSDI '99. (USENIX Association, USA, 1999), pp. 173–186. ISBN 978-1-8804-4639-1
6. B. David, P. Gazi, A. Kiayias, A. Russell, Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain, in *EUROCRYPT 2018, Part II* ed. by J.B. Nielsen, V. Rijmen. LNCS, vol. 10821 (Springer, Heidelberg, 2018), pp. 66–98. https://doi.org/10.1007/978-3-319-78375-8_3

7. C. Decker, R. Wattenhofer, Bitcoin transaction malleability and MtGox, in *ESORICS 2014, Part II*, ed. by M. Kutylowski, J. Vaidya. LNCS, vol. 8713 (Springer, Heidelberg, 2014), pp. 313–326. https://doi.org/10.1007/978-3-319-11212-1_18.
8. E. Deirmentzoglou, G. Papakyriakopoulos, C. Patsakis, A survey on long-range attacks for proof of stake protocols. IEEE Access **7**, 28712–28725 (2019). https://doi.org/10.1109/ACCESS.2019.2901858
9. D. Dolev, C. Dwork, M. Naor, Nonmalleable cryptography. SIAM Rev. **45**(4), 727–784 (2003)
10. J.R. Douceur, The Sybil attack, in *International Workshop on Peer-to-Peer Systems* (Springer, Berlin, 2002), pp. 251–260
11. C. Dwork, M. Naor, Pricing via processing or combatting junk mail, in *CRYPTO'92*, ed. by E.F. Brickell. LNCS, vol. 740 (Springer, Heidelberg, 1993), pp. 139–147. https://doi.org/10.1007/3-540-48071-4_10
12. Ethereum. Proof of stake faqs (2018a). https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs
13. Ethereum. Glossary: Account nonce (2018b). https://github.com/ethereum/wiki/wiki/Glossary
14. J.A. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in *EUROCRYPT 2015, Part II*, ed. by E. Oswald, M. Fischlin. LNCS, vol. 9057 (Springer, Heidelberg, 2015), pp. 281–310. https://doi.org/10.1007/978--3-662-46803-6_10
15. P. Gaži, A. Kiayias, A. Russell, Stake-bleeding attacks on proof-of-stake blockchains, in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, United States (Institute of Electrical and Electronics Engineers (IEEE), New York, 2018), pp. 85–92. ISBN 978-1-5386-7205-1. https://doi.org/10.1109/CVCBT.2018.00015. https://cryptovalley.swiss/event/crypto-valley-conference-2018/. 1st Crypto Valley Conference on Blockchain Technology, CVCBT 2018; Conference date: 20-06-2018 Through 22-06-2018
16. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: scaling byzantine agreements for cryptocurrencies, in *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, New York, NY, USA (Association for Computing Machinery, New York, 2017), pp. 51–68. ISBN 978-1-4503-5085-3. https://doi.org/10.1145/3132747.3132757
17. G. Itkis, L. Reyzin, Forward-secure signatures with optimal signing and verifying, in *CRYPTO 2001*, ed. by J. Kilian. LNCS, vol. 2139 (Springer, Heidelberg, 2001), pp. 332–354. https://doi.org/10.1007/3-540-44647-8_20
18. D. Karakostas, A. Kiayias, M. Larangeira, Account management in proof of stake ledgers, in *Security and Cryptography for Networks*, ed. by C. Galdi, V. Kolesnikov (Springer International Publishing, Cham, 2020), pp. 3–23. ISBN 978-3-030-57990-6
19. A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: a provably secure proof-of-stake blockchain protocol, in *CRYPTO 2017, Part I*, ed. by J. Katz, H. Shacham. LNCS, vol. 10401 (Springer, Heidelberg, August 2017), pp. 357–388. https://doi.org/10.1007/978-3-319-63688-7_12
20. S. King, S. Nadali, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake (2012)
21. D. Larimer, Transactions as proof-of-stake (2013)
22. D. Larimer, Bitshare: Delegated proof of stake consensus (2014)
23. W. Li, S. Andreina, J.-M. Bohli, G. Karame, Securing proof-of-stake blockchain protocols, in *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (Springer, Berlin, 2017), pp. 297–315
24. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008)
25. V. Zamfir, The friendly ghost (2015). https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost

Layer 2 Scaling Solutions for Blockchains

Subhra Mazumdar and Sushmita Ruj

1 Introduction

The property of immutability, security, and decentralization leads to acceptance of Bitcoin and other cryptocurrencies as alternative methods of payment [37]. However, the use of expensive consensus algorithms in the underlying blockchain framework [8, 38] impedes scalability [14, 45] and hinders its widespread adoption unlike conventional methods of electronic payments like Visa and PayPal [64]. In Bitcoin, every transaction needs to be verified for signature and correctness of format before it gets mined. Many blocks ends up in the queue, facing considerable waiting time before it gets verified due to increases in network size. Ethereum suffers from scalability issue as well due to constraint on gas limit for processing transactions. Increasing the throughput is a challenging task. Any modification made to the underlying blockchain or any new protocol used for enhancing scalability must not introduce vulnerability in the system or compromise privacy.

In this chapter, we first discuss the major challenge faced by a decentralized network, *scalability trilemma*, in Sect. 2. The factors affecting scalability of blockchain has been stated in Sect. 3. In Sect. 4, we discuss the various scaling solutions briefly. We next discuss in details the various *off-chain* solutions and their utility in Sect. 5. Channels, both payment and generalized state channels, have been explained in Sect. 5.1. Sidechain and commit chain have been discussed in Sects. 5.2 and 5.3, respectively, and conclude the chapter in Sect. 6.

S. Mazumdar (✉)
Security and Privacy Research Unit, TU Wien & CDL-BOT, Vienna, Austria

S. Ruj
School of Computer Science and Engineering, University of New South Wales, Sydney, Kensington, NSW, Australia
e-mail: sushmita.ruj@unsw.edu.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_9

2 Scalability Trilemma

It is hard to achieve *scalability, security,* and *decentralization* at the same time in a distributed network. Improving one of the factors might lead to comprising on at least one of the other two factors [50, 59], depicted in Fig. 1. For example, Bitcoin is decentralized and uses secure public blockchain but cannot process hundreds or thousands of transactions per second. Permissioned blockchains like Hyperledger Fabric have high transaction throughput, but they are not completely decentralized as only few nodes are involved in achieving consensus in the network. We discuss each component of the trilemma.

Decentralization
A system that is decentralized means the system is controlled by multiple owners, and one does not overpower the other. The more decentralized a system, the more secure it is. Ethereum and Bitcoin use PoW [8, 38] for mining new blocks. This consumes a lot of energy and reduces speed and efficiency. Additionally, there is no central moderator to resolve dispute.

Security
This property allows blockchain to be resilient toward external attacks as well as internal disruption. As seen previously, the more decentralized a system, the less risk of the system being susceptible to centralized attacks. For security, the blockchain network relies on consensus mechanism like proof of work (PoW), where miners are required to solve complex hash puzzle in order to include blocks. Consequently, this reduces throughput and increases network latency, losing out on speed and efficiency offered by centralized networks in processing transactions.

Scalability
The higher the scalability of the system, the greater is its capacity. In terms of blockchain, scalability ensures that a large number of transaction get processed. However, scalability is ensured at the cost of security. Ethereum intends to opt for alternate mining mechanism like proof of stake [29]. However, it compromises on decentralization. Further, if the difficulty of puzzles is reduced for faster validation process PoW [8], then this would compromise on the security aspect.

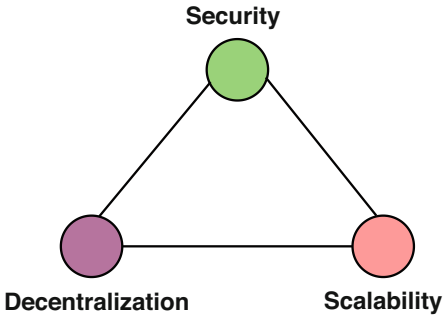


Fig. 1 Blockchain scaling trilemma

3 Factors Responsible for Decreasing Blockchain's Scalability

In order to form a decentralized peer-to-peer network having nodes with low bandwidth and limited computational powers a limit on the block size was introduced. However, this limit has put constraint on the number of transactions being processed. Since the network is decentralized, all the nodes must be in sync with the latest view of the blockchain. Once the transaction gets added in the blockchain, next set of blocks must be mined on top of it so as to ensure finality. We discuss certain factors which decreases the throughput [41]:

- **Latency** It is defined as the time taken to confirm that a transaction has been included in the blockchain. For Bitcoin, the average time to create and secure a block in the chain takes around 10 minutes, and for Ethereum, it takes around 12.5 seconds.
- **Block Size** This is the total size of the transaction that can enter the block. The limit of the block size is around 4 MB for Bitcoin.

Both these factors lead to a decrease in the number of transactions processed in a unit time. Other factors responsible are *Bootstrap time* and *cost per confirmed transaction* [14]. The time taken for a new node to download the entire blockchain in order to validate the current system state is the Bootstrap time. The time factor is generally linear in the size of the chain. The cost for confirming each transaction in blockchain involves mining cost, i.e., resource (in terms of hardware, electricity) expended by miners in generating PoW for each block; validation cost, i.e., computation needed to check the correctness of input, output, and other cryptographic verification, like signature verification, etc., and bandwidth needed for transmission and receipt of blocks; and the storage cost involved in downloading the entire blockchain history as well as set of unspent transactions.

We discuss how relaxing the limit on the block size or latency for achieving scalability is not a solution and state the drawbacks:

- Increasing Block Size Limit: The greater the block size, the more the number of transaction processed. However, this means more bandwidth and higher storage cost are required for processing each block. Nodes with greater processing power will be able to scale up transaction, and hence, it leads to centralization in the network. Larger blocks propagate slowly through the network leading to an increase in the number of orphan blocks and an increased probability of a successful double-spend [10].
- Decreasing Latency: The faster the blocks are produced, the higher the chance that the node produces a stale block. Given the block confirmation time in Bitcoin is 600 s, if in every 12 s a new block is produced, then 12/600 or 0.02 stale block is produced per valid block. If the time is reduced to 60 s, then 12/60 or 0.2 stale block is produced per valid block. This impacts finality of transactions, may lead to soft forks, and reduces the security of the network [12, 24].

In the next section, we discuss some of the scaling solutions which has been widely adopted by the blockchain community.

4 Scaling Solutions in Blockchain

We categorize the solutions developed as *on-chain scaling solutions* and *off-chain scaling solutions*:

- On-Chain Solutions: On-chain refers to the *blockchain layer* or *layer 1*. Solutions proposed in this layer aims at augmenting the base layer of the blockchain protocol for achieving scalability. This involves changing the rules of the protocol or increasing the amount of data in each block or reducing the latency in order to increase overall network throughput. We discuss few solutions proposed for layer 1 scalability:
  - Increased Block Size/Using Segregated Witness (SegWit): This involves making changes to the blockchain network. Increase in block size will lead to processing of more number of transactions. However, as argued earlier, this would require more storage capacity of a full node in order to store the entire history as well as centralization [10]. Later, SegWit was suggested. Since signature of the transaction data occupied around 65% of the block storage, SegWit proposed segregating the signature script part of each transaction from a block and store them into an extended block or witness [56]. This allowed packing more transactions into a single block.
  - Alternative Consensus Mechanism: Proof of work consumes too much of computation resource as well as hinders scalability. Proof of stake [29, 30] is comparatively faster as it involves participant to deposit certain amount of cryptocurrency as a commitment for behaving honestly. Participants who have a certain amount of stake in the network can validate a new block. If any validator produces an invalid block, his or her stakes will be slashed in the form of penalty. Other forms of consensus are as follows:
    (1) Proof of burn—The miner must "burn" tokens, i.e., they must be sent to an address so that these tokens are no longer accessible and in circulation, in order to gain eligibility for mining a block. As more and more miners want to mine, they need to consistently burn more and more tokens to stay competitive [49].
    (2) Proof of capacity—Miners dedicate available space of their storage device to store a list of possible solutions before the mining commences. If it is able to match the solution with the hash value presented, then it gets the mining reward [46].
    (3) Proof of elapsed time—Used mostly in permissioned blockchain network, each participant node in the network generates a random waiting time and goes to sleep for the duration [47]. The node with the shortest

waiting time wakes up and commits a new block to the blockchain. Prism [6, 67], a new consensus protocol achieves a throughput of over 70,000 transactions at low latency.

– Sharding: The blockchain is divided into many units or shards. The workload of the master blockchain gets distributed into each unit, which can act independently [32]. Parallel computation in each of these shards enables faster computation and increases scalability. Each shard can communicate between each other but reports directly to the master blockchain. However, sharded blockchain is susceptible to 51% attack, and a security failure in one of the shards can result in failure in the entire blockchain network.

On-chain scaling solutions may involve hard fork in the system, and these are not easy to implement. The amount of scalability achieved falls short of the expected level of throughput provided by conventional payment systems.

- Off-Chain Solutions: Off-chain scaling means creating alternative protocols on top of layer 1. Hence, this is often termed as *layer 2*. Instead of recording all the transactions on-chain, the transaction are executed *off-chain*. As an example, we show in Fig. 2 how payments can be carried out off-chain between *Alice* and *Bob* by constructing payment channels. *Alice* deposits 0.05 BTC in an escrow account formed with *Bob*. This transaction is termed as opening of channel and it gets recorded in the blockchain. Later, *Alice* executes three such payments, each of 0.01 BTC. Finally, they can settle by closing the channel, recording the final balance on-chain—with *Alice* getting 0.02 BTC and *Bob* getting 0.03 BTC. Dispute arising out of any of the payment execution can be tracked and resolved on-chain. Several types of *off-chain* solutions are as follows:

  – Channels: A channel is a private connection between two parties who deposit certain amount in a multi-sig account as per the terms and conditions of

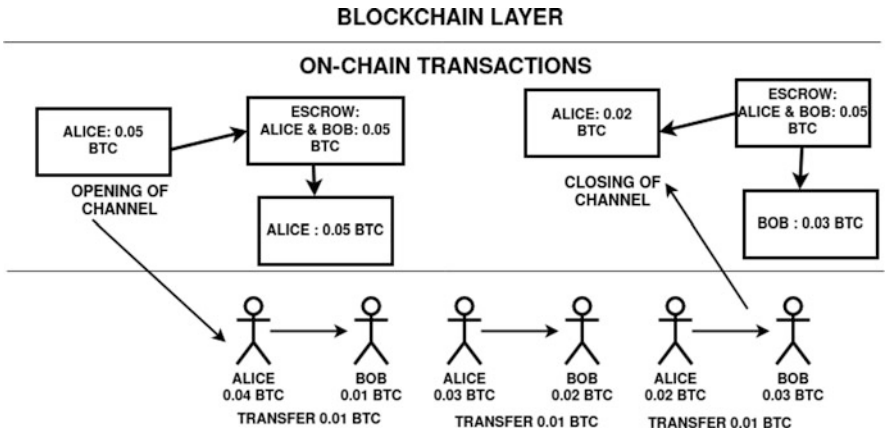


Fig. 2 An instance of off-chain payment between *Alice* and *Bob*

a contract. These two parties can carry out unlimited transactions among themselves. Several variants of channels have been discussed:

- Payment Channels: It supports off-chain payment between parties. Channels can be unidirectional, designed for one way payment, or bidirectional, where both the parties can send and receive payments. Several channel designs are as follows:
    - *Spilman-style payment channels*: This is an example of unidirectional payment channel where payment can occur in one direction [9, 36]. The receiver needs to close the channel before the channel lifetime expires; else, the sender reverts the fund back to its wallet after the elapse of lock time. It allows the sender to stay offline without the risk of losing funds. However, this type of channel suffers from risk of malleability. A refund transaction for sender is created from the deposit transaction. However, the deposit transaction isn't broadcasted before the refund transaction gets created. Later, if the deposit transaction gets broadcasted, a mismatch in the transaction id of refund transaction might render it invalid. This might cause sender's funds to be stuck forever.
    - *CLTV-style payment channels*: This is a unidirectional payment channel, but the problem of malleability is circumvented using lock time, corresponding Bitcoin script instruction being OP_CLTV [42]. The transaction cannot be spent before the lock time elapses.
    - *Poon-Dryja payment channels*: Funds are deposited by the parties willing to open this type of trustless bidirectional payment channel in two-of-two multi-sig address [45]. The lifetime of the channel is indefinite unless both the parties mutually decide to close or any of the party unilaterally closes the channel. We will discuss this model later in details.
    - *Decker-Wattenhofer duplex payment channels*: A duplex micropayment channel between two parties establishes two unidirectional channel in either direction. It has the property of atomically resetting the channel balance, without requiring to close the existing channel and opening a new one for restoring the balance [16].
      Invalidation tree is used to keep track of series of off-chain transactions, starting from the root or the funding transaction and ending at payment finalization transaction, which is the leaf node, as shown in Fig. 3. Each node of the tree is a multi-sig output. Each transaction in the tree is provided with a time lock, and there exists a unique transaction with minimal time lock among all transactions sharing the same parent. In order to invalidate a subtree, a new child transaction, with time lock less than all the existing transaction, is added to the subtree's root output. Though the lifetime of the channel is infinite, a maximum time period is set before which the parties must commit their state to the blockchain. Closing of duplex

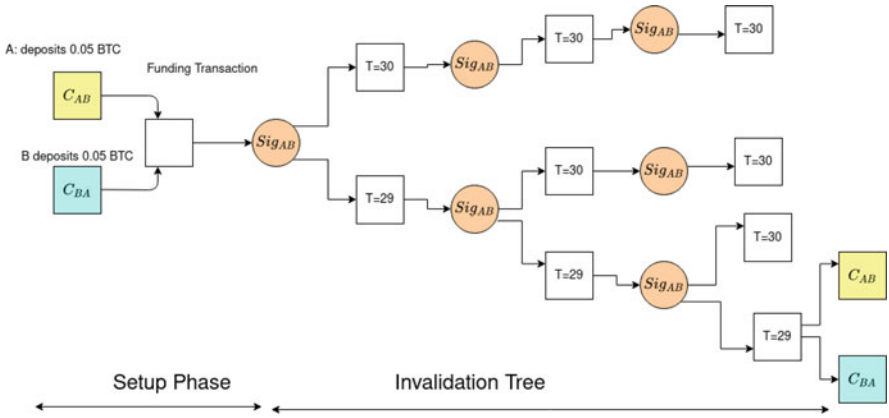


Fig. 3 Duplex micropayment channel

micropayment channel is a bit different for the noncooperative case. If both the parties agree, a transaction is created with payout being decided for each of the party, which can be spent immediately. In case of disagreement, a party needs to submit the latest branch of the invalidation tree, the number of transaction submitted being limited by the depth of the invalidation tree.

- *Eltoo channels*: In the channel design suggested in Poon-Dryja or Decker-Watternhofer, an older state of the channel gets replaced by a new state provided it has a higher *nsequence* number. Any conflicting transaction broadcasted by a party will be replaced by the latest state if it submits a state with stale *nsequence* number. However, miners can be easily bribed into accepting older state by offering higher transaction fee. To avoid such a situation and enforce correctness, *eltoo channels* [15] introduce the concept of state number which is enforceable unlike *nsequence* number. Any settlement specified by a later state, committed to the blockchain, is allowed to re-spend any of the previous states. In order to enable this, a new concept, known as *floating transactions*, has been introduced. These transactions can be bound to any previous transaction with matching scripts. The intermediate update transactions do not have to be committed to the blockchain.

- Channel factories [11, 66] are used to create more than one payment channels between two or more parties without recording every funding transaction on-chain. The state of the channel can be updated, unlike in normal payment channel where such balance updation was not possible. n parties can come together and deposit funds to create a channel factory. The number of input and outputs is $O(n)$. However, in PCN, this would have been $O(n^2)$. This is treated as the funding transaction and gets broadcasted

on-chain. This transaction must get sufficient number of confirmation. After this period, any number of secondary payment channels can be opened. If the parties decide to close these channels, then they can either go on-chain and close the channel factory or update the channel balance off-chain and form a new channel.
Apart from this, a generalized Bitcoin-compatible PCN [4] and, based on it, construction of virtual channels in such networks have been proposed in [3].

- State channels allow the execution of more complex contracts off-chain where more than two parties can deposit their asset and, with unanimous agreement, can undergo state transition. There are several types of state channel construction proposed in literature like Sprites [36], Perun [19], Counterfactual [13], Celer Network [17], etc. Generalized constructions of virtual state channels have been proposed in [20].

- Sidechains: This is a separate blockchain which runs alongside the main blockchain, running their own consensus mechanisms [5]. The two blockchains can communicate among themselves, and assets can be moved across these two chains. A more robust consensus mechanism can be used in the sidechain to enhance scalability. But this might result in compromising on the security. If the security fails, then a party can lose all funds.
- Plasma Chain: The concept is the same like sidechain except that the root of each plasma blockchain is made available on the main chain or parent chain [44]. Hence, it is more secure than sidechain, i.e., if the main blockchain fails, then there is a breach of security. This feature leads to loss of flexibility of performing complex operations unlike that in sidechains because of the extra precaution.

Since the chapter focuses on off-chain scaling solutions, in the next section, we discuss the utility of off-chain scaling and state each of the solutions in details.

5 Off-Chain Scaling

These solutions are implemented separately from *layer 1*. Some solutions, like channels, rely on the blockchain for security and resolution of dispute. Other solutions, like sidechain and commit chain, communicate with the underlying blockchain but have their own security properties. We discuss the utility of off-chain solutions followed by a detailed description of channels, sidechain, and commit chain.

Utility of Off-Chain Scaling Solutions
We discuss some of the advantages of off-chain solutions:

- It massively cuts down data processing on the blockchain by running computations off-chain.

- The trust assumption or the consensus mechanism used in underlying blockchain need not be changed, and the amount of data stored in the base layer gets minimized.
- It cuts down on the expensive consensus mechanism since it relies on layer one to resolve dispute and reach an agreement.
- Scalability may be bounded by the communication bandwidth and latency of the participants since none of the transaction gets recorded on-chain.

In the next section, we discuss some of the layer 2 solutions in details.

5.1 Channels

State channel allows parties to perform multiple transactions without recording any state change in the Blockchain, except the opening and closing of channel. The life cycle of a channel is divided into three phases: (1) *Establishing channels*—All parties willing to establish a channel lock some amount of collateral in the blockchain. The channel can be closed if all the parties provide their consent. It can be closed unilaterally by a party via appropriate refund mechanism. (2) *Channel state transition*—If a party proposes a state transition, it broadcasts the new state with its signature to the rest of the participants. The new state is verified based on the previous state and other auxiliary conditions. (3) *Channel dispute/closure*—If there is any disagreement over the new state proposed by a party or if one of the parties has stopped responding, then the party can go on-chain and enforce the new state. This would also result in closure of the channel and settlement of funds as per the last updated valid state. If the party is honest, then the proposed state gets accepted; else, it will be penalized for cheating. We will discuss in details the construction of Poon-Dryja style payment channels. This deals with simple money transfer. Later, we will also discuss how state channels can be used for handling complex state change and arbitrary smart contract logic.

5.1.1 Channels Dealing with Simple Payments: Poon and Dryja Bidirectional Payment Channel

The payment layer, also termed as Lightning Network, proposed by Poon and Dryja [45] enables large amount of off-chain payments without recording the same in the blockchain, leveraging on the security of the underlying Bitcoin blockchain. The bidirectional channel between two parties allows the participants to *lock* funds in a joint multi-sig account, which is also termed as *creation of channel*. Parties can then shift funds between themselves as per the transaction request. Such parties are also called as *lightning nodes*. Such set of nodes can be interconnected to form a network which can route payment between any parties. In the following sections,

we describe the process of opening of the channel, updating the channel balance and closing of channel.

Opening of Channel

With mutual consent from both the parties, they can deposit funds in a two-out-of-two multi-sig address to initiate a funding transaction. A *multi-sig address* is a Bitcoin address which requires multiple private keys or signature of the owners of such private keys in order to spend the Bitcoins from the address. Once the amount is sent to the multi-sig address, the parties create the first commitment transaction before exchanging signature for the funding transaction.

In the example illustrated in Fig. 4, parties *Alice* and *Bob* send 0.005 BTC in the two-out-of-two multi-sig address but don't broadcast the funding transaction as of now. They create a first commitment transaction which allows each party to spend 0.005 BTC before the funding transaction gets broadcasted. *Alice* and *Bob* both create a secret and exchange the hash H_{Alice} and H_{Bob}.

Alice creates a copy of first commitment transaction for *Bob* where *Alice* sends 0.005 BTC to herself and the rest of the amount to *Bob* to a second multi-sig address. But *Bob* is allowed to spend the money sent to the second multi-sig address after a relative lock time from the time it got broadcasted in blockchain. This is realized in

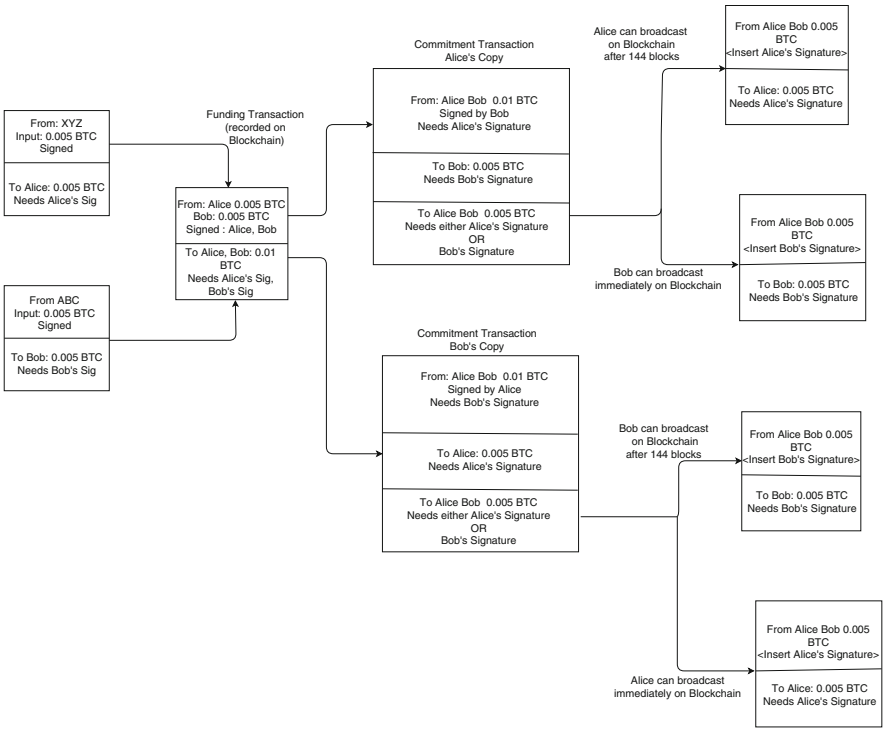


Fig. 4 Setting up bidirectional channel of 0.01 BTC between *Alice* and *Bob*

the output script using a CheckSequenceverify-Lock (CSV-lock). *Alice* can spend the amount from second multi-sig address if *Bob* shares the preimage of the hash H_{Bob}. She signs her copy of commitment transaction and sends to *Bob*. The latter creates a similar copy of commitment for *Alice* providing his signature. It contains an output of 0.005 BTC which can be immediately spend by *Bob* and the rest of the output of 0.005 BTC for *Alice* to a second multi-sig address. *Alice* cannot spend this output before the elapse of relative time lock period. *Bo* can spend the output of second multi-sig address, provided *Alice* has provided the preimage of the hash H_{Alice}. With the half-signed transaction in place, both parties now exchange their signature and broadcasts the funding transaction, which confirms the opening of channel [63].

Why is it a bad strategy to sign the funding transaction before the creation of first commitment transaction? *Suppose both the parties exchange signature for the funding transaction. Then either of the parties have the right to broadcast the transaction and record the channel opening on-chain. If they do so, the funds can remain locked forever if either of the parties doesn't cooperate while spending from the funding transaction. Once signature gets exchanged over first commitment transaction and then signatures for funding transaction is exchanged, then either of the parties have the flexibility to unlock the coins from the multi-sig address.*

Updating Channel Balance

Before creating a fresh commitment transaction, the parties exchange the preimage of the hashes exchanged for the previous commitment transaction. This procedure is known as *invalidating a transaction*. In the next step, both of them update the balance as per the terms of the transaction. Say if *Alice* wants to transfer 0.001 BTC to *Bob*, both of them create fresh hash values H_{Alice1} and H_{Bob1} and exchange it. Then *Alice* creates a fresh commitment transaction with the former sending 0.004 BTC to herself and 0.006 BTC to a second multi-sig address which can spend either by *Bob* after a relative time lock period or if *Alice* gets the preimage of the new hash value H_{Bob1}. Similarly, *Bob* creates a similar copy of the transaction and share it with *Alice*. If either of the parties tries to broadcast an older transaction, then the counterparty uses the preimage of the stale hash value, H_{Alice} or H_{Bob}, and spends the output of such transaction immediately. The malicious party gets penalized since all the fund it had deposited into the channel is slashed from its account.

Closing Channel

Both the parties can mutually decide to close the channel by creating a new transaction reflecting the final balance based on the latest state of the channel. *Alice* and *Bob* put their signature on the closing transaction so that they can immediately spend the output of the transaction. If a party doesn't cooperate, then the counterparty can unilaterally close the channel by broadcasting its copy of the final transaction. In such a case, it has to wait for a relative time period before spending the output.

5.1.2 Lightning Network Construction

Instead of opening a new channel, a party can find a path connecting it to a destination via some intermediaries. This saves the cost of locking fresh collateral in the channel as well as recording another transaction on blockchain. This set of interconnected channels forms the *Lightning Network*. To send money from a payer to payee via a set of nodes, it must be ensured that the intermediaries do not cheat and extort money from the payer and at the same time the payee receives the intended amount. We describe a very simple payment protocol which uses hash as commitment.

Hashed Time Lock Contract
Alice wants to transfer 0.001 BTC to *Charlie* via *Bob*, the latter claiming a processing fee of 0.001 BTC. *Charlie* samples a random value x and computes $H = \mathcal{H}(x)$ and sends it to *Alice*, as shown in Fig. 5. The payer uses onion routing for forwarding the conditional payment across the path $Alice \rightarrow Bob \rightarrow Charlie$. She forwards a conditional payment of 0.002 BTC to *Bob*. She sends the money to a different mutlisig address with the hash H. The output from the multi-sig address can be spent in two ways. Either *Bob* can provide the preimage x along with his signature and claim 0.002 BTC from *Alice*. Else, after a time period of say t units, *Alice* can sign and unlock this money. Note that this time period is not a relative time lock but an absolute time lock (mentioned in the script as CheckLockTimeVerify (CLTV)). Either *Bob* can settle the transaction off-chain with *Alice* by updating the channel balance upon providing the preimage x. But *Bob* still has not got the

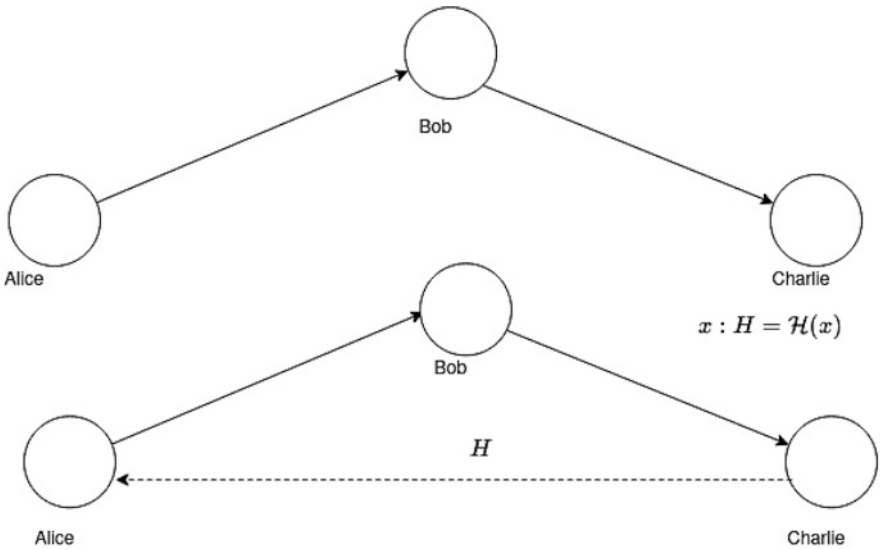


Fig. 5 Setup phase of HTLC

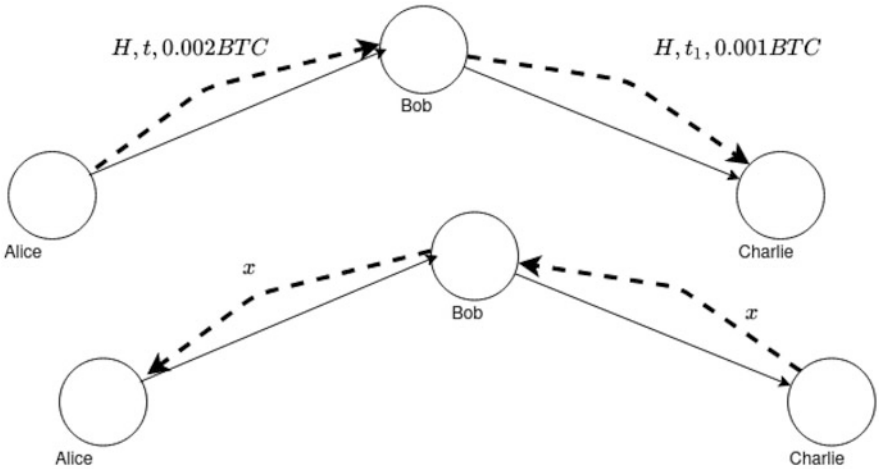


Fig. 6 Routing payment from *Alice* to *Carol* using HTLC

preimage. Upon decrypting the encrypted onion blob, he finds that the conditional payment has to be forwarded to *Charlie*. Hence, he will create a similar contract with *Charlie*, whereby it transfers 0.001 BTC to a new multi-sig address. The time period within which *Charlie* must provide the preimage is t_1 where $t_1 < t$, as shown in Fig. 6. This will allow *Bob* to get the preimage x and claim money from *Alice* [39, 62].

Let us define the procedure in details. The initial balance of the channel between *Alice* and *Bob* is 0.01 BTC each, with either of the parties depositing 0.005 BTC. *Alice* creates a new multi-sig address sending 0.002 BTC with two options of spending the amount as mentioned before. She has a balance of 0.003 BTC, *Bob* has a balance of 0.005 BTC, and the money locked in hashed time lock contract is 0.002 BTC. He can use the preimage to claim the amount. If he goes on-chain and broadcasts the transaction, then *Alice* can spend 0.003 BTC immediately. *Bob* can claim the money locked in hashed time lock contract by providing his preimage and signature. However, again, a relative time lock is added to this output as well where *Bob* can actually spend a transaction after a certain time period has elapsed. Had *Alice* broadcasted the commitment transaction, then *Bob* could have spent immediately 0.005 BTC as well as the money locked in the hashed time lock contract.

Apart from Lightning Network, hashed time lock contract is used for routing payments in Raiden Network as well, a payment channel network for Ethereum.

It might be possible that the path selected for routing payment may suffer from failure of intermediate nodes. This might hinder the payment, but we state two countermeasures to handle such situations [45]. We discuss in the next section certain protection mechanisms in order to prevent an honest node from losing money if a malicious node tries to cheat.

Protection Mechanisms in Lightning Network

- *Settling on-chain*. While resolving a conditional payment, if a party finds its counterparty has stopped responding, then it goes on-chain, reveals the preimage in order to claim payment, and closes the channel. For example, in Fig. 7, if *Bob* doesn't respond, *Charlie* broadcasts its transaction with preimage and signature and closes the channel formed with *Bob*. *Charlie* claims the money, and *Alice* withdraws the money locked in the contract with *Bob*, after lock time elapses.
- *Rerouting*. An intermediate can stop forwarding the conditional payment to the next neighbor. For example, in Fig. 8, in the path $Alice \rightarrow Bob \rightarrow Charlie \rightarrow Dave$, *Charlie* may not forward the payment. Ultimately, the contract doesn't reach the recipient *Dave*. *Alice* may not wait for indefinite time period and start forming HTLCs across another path $Alice \rightarrow Bob \rightarrow Mathew \rightarrow Dave$. Simultaneously, a new path connecting *Dave* to *Alice* is searched for so that in case the HTLC via path $Alice \rightarrow Bob \rightarrow Charlie \rightarrow Dave$ gets established, the money can be transferred back to *Alice* via path $Dave \rightarrow Eric \rightarrow Bob \rightarrow Alice$. This ensures a risk-free payment via another path.

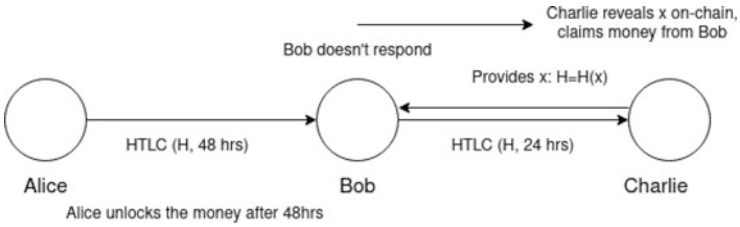


Fig. 7 Settling of payment on-chain

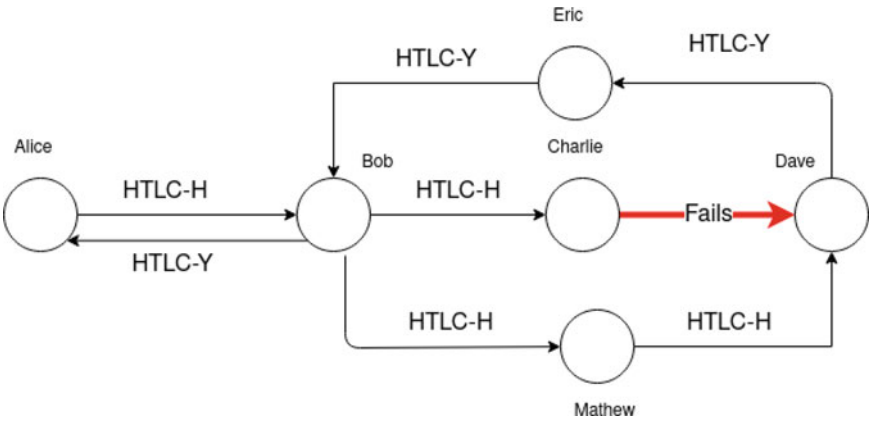


Fig. 8 Rerouting payment from *Alice* to *Dave* via $Alice \rightarrow Bob \rightarrow Mathew \rightarrow Dave$ using condition H. If *Charlie* establishes HTLC with *Dave*, then the same must be refunded via path $Dave \rightarrow Eric \rightarrow Bob \rightarrow Alice$ using condition Y

Vulnerabilities in Lightning Network

We state the plausible attacks on Lightning Network which leads to loss of funds, loss of privacy, or even jamming the network, rendering it unsuitable for processing any payment:

- Channel Probing: An attacker can easily discover channel balances using probing [27, 60]. It performs multiple payments adjusting the maximum and minimum capacity of the channel using binary search.
- Time Dilation Attack: If an attacker sets up several Sybil nodes and forms connection with a Lightning node such that it is no longer connected to any honest users, the attacker can isolate that node from receiving real network data [51]. This is termed as *eclipsing a node*. Attacker provides a stale view of the blockchain to the victim, forcing the latter to process transaction at a slower rate than normal. Once channel is closed, the attacker can steal funds from its victim taking advantage of the fact that the victim has not seen the closing transaction.
- Linking Attacks: Lightning nodes can be linked with Bitcoin addresses [53]. Privacy of these nodes in layer 2 gets compromised, making it susceptible to denial-of-service attack.
- Griefing Attack: An adversary may initiate an arbitrary number of HTLC payments to a node under her control [31, 52]. This node may then simply ignore incoming HTLC requests, forcing the nodes involved to wait for the time locks to expire. Upon expiry, the entire state is rolled back, circumventing fee deduction. Therefore, payment griefing allows an adversary to temporarily claim channel capacity free of charge. Some edges in the network can get eliminated as well due to channel closure.
- Flood and Loot Attack: An attacker initiates multiple two hop self-payment, with the victim node lying in the middle, blocking the entire channel capacity. Once the HTLC's reach the target, the receiver starts claiming payment by releasing the preimage to the victim node and settles the payment off-chain [26]. The victim node attempts to send back the preimage to the source node. The source node doesn't respond, forcing the victim node to settle the transaction on-chain. As the time-out approaches, the victim submits several such transactions in the blockchain. Some of these will fail to enter into blocks by the time-out due to congestion and high blockchain fees. The attacker claims any HTLCs by raising the fee of the refund transactions than that of the victim's redeem transactions.
- Wormhole Attack: Hashed time lock contract uses the same hash across the route forwarding payment. It is highly likely that two colluding nodes in the path share the preimage directly [34]. They prevent honest intermediaries from participating in the successful completion of the payment, stealing their fees.

5.1.3 Routing Algorithms in Payment Channel Network

Routing of payments from sender to receiver via set of intermediate nodes without trusting any of them is a crucial problem in PCN. The difficulty lies in figuring out

set of channels with sufficient capacity for routing a payment. Only the opening balance of the channel is known publicly, the residual capacity of channel might be different due to several off-chain payments. The routing algorithm must not reveal the identity of payer or payee. We discuss few routing algorithms and the security guarantee it offers.

Flare
This routing algorithm for Lightning Network uses a trustless source routing scheme. It can also be scaled to the network size of at least several hundred or thousand nodes [48]. The algorithm comprises two different stage—*proactive* and *reactive*:

- Route Discovery (Proactive Part): Each node in Lightning Network constructs a routing table. The routing table is a subset of channels and helps in finding a path to the recipient. If the information is not sufficient to find a route, the table is used to determine the beacon nodes which assist in finding the route. In the initial stage, a node has clear view of the neighboring nodes. With the help of beacon nodes, the visibility of the network for a given node is increased beyond its local neighborhood by incorporating random nodes. Hence, with high probability, a node determines whether it can reach a particular node or not.
- Route Selection (Reactive Part): When a node receives a request to route a payment to a particular payee, it makes use of its own routing table and the routing table of the payee to figure out a route from the payer to the payee. If no such routes exist, beacon node corresponding to the payee and other nodes can pitch in with their routing table in constructing an appropriate path. If there exists several such routes, they can be ranked accordingly. Ranking is done based on some predetermined cost function. The cost function is defined based on static information, like route length, and dynamic information, like fees to use channels. Based on ranks assigned to the routes, the sender selects the best feasible path for routing its payment.

The problem with this routing algorithm is that it violates privacy. Nodes keep track of their $k - neighborhood$, i.e., all the nodes at a distance of hop length k. In the process, the weight of the links in this $k - neighborhood$ gets revealed. It is quite inefficient since nodes spread all updates to the $k - neighborhood$. Each credit change needs to be communicated leading to flooding of message in the network.

SpeedyMurmur
The algorithm is based on landmark routing [54], hiding the payment value as well as the identity of payer and payee. It requires to first identify a set of *landmark nodes*. Generally, a node which is highly connected forms the landmark. A payer traces a route to landmark, and similarly, a route from payee to landmark can be figured out. Concatenating both, a route from payer to payee gets constructed. We describe the main subroutines as follows:

- *setRoutes*: Assuming k such landmarks have been selected, iterate over each landmark, and assign BFS-based coordinate to each of the nodes per landmark.

Once a landmark is selected, it is assigned an empty vector as its coordinate. This node now selects all its neighbors and probes whether the links to such nodes have enough credit and has not been assigned any coordinate. If such a node is found, it is added to the spanning tree, and the coordinate assigned is the concatenation of the parent's coordinate along with a random $b - bit$ string. In an asynchronous setting, an upper bound on the time limit is set for a node to expect a message from any of its neighbor to be added to the spanning tree. Once such a request is received, the node assigns this neighbor as its parent. Initially, a node adds all those neighbors which have bidirectional nonzero capacity links. Later, all nodes having unidirectional nonzero capacity links get added.
The process is repeated for the k landmarks resulting in k such spanning trees and coordinates based on each of the spanning tree. This is termed as *prefix embedding*. Let the coordinate with respect to landmark L_1 for node u and v be $id_1(u)$ and $id_1(v)$; then the distance between node u and v is

$$dist(id_1(u), id_1(v)) = |id_1(u)| + |id_1(v)| - 2cpl(id_1(u), id_1(v)) \quad (1)$$

where $id_1(u)$ denotes coordinate length and $cpl(id_1(u), id_1(v))$ denotes the length of the common prefix, i.e., when the id is assigned to each node u and v, this is the portion of the string matching in both. Such common prefix signifies the shared ancestor. To avoid double counting, this value needs to be deducted.

- *setCred*: A pair of nodes may need to change the capacity of the link (or the channel) present between them. Later, they check if such a change of capacity leads to change in coordinate. In three situations, there is a possibility that the coordinates might change:
  - New nonzero link: Given a pair of nodes, one of them is not yet part of the tree and should choose the other as their parent.
  - New nonzero link: Given a pair of node say (u, v), if u has a parent with unidirectional link and later v forms a bidirectional link with u, then u must change its parent to v.
  - Removed link: If a node loses its link to its existing parent (i.e., credit in the link drops to 0), it should find a new parent.
    If a node changes its parent, then its descendants must undergo a coordinate change as well. They remove their existing coordinates and inform their neighbors about the same. Later, they choose a new parent and undergo a coordinate reassignment. If a node receives several requests from neighbors who can be its potential parent, it chooses the one having the shortest route to the landmark. Change in configuration is recorded with respect to each of the landmark nodes.
- *routePay*: For payment of value v from payer to payee, the payer splits the total transaction value into $k - shares$, i.e., $v_1, v_2, \ldots, v_k$ for each of the k landmarks. k such paths need to be discovered for transmitting each payment, and k different addresses of the receiver are generated, i.e., $dest_i(recv), i \in [1, k]$. For the share v_i, the path needs to be found out based on the coordinate assigned using

landmark $L_i, i \in [1, k]$ as the reference. Starting from the payer, a neighbor, say m, is selected such that the distance to the destination node is shortest, i.e., $d(id_i(m), dest_i(recv))$ is minimum and the link has capacity at least v_i. This is repeated for the rest of the $k-1$ shares as well. If the routing fails, then the route search procedure is repeated for a different split of the transaction value.

Flash

SpeedyMurmur suffers from low transaction throughput and susceptible to failure because the split in transaction value doesn't take into consideration the change in capacity of each link. Flash deals with this problem by differentiating large payments, termed as *elephant payments*, from small-valued payments, termed as *mice payments* [65]. For elephant payments, a modified max-flow algorithm based on Edmond-Karp is used for finding k such set of good paths with sufficient capacity for routing large valued payments. Given the set of k paths, in order to route the payment using minimum fee, an optimization program is designed to split the payments over k paths. For mice payments, each node maintains a routing table which figures out the path to the receiver. Each payer figures out m such shortest path where $m < k$. If such paths don't exist, they can be computed dynamically using the local topology and existing routing information. The routing table is periodically refreshed based on the change in topology of the payment channel network. Once m such paths are found, the sender tries to route the full payment value via one path. If it doesn't succeed, it checks the amount remaining to be routed and tries to route it via the next available path.

Spider

It is a routing solution that packetizes transactions, splits them into several transaction units, and sends them via different paths at different rates [57]. Each transaction unit can be transmitted using onion encryption, to hide the full route from intermediate routers. Breaking up of payments into packets allows to complete even large transactions on low-capacity payment channels over time. Congestion control across multiple paths ensures balance utilization of channels and fairness across flows.

Each payment channel has a router which maintains price variables, updated periodically based on the current arrival rate of transaction units in channel, the available channel balance, and the number of transactions currently queued up in the router. Transaction queues up in the router present in each payment channel whenever the channel lacks funds to forward them immediately. Queuing of transaction indicates that either the transaction is being processed at a faster rate and the rate needs to be brought down or the channel lacks capacity. If the node uses a congestion control protocol that controls queues, it could detect both capacity and imbalance violations. Any standard congestion control algorithm used for routing in network can be adapted in PCN. The routers monitor the time that each packet spends in their queue. If the time spent exceeds a certain threshold, the packet is marked. If the packet is already marked, then these are forwarded. Once receiver receives the packet, it sends acknowledgment back to the sender that can interpret whether a route exists.

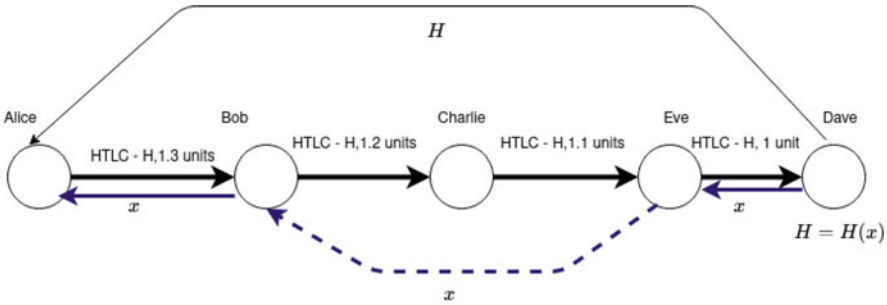


Fig. 9 Wormhole attack, *Bob* and *Eve* steal *Charlie*'s fee

5.1.4 Payment Protocol in Payment Channel Network

The hashed time lock contract (HTLC) protocol uses the same commitment across the path routing the payment. For example, *Alice* wants to transfer 1 unit of coin to *Dave* via *Bob, Charlie*, and *Eve*. *Dave* constructs a hash H from preimage x and shares it with Alice. We assume that each party charges 0.1 unit as processing fee. Thus, *Alice* will transfer 1.3 coins via this path. After the conditions of payment reach *Dave*, he releases the preimage to *Eve*. However, as discussed before under *vulnerabilities in Lightning Network*, this is susceptible to *wormhole attack* [34]. *Eve* and *Bob* can collude. Upon seeing that both have received the same condition for payment, instead of sharing the preimage with *Charlie*, she shares it directly with *Bob*, as shown in Fig. 9. Just before elapse of lock time, *Eve* cancels the contract with *Charlie*, and the latter cancels the contract with *Bob*. But *Bob* and *Eve* manage to steal *Charlie*'s processing fee. HTLC also leads to privacy leakage as we see that any two users in the path can trivially figure out that they are part of the same payment. This can be used for figuring out the identity of sender and receiver. If the transaction is settled on-chain, the complete route used for payment is revealed. We discuss some construction for single path and multipath payment which is privacy-preserving as well as quite efficient.

Fulgor and Rayo
We discuss here two protocols which deals with concurrency and privacy of payments in PCN. Before discussing the concurrency issue in PCN, we discuss a smart contract construction termed as multihop HTLC which preserves privacy of payment and hides the identity of payer and payee [33].

Multihop HTLC In Fig. 10, *Alice* samples four random numbers x_1, x_2, x_3, and x_4. It constructs $y_4 = \mathcal{H}(x_4)$ where $\mathcal{H}$ is any standard one-way hash function. Next, it constructs $y_3 = \mathcal{H}(x_3 \oplus x_4)$ and a zero-knowledge proof π_3 for the statement "*given* y_3 *and* y_4*, there exists an* $x : y_4 = \mathcal{H}(x)$ *and* $y_3 = \mathcal{H}(x_3 \oplus x)$." Similarly, it constructs $y_3 = \mathcal{H}(x_2 \oplus x_3 \oplus x_4)$ and a zero-knowledge proof π_2 for the statement "*given* y_2 *and* y_3*, there exists an* $x : y_3 = \mathcal{H}(x)$ *and* $y_2 = \mathcal{H}(x_2 \oplus x)$." It constructs $y_1 = \mathcal{H}(x_1 \oplus x_2 \oplus x_3 \oplus x_4)$ and a zero-knowledge proof π_1 for the statement "*given*

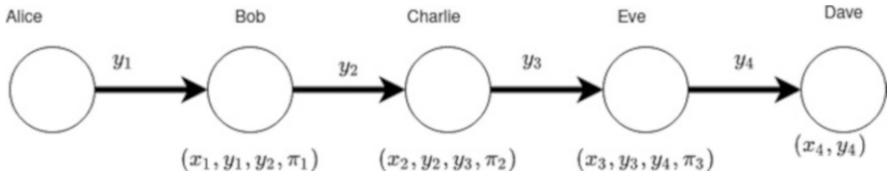


Fig. 10 Mutlihop HTLC construction for payment from *Alice* to *Dave*

y_1 *and* y_2*, there exists an* $x : y_2 = \mathcal{H}(x)$ *and* $y_1 = \mathcal{H}(x_1 \oplus x)$." It sends the value (x_1, y_1, y_2, π_1) to *Bob*, (x_2, y_2, y_3, π_2) to *Charlie*, (x_3, y_3, y_4, π_3) to *Eve*, and (x_4, y_4) to *Dave* via a secure anonymous channel.

Contract Creation Phase *Bob, Charlie,* and *Eve* check whether the zero-knowledge proof received is correct or not. *Alice* forms the contract with *Bob* using condition y_1. If π_1 is correct, *Bob* accepts the payment; else, he will abort. He forwards the payment to *Charlie* using the condition y_2. *Charlie* forwards the payment to *Eve* using the condition y_3, and *Eve* does the same to *Dave* using the condition y_4.

Contract Release Phase Upon receiving the conditional payment, *Dave* checks if $y_4 \stackrel{?}{=} \mathcal{H}(x_4)$. If this holds true, *Dave* sends x_4 to *Eve*, and claims payment. *Eve* calculates $x_3 \oplus x_4$, sends it to *Charlie* and claims payment. *Charlie* computes $x_2 \oplus x_3 \oplus x_4$ and claims payment from *Bob* upon releasing this key. *Bob* computes $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ and claims payment from *Alice*. None of the intermediate participants can correlate the payments as every channel uses a different condition.

Concurrency Issue in PCN It might be the case that several concurrent payments might enter a deadlock due to the lack of channel capacity. Consider the example shown in Fig. 11. *Alice* tries to send a payment of 1 unit to *Eve* via path $Alice \rightarrow Bob \rightarrow Charlie \rightarrow Dave \rightarrow Eve$, and simultaneously, *Mathew* wants to send a payment of 1 unit to *Charlie* via path $Mathew \rightarrow Dave \rightarrow Eve \rightarrow Bob \rightarrow Charlie$. Assuming each channel has a capacity of 1 unit, the payment from *Mathew* to *Charlie* gets blocked because the link *Bob-Charlie* has forwarded the payment request sent by *Alice*. Payment from *Alice* to *Eve* gets blocked because the link *Dave-Eve* has forwarded the payment request by *Mathew*.

The problem can be solved by either using the concept of *blocking payments* or *Fulgor* or the concept of *nonblocking payments* or *Rayo*. In Fulgor, both the payments stated in Fig. 11 are allowed to fail. The money locked in each contract is sent back to the party forwarding conditional payment. Each sender, here *Alice* and *Mathew*, waits for a random amount of time before reattempting the payment. Throughput decreases in this case. In Rayo, at least one of the concurrent payments is allowed to complete. A global ordering of payment is maintained. A user can queue payments with identifier higher than the payment currently being served and ignore all such payments with low identifier. Rayo provides nonblocking progress of payment and higher throughput but at cost of less anonymity. On the other hand, Fulgor provides stronger anonymity.

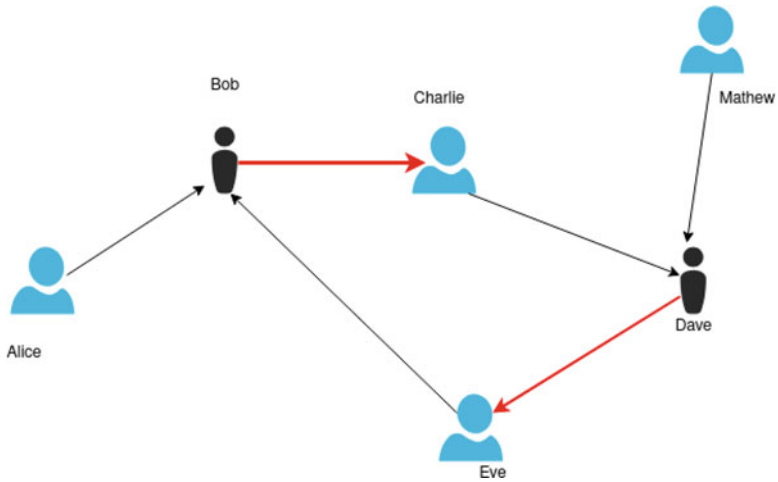


Fig. 11 Payment from *Alice* to *Eve* and from *Mathew* to *Charlie* enters a deadlock

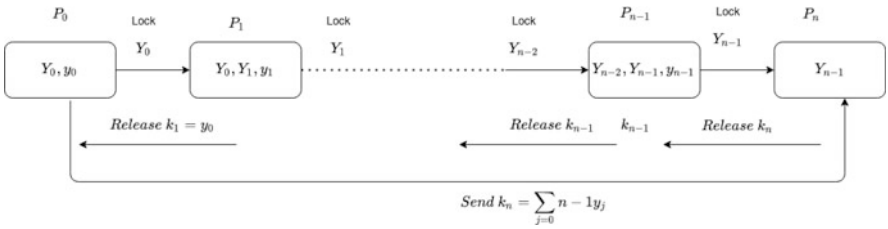


Fig. 12 Using AMHL for secure payment from P_0 to P_n

Anonymous Multihop Locks

Multihop HTLC requires exchanging nontrivial amount of data and computation of complex zero-knowledge proof during setup phase. It lacks interoperability since it requires that any cryptocurrency using it must support HTLC. Anonymous multihop locks (AMHL) [34] can be used for both script-based setting or even scriptless setting leveraging on Schnorr signature and ECDSA signature. It reduces the transaction size and blockchain load and guarantees interoperability.

We state a generic script-based construction of AMHL. Given a party P_0 wants to transfer an amount to party P_n via path $\mathcal{P} = \langle P_0, P_1, \ , P_n \rangle$ and a homomorphic one-way function $g : \mathcal{D} \rightarrow \mathcal{R}$, in the setup phase P_0 samples n values $(y_0, y_1, \ldots, y_{n-1})$. Using anonymous communication channel, P_0 sends $(g(\sum_{j=0}^{i-1} y_j), g(\sum_{j=0}^{i} y_j), y_i)$ to each of the party $P_i, i \in [1, n-1]$ along with a zero-knowledge proof π_i for the statement "$\exists y : g(y + y_i) = g(\sum_{j=0}^{i} y_j)$." P_0 sends a key $k_n = \sum_{j=0}^{n-1} y_j$ to P_n, as shown in Fig. 12. Each party checks whether the received message is consistent

using homomorphic properties of g. A pair of party P_i and P_{i+1} sharing a channel agrees on the common value $Y_i = g(\sum_{j=0}^{i} y_j)$ serving as the *lock*. Rest of the parties does the same. Once P_n receives the conditions of payment from P_{n-1}, it checks whether $Y_{n-1} \stackrel{?}{=} g(k_n)$. If this holds true, then it releases k_n to P_{n-1}. P_{n-1} upon receiving k_{n-1}, computes $k_{n-1} = k_n - y_{n-1}$ where $Y_{n-2} = g(k_{n-1}$, and sends this value to P_{n-3}. In general, this can be stated as parties P_i and P_{i+1} open the lock by using $g(k_{i+1} - y_i) = g(\sum_{j=0}^{i} y_j - y_i) = g(\sum_{j=0}^{i-1} y_j) = Y_i$.

For scriptless construction, we can use either Schnorr or ECDSA (compatible with Bitcoin) signature scheme. We discuss the Schnorr signature scheme.

Scriptless Schnorr Construction Given a message $m \in \mathcal{M}$, for a given public key pk, any two party can construct an *incomplete signature* over message m. This incomplete signature acts like the lock. Completion of signature acts like the release of the lock. For an elliptic curve group $\mathbb{G}$ having generator $\mathcal{G}$ of prime order q, we discuss the steps in details:

- Two parties P_i and P_{i+1} have public key $PK_i = x_0\mathcal{G}$ and $PK_{i+1} = x_1\mathcal{G}$ where $x_0 \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ and $x_1 \stackrel{\$}{\leftarrow} \mathbb{Z}_q$.
- Using the common lock value $Y_i = \sum_{j=0}^{i} y_j\mathcal{G}$, they chooses random value R_0 and R_1 where $R_0 = r_0\mathcal{G}$, created by P_i, and $R_1 = r_1\mathcal{G}$, created by P_{i+1}.
- Both the parties generate $R = R_0 + R_1 + Y_i$ and construct $s = r_0 + r_1 + e(x_0 + x_1)$ where $e = \mathcal{H}((PK_i + PK_{i+1})||R||m)$ and $\mathcal{H}$ is a standard hash function. The pair (R, s) forms the lock. It is not a valid signature on m, since it lacks the term $y : Y_i = y\mathcal{G}$.
- Once y is revealed, P_{i+1} can construct $s^* = s + y$ and complete the signature.
- Note that the lock between P_i and P_{i+1} is conditioned on the lock between P_{i+1} and P_{i+2}. Once these parties complete their signature and release the lock, only then P_{i+1} can complete the signature on P_i and P_{i+1}.

Atomic Multipath Payment

For high-valued payments, it might not be feasible to find a single path in a network to route a payment. Instead, it is advisable to split such payments into several micropayments. Routing each micropayment can be accomplished by using HTLC where the same hash value is reused across all the paths. The downside of an approach is that the payments can be correlated; it is susceptible to wormhole attack. Even if the payment fails in one of the paths, the receiver might still be able to claim the full payment as same preimage can solve all the conditional payments.

Atomic multipath payment [1] resolves these issues guaranteeing atomicity, either the payment succeeds or fails entirely. Once the receiver receives all the conditional payments from different routes, only then it can claim the payment. The

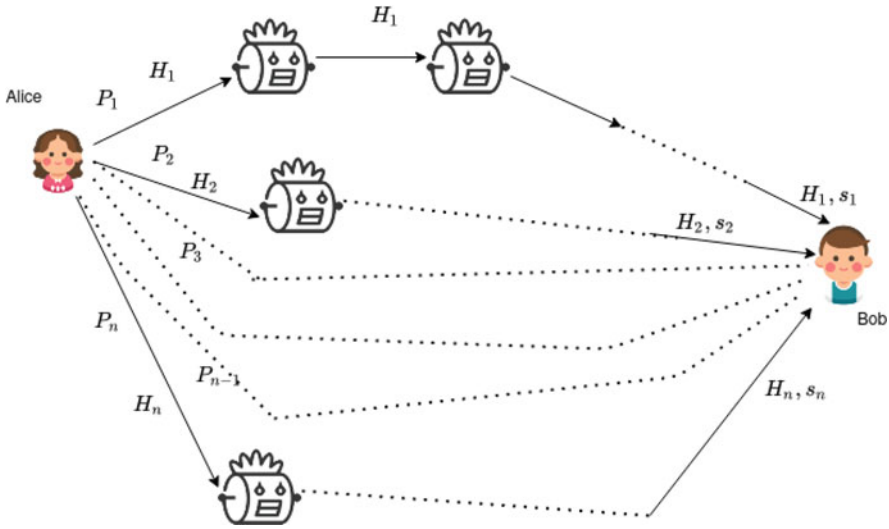


Fig. 13 Atomic multipath payment from *Alice* to *Bob*

payment hash used across different routes is different, preventing any correlation. The setup is noninteractive where the payer need not coordinate with the payee.

Construction *Alice* needs to send an amount v to *Bob*. It sends the amount via n paths, listed as $P_1, P_2, \ldots, P_n$. Each path P_i transferring $v_i : v = \sum_{i=1}^{n} v_i$ as shown in Fig. 13. *Alice* samples secret $s_1, s_2, \ldots, s_n$. The master secret $s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$ can be generated. Using s, she generates the condition of payment for each path P_i as follows: $H_i = \mathcal{H}(s||i), i \in [1, n]$. For each path P_i, the conditions of payment H_i is forwarded using onion routing, where the tuple (s_i, i) is sent as an encrypted onion blob (EOB) which can only be decrypted by *Bob*. Upon receiving all the conditions from n paths, *Bob* computes s and constructs the preimage $s||i$ for each path P_i in order to claim payment. If any of the path fails, then *Bob* will not be able to claim payment.

The drawback of the construction is the latency. The time taken by receiver to receive all the conditional payments is slowed down by the path having slowest propagation rate. Another multipath payment, *Boomerang* [7], addresses these issues by enabling redundancy. *Alice* can use more than the transaction value v for forwarding her payment via k such paths where $k > n$. Later, she can reclaim the funds that exceed v. If some of the paths fail, still the success rate of payment increases as the rest of the paths might be successful in propagating the desired value. The disadvantage of Boomerang is that due to the use of redundant paths, excess collateral gets locked up in paths which could have been utilized for routing other payments.

5.1.5 State Channels: Dealing with More than Simple Payments

A further extension of payment channel is state channels, with facility of handling arbitrary complex smart contracts. Allowing execution of any off-chain transaction leads to change in state of the channel. A new state invalidates the last state which can be proven cryptographically in blockchain if there is dispute. It has got myriad of application in online gaming industry, digital content distribution, asset transfer, and asset exchange. A state channel can be opened by more than two parties as well.

We discuss few variants of state channels and their constructions:

Sprites

Routing payment Lightning Network is susceptible to *griefing attack* where a receiver can stop responding. If the path routing payment from payer to payee is of length n and the time taken to settle a transaction on-chain is Δ, then in the worst-case if all parties go on-chain and settle the payment, the collateral across the path is locked for $O(n\Delta)$. To reduce the worst-case collateral locked for time period of $O(\Delta)$, a variant of payment channel construction supporting Ethereum-styled smart contracts, Sprites [36] was proposed. Sprites supports partial withdrawals and deposits, and the channel can continue to operate without interruption while such operations are being performed.

In Fig. 14, each party P_i opens a channel with its counterparty $P_{i+1}, i \in [1, n-1]$ by making a deposit and locking the amount in a multi-sig account, recording it on-chain. Suppose party P_1 wants to transfer amount f to party P_n. Ignoring any processing fee charged by intermediate party $P_j, j \in [2, n-1]$, a smart contract called as preimage manager (PM) is used where an assertion of the form *the preimage of x of hash H=$\mathcal{H}(x)$ gets published in the blockchain before T_{expiry}*.

P_n samples x and produces H and shares it with P_1. Next, P_1 forms the global contract PM and broadcasts it on-chain. Each party P_i establishes off-chain contract with P_{i+1} using the same condition H as used in PM, where P_{i+1} cancels the contract any moment if things go wrong, and in case of dispute, the global smart contract PM comes to the rescue. Once P_n receives the conditional payment, it will start resolving payment by releasing x. This solution is used by each party to solve the contract and claim payment, taking around $O(n)$ rounds. An honest party will definitely broadcast x where it takes time Δ units for the transaction to

Fig. 14 Payment across multiple channels in Sprites

get confirmed on-chain. This ensures that if a party acts honestly, it will definitely receive x before time $T_{expiry} - \Delta$. In case of dispute in any of the channel, since the solution is available, the payment gets completed by T_{expiry}. If receiver doesn't release the preimage, then no solution gets broadcasted; hence, all the conditional payment will get canceled. In the worst case, the time taken to settle the payment is $O(n + \Delta)$. This is different from what Lightning Network does, since it relies on each party to broadcast the transaction and settle each transaction on-chain, taking time $O(n\Delta)$.

Incremental Deposit and Withdrawal Sprites permit withdrawal and deposit of funds without going for repeated opening and closing of channel. The off-chain state is extended to maintain states $Deposit_{\{L,R\}}$ which shows the local view of how much each party has deposited, recorded in the smart contract. If a party produces a stale view of the deposit, having a timestamp of around Δ units or more behind the current timestamp, then other parties can raise a dispute on-chain. The current state can be read and funds get settled accordingly.

Similarly, an optional extended off-chain state for withdrawal is marked as wd. If a party wishes to make withdrawal, it sets this state to true. The on-chain smart contract is provided with an update method, which, if invoked, allows parties to submit signed message for withdrawal of money from the channel. Note that this does not lead to closure of channel; rather, the signature is verified, and the party gets refunded as per the amount stated for withdrawal.

General State Channel Networks

We first discuss the construction of ledger state channel followed by a brief overview of how virtual state channel can be constructed by leveraging on such state channels [20].

Ledger State Channel Two parties can open a state channel by locking money in a smart contract termed as state channel contract (*SCC*). After deployment of the contract, the channel is said to be established. Now, these parties can run multiple instance of an application-specific contract in this channel.

Suppose *Alice* and *Bob* have deposited each 5 units of coin in *SCC*. Now, they want to play a game of lottery where each party place a bet of 1 unit and select a bit either 0 or 1. If the result of the xor of bits selected by *Alice* and *Bob* is 1, *Alice* gets 2 units of coin, and if the result is 0, *Bob* gets 2 units of coin. The terms of the game are encoded in the smart contract $C_{lottery}$. This contract is instantiated off-chain with each party locking 1 unit into the contract instance. Let the instance executed in the channel be termed as S. This is the initial state of the contract termed as S_0, and the version number is set to 0. Both parties now exchange signature over $(S_0, 0)$. Next, *Alice* selects a bit r_A and submits the commitment of the bit to *Bob* along with the new state of the game $(S_1, 1)$. He signs the state and sends his bit r_B along with the new state of the contract, termed $(S_2, 2)$ and signed state $(S_1, 1)$. *Alice* signs $(S_2, 2)$, and using the value r_B, she decommits her bit and computes $r_A \oplus r_B$. Note

that all the computation and change of state are local and not recorded on-chain. She now sends her bit r_A along with the computed value and the state $(S_3, 3)$ along with the signed state $(S_2, 2)$. *Bob* now checks whether the decommitment of r_A is correct and computes $r_A \oplus r_B$. If the result is 0, then he claims 2 units, and if 1, *Alice* claims 1 unit. This happens when both parties are honest and terminates the game off-chain. *Bob* signs $(S_3, 3)$ and sends it to *Alice* which marks the end of the game.

For the bad case, suppose one party stops responding or if *Alice* sends a wrong decommitment of the bit r_A. In that case, *Bob* refuses to sign the latest state of the contract. He now fall backs on the state channel contract in order to resolve the dispute by providing the latest signed state of the contract. If *Bob* is malicious and tries to submit a stale state of the contract, then *Alice* can overwrite the same by submitting a higher version of the contract. *SCC* finishes the game based on the latest signed state of the contract and settles the dispute on-chain.

Virtual State Channel Virtual state channels can be built recursively on top of state channel. Consider two parties, *Alice* and *Bob*, who want to run a contract code off-chain but do not share a state channel. They are connected to each other via *Charlie*, i.e., *Alice* has a channel with *Charlie*, and *Charlie* has a channel with *Bob*. *Alice* and *Bob* can open a virtual state channel γ with *Charlie* in between, without ever recording the opening of channel in blockchain. The virtual state channel functions like a state channel between *Alice* and *Bob*. It can also run any contract instance. If both the parties are honest, then the contract gets executed off-chain without involving *Charlie*. If not, then special virtual state channel contracts are used which leverage on the channel α (between *Alice and Charlie*) and β (between *Charlie and Bob*). We discuss the construction in details.

Creation of Virtual State Channel of Length 2 *Alice* and *Bob* form a mutual agreement with *Charlie* to form the virtual state channel γ, as shown in Fig. 15.

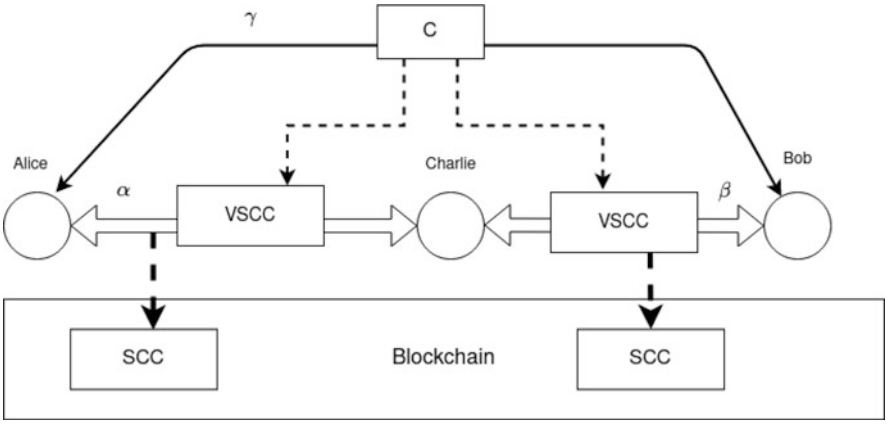


Fig. 15 Construction of a virtual state channel γ of length 2

To prevent cheating, each party creates a special contract in its respective channel, termed as virtual state channel contract (*VSCC*). *Alice* requests opening an instance of VSCC, termed as v_α in channel α. Similarly, *Bob* requests opening an instance of VSCC, termed as v_β in channel β. Both VSCC instances v_α and v_β will have complete information of the channel γ. In v_α, *Charlie* will play the role of *Bob*, and in v_β, *Charlie* will play the role of *Alice*. For example, if *Alice* and *Bob* want to form the channel γ, with *Alice* locking 5 units of coin and *Bob* locking 5 units of coin, then in v_α, *Alice* locks 5 units, and *Charlie* locks 5 units. In v_β, *Charlie* locks 5 units and *Bob* locks 5 units. If *Charlie* accepts both the request, then the channel γ gets instantiated.

Once the channel gets instantiated, *Alice* and *Bob* can instantiate any contract code. The example of lottery game stated in ledger state channel can be executed here as well. We discuss how dispute can be handled. Suppose *Bob* doesn't send his signature in state $(S_3, 3)$. In that case, *Alice* will inform *Charlie* about the dispute by executing the contract instance v_α. *Alice* sends the state $(S_2, 2)$ along with the signature of *Bob* on the state to v_α by calling the function *register*. To verify the claim made by *Alice*, *Charlie* sends $register(S_2, 2)$ along with the signature of *Bob* to v_β. If *Bob* can respond with a state of the contract having version number higher than that submitted by *Charlie* with *Alice*'s signature, then the state S_2 will be discarded by the contract. Else, S_2 gets accepted as the latest state of the contract, and both v_α and v_β reach a consensus on it. This state now serves as the latest state for any function call made on the contract instances v_α and v_β.

If *Charlie* is malicious and colludes with *Bob*, then *Alice* can consider S_2 as the latest state of the contract and execute the contract instance v_α by falling back on *SCC*, instantiated on channel α. So for the pessimistic execution, the run times grow linearly with the number of intermediaries involved. Time complexity is $O(\Delta)$.

Constructing Longer Virtual State Channel Longer virtual state channel can be constructed, applying recursion. As shown in Fig. 16, once a virtual state channel is

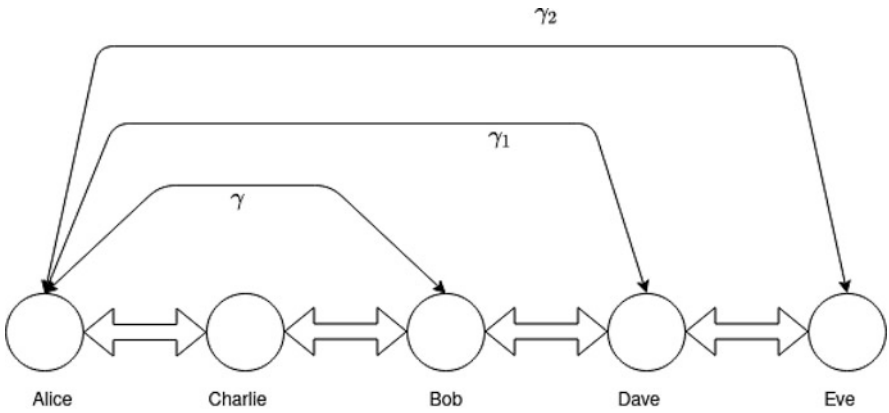


Fig. 16 Construction of a virtual state channel γ_2 of length 4

established between *Alice* and *Bob*, another virtual channel between *Alice* and *Dave* can be established using virtual channel γ and the ledge state channel between *Bob* and *Dave*. Let us denote this virtual state channel as γ_1. This can be extended to construct a virtual channel between *Alice* and *Eve* as well leveraging on γ_1 and state channel between *Dave* and *Eve*. Not only just using the combination of state channel or virtual channel and state channel, a longer virtual channel can be constructed on top of multiple shorter virtual channels as well. If a virtual channel existed between *Bob* and *Eve*, then using γ and the latter, γ_2 can be constructed as well.

When the virtual channel is of length n, it takes around $O(n\Delta)$ to resolve dispute in the pessimistic case. We discuss how the worst-case timing was reduced to $O(\Delta)$ in the next part.

Virtual Payment Channel with Direct Dispute Given the virtual state channel between *Alice* and *Eve* as shown in Fig. 16, suppose it is built recursively using the virtual state channels between *Alice* and *Bob* and another one between *Bob* and *Dave*. If *Alice* misbehaves and doesn't acknowledge the latest state of the contract, then *Dave* will try to resolve via *Bob* as per the dispute resolution handling stated previously for channel of length 2. Instead of that, it resolves the dispute directly with the help of a global *dispute board*, denoted as $\mathcal{D}$. It is a contract running directly on the ledger whose state can be accessed by anyone. Similar to state registration stated previously, *Eve* registers her state in $\mathcal{D}$ regarding the channel γ_2 [18]. *Alice* is notified and gets a time period of Δ to send her own version of latest state to $\mathcal{D}$. All the parties can see the final result of the dispute. With the introduction of the dispute board, the closing of virtual state channel becomes complicated since any dispute must be resolved on $\mathcal{D}$. The final state registered is then used for closing of γ_2.

Multiparty Virtual State Channels

The requirement for multiparty virtual state channel construction between any subset of parties existing in the set $\{P_0, P_1, \ldots, P_n\}$ is that there must exist a ledger state channel between parties P_i and $P_{i+1}, i \in [0, n-1]$ [18]. It is quite possible that one of the parties might not be interested in participating in the process.

Consider a situation where *Alice*, *Bob*, *Dave*, and *Eve* want to construct a multiparty virtual state channel γ, but *Charlie* is not interested. Let each of the interested parties deposit 5 units of coin in γ as shown in Fig. 17. *Alice* and *Bob* first construct a virtual state channel with *Charlie* agreeing to the construction. Using the virtual state channel between *Alice* and *Bob* and ledger state channels between *Bob-Dave* and *Dave-Eve*, we discuss how multiparty virtual state channel can be constructed. In each of these channels, an instance of the special *multiparty virtual state channel contract*, termed as *mpVSCC*, is opened. Each of these instance is the copy of the information retained in *mpVSCC*. In channel γ, the amount deposited by each party is 5 units, and thus the capacity of the channel is 20. This information is replicated in channel *Alice-Bob*: *Alice* deposits 5 units, and *Bob* (playing the role of

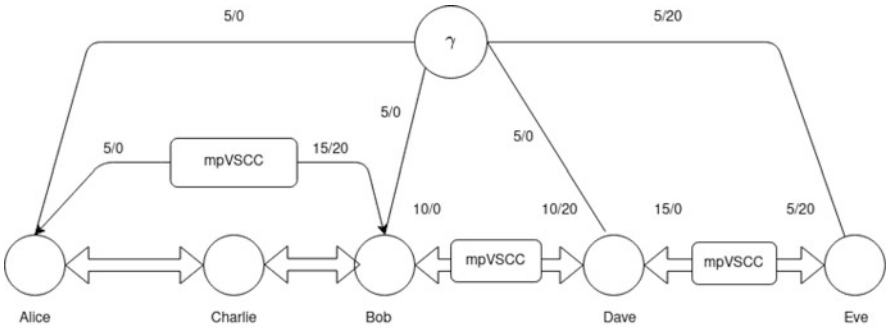


Fig. 17 Construction of a multiparty virtual state channel γ between *Alice*, *Bob*, *Dave*, and *Eve*

Bob, *Dave*, and *Eve*) deposits 15 units in the contract mpVSCC. In the ledger state channel *Bob-Dave*, *Bob* (playing the role of both *Alice* and *Bob*) deposits 10 units, and *Dave* (playing role of *Dave* and *Eve*) deposits 10 units in the contract mpVSCC. In the ledger state channel *Dave-Eve*, *Dave* (playing role of *Alice*, *Bob*, and *Dave*) deposits 15 units, and *Eve* deposits 5 units in the contract mpVSCC.

Once γ is created, *Alice*, *Bob*, *Dave*, and *Eve* can execute multiple contract instances without involving *Charlie*. Suppose a game is played between these four parties so that if *Eve* ends the game, it can claim all the 20 coins. Given that all parties played honestly, *Eve* owns all the 20 coins. Once γ is closed, the coins locked in mpVSCC in the subchannels are unlocked as per the final balance of γ. *Dave* loses 15 coins but gets back 20 coins from the mpVSCC in channel *Bob-Dave*. Since it had deposited 15 coins in channel *Dave-Eve* and 10 coin in channel *Bob-Dave* and gains 20 coins, that means it has lost 5 coins. *Bob* had deposited 10 coins in channel *Bob-Dave* and 15 coins in channel *Alice-Bob*. It gets back the 15 coins, loses 10 coins to *Dave*, and gets back 5 coins from *Alice*. *Alice* loses 5 coins to *Bob* in mpVSCC in channel *Alice-Bob*. Hence, except *Eve*, each party has lost 5 coins.

Update and Execution in Multiparty Virtual State Channel While executing the contract instance, say *Alice* executes it locally, gets a new state, and requests for state update of the channel γ by sending her signature on it. It might happen that other parties say *Eve* or *Dave* might also be executing their contract instance locally and request for state update. This might lead to race condition. To avoid such a situation, an execution scheduling is used which leads to a constant time protocol.

In case of dispute over state updation, all parties involved can opt for state registration using the direct dispute process. Suppose *Alice* wants to update a state and in the process it sends one state to *Bob* and *Eve* and a different state to *Dave*. Since the rest of the parties now have different view of the state, the inconsistency can be resolved in $O(1)$ round by taking help of the dispute board where the party that submits the latest contract state is accepted.

5.2 Sidechain

Sidechains are independent blockchains, allowing assets and currency to be transferred across multiple blockchains. These sidechains employ their own consensus models and block parameters to efficiently process transactions. In case of damage to the sidechain, the main blockchain remains unaffected. We define *pegged sidechain* in the next section.

5.2.1 Pegged Sidechains

A sidechain is attached to a public blockchain with a two-way peg, the latter being termed as the parent chain. Two-way peg refers to the mechanism by which coins are transferred between sidechains and parent chain at a fixed or otherwise deterministic exchange rate. Simplified payment verification peg (SPV peg) serves as a symmetric two-way peg. It works as follows: coins from parent chain can be transferred to the sidechain by sending those coins to a special output. The output can be unlocked using an *SPV proof of possession* in the sidechain. *SPV proof* (simplified payment verification proof) is composed of a list of blockheaders demonstrating proof of work and a cryptographic proof used for proving that an output was created in one of the blocks in the list (proof of inclusion in the form of a Merkle Branch). This allows verifiers to check that some amount of work has been committed to the existence of an output, without having to download the entire chain, starting from the genesis block. They download the block headers of the entire blockchain, which are much smaller in size than the actual block itself.

To synchronize the parent chain and sidechain, there are two waiting periods:

- *The Confirmation Period*: Before transferring coins from parent chain to sidechain, it must be locked in the parent chain. This period is allowed for sufficient work to be created in order to prevent denial-of-service attack in the next waiting period. After creating the special output in the parent chain, it waits for this period before a transaction gets created on the sidechain referencing this output, providing a corresponding *SPV proof* that it was created in the parent chain. The transaction in the parent chain must wait for finality.
- *The Contest Period*: After transferring the newly transferred coins to the sidechain, it must wait for certain amount of time before spending the same. Suppose during this period a new proof is published with a chain containing more aggregate work. However, if this chain doesn't contain the block in which the locked output was created in the parent chain, then this conversion is invalidated. This is termed as *reorganization proof*. All users of the sidechain have an incentive to produce reorganization proofs. Submitting a bad proof leads to dilution of the value of coins.

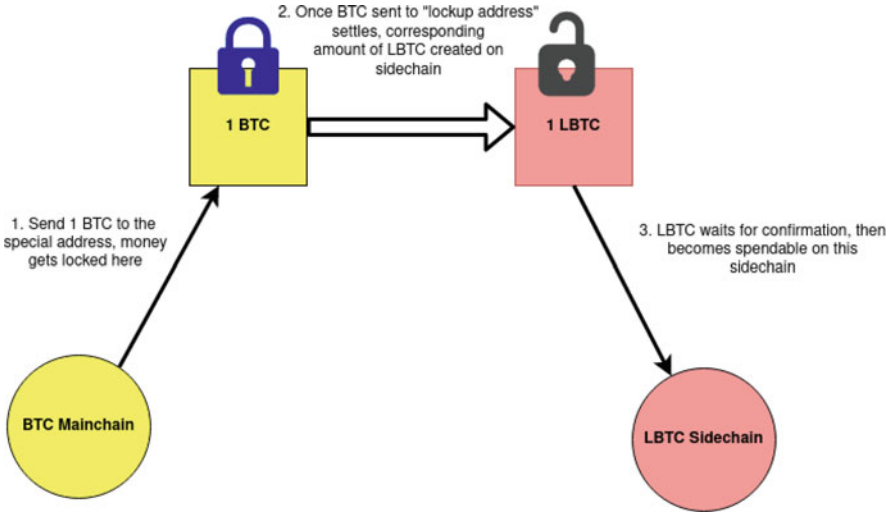


Fig. 18 Liquid sidechain

As shown in Fig. 18, Bitcoins from the parent chain are transferred to the *liquid sidechain*. Liquid requires 102 block confirmation before the LBTC minted becomes available for spending [21].

If a user wants to transfer coins from sidechain back to parent chain, the same procedure is followed. The coins in sidechain are locked in special output, where the output can be claimed upon providing an *SPV proof*. This can be used to unlock a number of previously locked outputs with equal denomination on the parent chain. While locked on the parent chain, the coin can be freely transferred within the sidechain without further interaction with the parent chain. However, except the parent chain, the sidechain cannot transfer this coin to any other blockchain since it retains the identity of parent chain's coin.

For *asymmetric two-way peg sidechain*, users of the sidechain download the full blockchain and do not act as light client. Hence, there is no need of *SPV proofs* as all validators are aware of the state of the parent chain. But the parent chain is still unaware of the sidechain, so *SPV proofs* are required to transfer back.

The two-way pegged sidechain are however not devoid of faults. It is quite slow as a user needs to wait for confirmation and contest periods before having access to his or her funds on either parent chain or sidechain. There is also a risk of centralization of mining as well as soft fork. However, these problems can be mitigated by the use of federated or multi-signature two-way pegs which is more decentralized.

Consensus Sidechains often incorporate alternate validator selection and consensus mechanisms to provide faster transaction confirmation times and increased throughput. This is a stark departure from the properties of layer 2 where off-chain transaction falls back upon the security of layer 1. It uses a variety of validator selection methods to achieve these goals but not compromising on security.

Selection of validators is done based on either *proof of authority* or *proof of stake*. If an honest party loses coins in sidechain, the dispute cannot be resolved in parent chain since sidechain is responsible for its own security [61].

Pros and Cons Sidechains are permanent and need not be closed. Participants can be added without creating a new chain, unlike state channel where every participant willing to join the network has to create a new state channel. Testing of new applications can be done on sidechain before moving it to parent chain. However, investment made for sidechain is substantial [55]. A lot of miners are needed so that network is safe from attackers. Interoperability across multiple blockchain is difficult unlike the use of smart contracts in state channel for running an application.

5.3 Commit Chain

Though it is simple to execute payments in PCN, it has several problems, for example, any party willing to join the network has to open a channel with an existing party in the network. Funds locked in the channel cannot be used for other purpose, unless it is relinquished upon the closure of channel. Parties remaining offline cannot receive any fund transfer. To alleviate such problems, *commit chain* was introduced as an off-chain scaling solution [28]. A commit chain comprises a *noncustodial operator* which processes the payments across users off-chain and a smart contract deployed on-chain enforcing correct behavior of the operator. The operator commits the latest state of the user's account balances to the smart contract in regular intervals via *constant-sized checkpoints*. Regular checkpointing allows users to challenge the operator in case of dispute. Correct operation can be enforced by the smart contract upon verification. Commit chain relies on the consensus of the parent chain unlike sidechain, making it as secure as the parent chain.

The life cycle of commit chain is defined as follows [28]:

- Register: A user can join the commit chain by exchanging authenticated off-chain messages with the operator. It can also receive payments by remaining offline and need to resurface once in an interval between two checkpoints.
- Deposit funds: Users lock the desired amount of coins in a smart contract. These funds can be sent to any other user off-chain and can be claimed back, whenever required.
- Transfer: If a user requests a fund transfer to a recipient, it asks the operator to authorize this action.
- Withdraw: A user can withdraw funds from the commit chain by submitting a withdrawal request to the operator. After verifying the request, the operator authorizes the user. The latter submits this request to the smart contract. If there is no dispute regarding the withdrawal, then in the next round, contract credits the balance of user with the funds deposited in the parent chain. The process can be expedited if operator pays the user from his own personal funds and later submits a withdrawal request to the contract.

- Exit: A user can exit the commit chain by submitting a request to the contract. The operator is alerted regarding this event. It verifies the request and provides a proof of exclusive exit. The user uses this proof for claiming funds deposited in the parent chain.
- Checkpointing: After a given period of time, the operator commits the latest state of all user accounts to the smart contract. Upon finalization, these changes get recorded on-chain. Collateral used in a checkpoint can be reused in the future checkpoints. This amount can be distributed among all the participants using a single on-chain transaction.
- Challenge: A user publishes challenges on the smart contract if it detects that an operator has misbehaved. If the operator produces invalid response, then the operator is penalized by slashing of funds.
- Recover: Once an operator gets penalized for misbehavior, the smart contract stops accepting checkpoints. Users can recover as per the state update provided in the last valid checkpoint via on-chain transactions.

We discuss few properties of commit chain:

- Periodic Checkpoint Commitments: On-chain checkpoint commitments can be instantiated using either zero-knowledge proof (ZKP) or Merkle tree root. Participants can resurface online periodically to check whether the commitment is valid or not. Verification of ZKP is easier than Merkle tree root; however, proof construction is complex in the former case.
- Data Availability: Commit chain data is not broadcasted on-chain while checkpointing; however, users need to maintain the data or have to retrieve the same while exiting the commit chain or in case of dispute resolution.
- Centralized but Untrusted Operator: A single untrusted operator is susceptible to a single point of failure. However, this operator doesn't have custody of funds of any of the participants.
- Eventual Transaction Finality: Commit chain doesn't offer instant transaction finality unlike PCN. Finality is guaranteed eventually during periodic on-chain checkpointing. However, here, funds need not be routed via set of intermediaries between parties not connected directly unlike in PCN.

The most prominent commit chain is Nocust [25], which follows an account-based model. The on-chain address is associated with a commit chain account. The noncustodial operator must periodically submit constant-sized commitment of the state of the commit chain ledger to the on-chain contract. This commitment contains each user's account present in the collateral pool.

5.3.1 Plasma

Plasma is a commit chain but has its own consensus mechanism different from the parent chain, just like sidechain [43, 44]. One can think of plasma as series of smart contracts allowing multiple blockchain to exist, being dependent on one parent

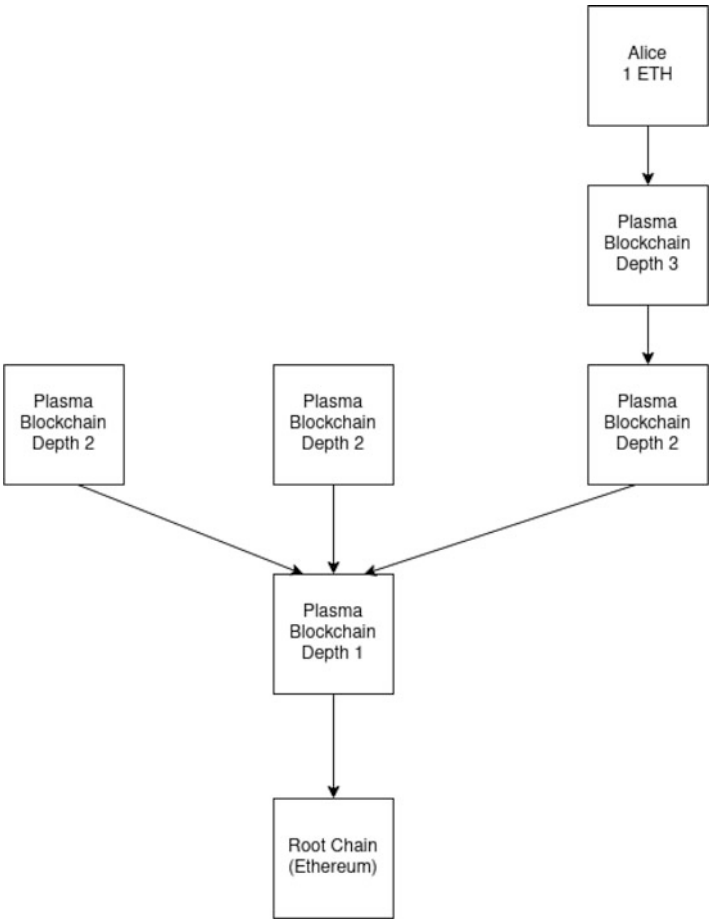


Fig. 19 Plasma as collection of blockchains

chain. Many blockchains can have smart contract with their own business logic, but the main chain enforces correct behavior across these blockchains. Deposit and withdrawal of funds from the plasma chain with state transition are allowed by submission of transaction proofs.

In plasma, the blockchains are composed in a tree format with the root being the root blockchain, say Ethereum, as shown in Fig. 19. Since the chains are organized in a hierarchy, the intermediate blockchains act as a parent chain to their child chains. The main blockchain is thus referred to as the root chain.

The funds held in the plasma block must be accounted in the parent chain periodically and propagated to the root chain. The contents of the plasma block are not disclosed. Rather, block hashes are submitted which acts like a summary of the state transitions. A Merkle tree is constructed by the noncustodial operator,

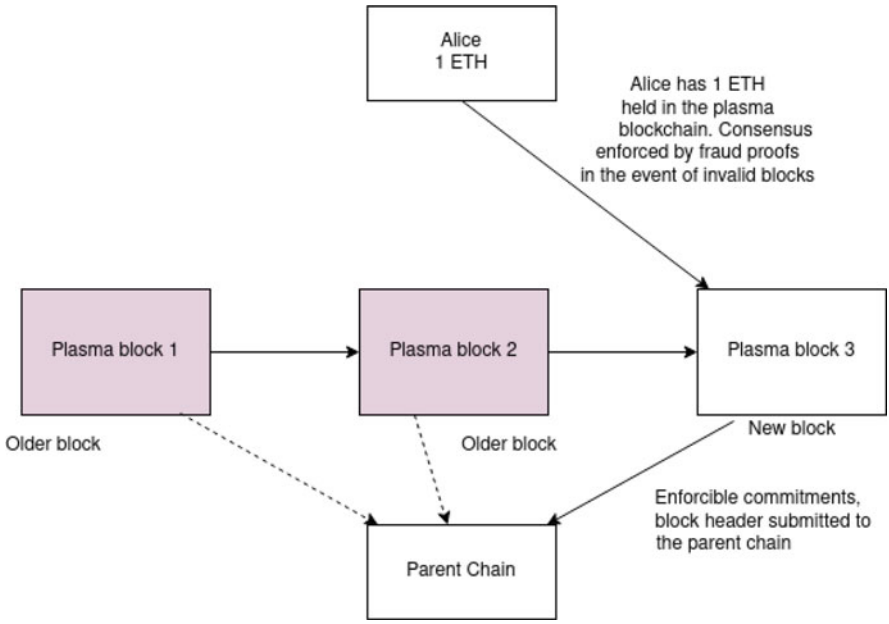


Fig. 20 *Alice* submits a transaction depositing 1 ETH in plasma chain. Root chain is unaware of the balance of *Alice*, but the balance of the entire plasma chain is updated in the root chain

and the state of the plasma chain is represented by the Merkle root. In Fig. 20, a magnified view of plasma chain in depth 3 is shown where *Alice* submits a transaction depositing 1 ETH in the plasma chain.

In the event of a block withholding attack, if a user wants to exit, he or she can submit exit transaction along with the Merkle proof of the ownership of funds to the parent chain in the next level which ultimately gets propagated to the root chain. As shown in Fig. 21, *Alice* cannot validate *block 4* due to block withholding attack. She exits by broadcasting a proof of funds on the root blockchain, and her withdrawal is processed after a certain time period within which this withdrawal can be disputed.

Fraud proofs present an evidence that a state transition in the child blockchain is not valid, presuming block data availability. The root blockchain possesses collection of fraud proofs. These proofs ensure that all state transitions are valid when block data is available.

A party willing to exit the plasma chain attests to a bitmap of the participant's ledger output arranged in a UTXO model requesting a withdrawal. After submission of the request, it has to wait for a certain interval time. This time, interval is called as the *challenge period*. Within this interval, any user, termed as *challenger*, can submit a fraud proof if it finds that the funds have already been spent. If the proof is correct, the challenger is rewarded with the bond deposited by the exiting participant. This ensures that even if plasma chain is compromised, rest of the users can be assured that the states/assets on this sidechain are safe [2]. After the interval has passed,

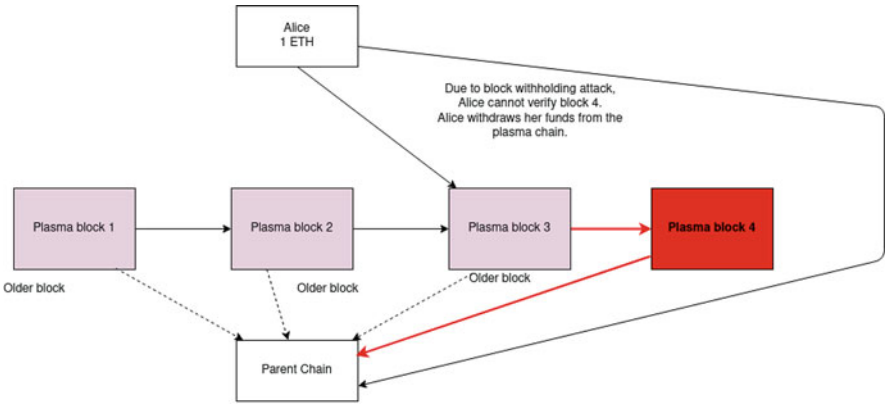


Fig. 21 *Alice* withdraws money from the plasma chain

another bonded round allows the withdrawal to take place. It bonds to a state before committing to a timestamp.

A block leader can propose a block for the plasma chain. If the block has an invalid state, any participant in this chain upon receiving the block can submit a fraud proof to the root chain. The invalid block is rolled back and the block proposer is penalized.

Several different versions of plasma currently in existence are *Plasma Cash, Plasma Debit, Plasma Prime, Plasma Snapp, Plasma Bridge*, and *Minimum Viable Plasma* [2]. Some of the popular platforms using *plasma* are *OMG Network* [40], *Matic Network* [35], *Gluon* [23], and *Gazelle* [22].

Disadvantages Scalability in plasma is achieved by allowing several plasma transactions to be processed with only single hash of the state update of the plasma chain being added to a parent chain. Some of the disadvantages faced by plasma chain include the following:

- Longer waiting period for users who wish to exit the plasma chain is a major drawback. The challenge period is set for 1 week during which any user can challenge another user's exit by providing a proof that the exit is invalid. Only after a week, the exits can be processed.
- Users must retain high amount of data in order to construct fraud proofs since root chain doesn't retain any data.
- If an operator withholds data, user needs to submit the last valid state in order to exit the chain. If it is the case that all the users want to exit the plasma chain, then the entire valid state of the chain has to be submitted to the root chain. A lot of data will be dumped into the latter, which might lead to congestion in the network and collapse of plasma chain [58]. This is termed as *problem of mass exit*.

6 Conclusion

In this chapter, we state the scalability issue in blockchain and discuss some of the off-chain scaling solutions in details, categorized into payment and state channels, sidechains, and commit chains. The benefit offered by layer 2 solutions is that improvement in performance can be achieved without making any change in the base layer. It is observed that finality of transaction is inversed to the amount of collateral locked. Transactions in channels ensure instant finality due to the huge amount of collateral locked. In commit chains and sidechain, the amount of collateral locked is reduced in exchange of eventual finality.

References

1. Amp: Atomic multi-path payments over lightning (2018). https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html
2. An introduction to plasma (2018). https://hackernoon.com/plasma-8bba7e1b1d0f
3. L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, S. Riahi, Bitcoin-compatible virtual channels. IACR Cryptol. ePrint Arch. **2020**, 554 (2020)
4. L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, S. Riahi, Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. **2020**, 476 (2020)
5. A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains. http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains **72**, 201–224 (2014)
6. V. Bagaria, S. Kannan, D. Tse, G. Fanti, P. Viswanath, Prism: Deconstructing the blockchain to approach physical limits, in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), pp. 585–602
7. V. Bagaria, J. Neu, D. Tse, Boomerang: redundancy improves latency and throughput in payment networks (2019). arXiv preprint arXiv:1910.01834
8. S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, G. Danezis, Consensus in the age of blockchains (2017). arXiv preprint arXiv:1711.03936
9. [bitcoin-development] anti dos for tx replacement (2013). https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html
10. Block size limit controversy (2015). https://en.bitcoin.it/wiki/Block_size_limit_controversy
11. C. Burchert, C. Decker, R. Wattenhofer, Scalable funding of bitcoin micropayment channel networks. R. Soc. Open Sci. **5**(8), 180089 (2018)
12. V. Buterin, Toward a 12-second block time (2014). https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/
13. J. Coleman, L. Horne, L. Xuanji, Counterfactual: generalized state channels. Acessed: https://l4.ventures/papers/statechannels.pdf **4**, 2019 (2018)
14. K. Croman, C. Decker, I. Eyal, A.E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E.G. Sirer, et al., On scaling decentralized blockchains, in *International Conference on Financial Cryptography and Data Security* (Springer, Berlin, 2016), pp. 106–125
15. C. Decker, R. Russell, O. Osuntokun, eltoo: A simple layer2 protocol for bitcoin. White paper: https://blockstream.com/eltoo.pdf (2018)
16. C. Decker, R. Wattenhofer, A fast and scalable payment network with bitcoin duplex micropayment channels, in *Symposium on Self-Stabilizing Systems* (Springer, Berlin, 2015), pp. 3–18

17. M. Dong, Q. Liang, X. Li, J. Liu, Celer network: Bring internet scale to every blockchain (2018). arXiv preprint arXiv:1810.00037
18. S. Dziembowski, L. Eckey, S. Faust, J. Hesse, K. Hostáková, Multi-party virtual state channels, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 2019), pp. 625–656
19. S. Dziembowski, L. Eckey, S. Faust, D. Malinowski, Perun: virtual payment channels over cryptographic currencies. IACR Cryptol. ePrint Arch. **2017**, 635 (2017)
20. S. Dziembowski, S. Faust, K. Hostáková, General state channel networks, in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (ACM, New York, 2018), pp. 949–966
21. Exploring liquid sidechain. https://blog.goodaudience.com/overview-7b9ea0b0d5af
22. Gazelle. https://gzle.io/
23. Gluon network. https://gluon.network/
24. Grayblock: Blockchain scaling: Why PoW networks can't scale (2018). https://medium.com/coinmonks/blockchain-scaling-30c9e1b7db1b
25. L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, Sok: Layer-two blockchain protocols, in *International Conference on Financial Cryptography and Data Security* (Springer, Berlin, 2020), pp. 201–226
26. J. Harris, A. Zohar, Flood & loot: A systemic attack on the lightning network (2020). arXiv preprint arXiv:2006.08513
27. J. Herrera-Joancomartí, G. Navarro-Arribas, A. Ranchal-Pedrosa, C. Pérez-Solà, J. Garcia-Alfaro, On the difficulty of hiding the balance of lightning network channels, in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security* (2019), pp. 602–612
28. R. Khalil, A. Zamyatin, G. Felley, P. Moreno-Sanchez, A. Gervais, Commit-chains: Secure, scalable off-chain payments. Tech. rep., Cryptology ePrint Archive, Report 2018/642, 2018. https://eprint.iacr.org (2018)
29. S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August **19** (2012)
30. W. Li, S. Andreina, J.M. Bohli, G. Karame, Securing proof-of-stake blockchain protocols, in *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (Springer, Berlin, 2017), pp. 297–315
31. Z. Lu, R. Han, J. Yu, Bank run payment channel networks. IACR Cryptol. ePrint Arch. **2020**, 456 (2020)
32. L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, A secure sharding protocol for open blockchains, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (ACM, New York, 2016), pp. 17–30
33. G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, S. Ravi, Concurrency and privacy with payment-channel networks, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (ACM, New York, 2017), pp. 455–471
34. G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, M. Maffei, Anonymous multi-hop locks for blockchain scalability and interoperability, in *NDSS* (2019)
35. Matic network. https://matic.network/
36. A. Miller, I. Bentov, R. Kumaresan, P. McCorry, Sprites: Payment channels that go faster than lightning, in *Twenty-Third International Conference on Financial Cryptography and Data Security 2019* (2019)
37. S. Nakamoto, Bitcoin whitepaper (2008)
38. K.J. O'Dwyer, D. Malone: Bitcoin mining and its energy footprint (2014)
39. T. Ometoruwa, An introductory guide to hashed timelock contracts (2019). https://btcmanager.com/an-introductory-guide-to-hashed-timelock-contracts/
40. Omg network. https://omg.network/

41. iryna pavlenko: Block size limit controversy (2018). https://applicature.com/blog/blockchain-technology/blockchain-scalability
42. Payment channels (2019). https://en.bitcoin.it/wiki/Payment_channels
43. Plasma. https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/plasma/
44. J. Poon, V. Buterin, Plasma: Scalable autonomous smart contracts. White paper (2017), pp. 1–47
45. J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments (2016)
46. Proof of capacity (cryptocurrency) (2020). https://www.investopedia.com/terms/p/proof-capacity-cryptocurrency.asp
47. Proof of elapsed time (poet) (cryptocurrency) (2020). https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp
48. P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, O. Osuntokun, Flare: An approach to routing in lightning network. White Paper (2016). bitfury.com/content/5-white-papers-research/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf
49. Proof of burn. https://en.bitcoin.it/wiki/Proof_of_burn
50. K. Qin, A. Gervais, An overview of blockchain scalability, interoperability and sustainability. Hochschule Luzern Imperial College London Liquidity Network (2018)
51. A. Riard, G. Naumenko, Time-dilation attacks on the lightning network. arXiv preprint arXiv:2006.01418 (2020)
52. E. Rohrer, J. Malliaris, F. Tschorsch, Discharged payment channels: quantifying the lightning network's resilience to topology-based attacks, in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (IEEE, New York, 2019), pp. 347–356
53. M. Romiti, F. Victor, P. Moreno-Sanchez, B. Haslhofer, M. Maffei, Cross-layer deanonymization methods in the lightning protocol (2020). arXiv preprint arXiv:2007.00764
54. S. Roos, P. Moreno-Sanchez, A. Kate, I. Goldberg, Settling payments fast and private: efficient decentralized routing for path-based transactions, in *Network and Distributed System Security Symposium* (2018)
55. Sidechains. https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/sidechains/
56. A. Singh, R.M. Parizi, M. Han, A. Dehghantanha, H. Karimipour, K.K.R. Choo, Public blockchains scalability: an examination of sharding and segregated witness, in *Blockchain Cybersecurity, Trust and Privacy* (Springer, Berlin, 2020), pp. 203–232
57. V. Sivaraman, S.B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, M. Alizadeh, High throughput cryptocurrency routing in payment channel networks, in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)* (2020), pp. 777–796
58. The life and death of plasma. https://medium.com/dragonfly-research/the-life-and-death-of-plasma-b72c6a59c5ad
59. The scalability trilemma in blockchain (2018). https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df
60. S. Tikhomirov, R. Pickhardt, A. Biryukov, M. Nowostawski, Probing channel balances in the lightning network (2020). arXiv preprint arXiv:2004.00333
61. Understanding: Plasma chains vs. sidechains (2019). https://docs.plasma.group/en/latest/src/plasma/sidechains.html
62. A. van Wirdum, Understanding the lightning network, part 2: creating the network (2016). https://bitcoinmagazine.com/articles/understanding-the-lightning-network-part-creating-the-network-1465326903
63. A. van Wirdum, Understanding the lightning network, part 1: Building a bidirectional bitcoin payment channel (2016). https://bitcoinmagazine.com/articles/understanding-the-lightning-network-part-building-a-bidirectional-payment-channel-1464710791
64. R. Vlastelica, Why bitcoin wonť displace visa or mastercard soon (2017). https://www.marketwatch.com/story/why-bitcoin-wont-displace-visa-or-mastercard-soon-2017-12-15

65. P. Wang, H. Xu, X. Jin, T. Wang, Flash: efficient dynamic routing for offchain networks, in *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies* (2019), pp. 370–381
66. What are channel factories and how do they work? (2018). https://bitcoin.stackexchange.com/questions/67158/what-are-channel-factories-and-how-do-they-work
67. L. Yang, V. Bagaria, G. Wang, M. Alizadeh, D. Tse, G. Fanti, P. Viswanath, Prism: Scaling bitcoin by 10,000 x (2019). arXiv preprint arXiv:1909.11261

Illicit Blockchain Content: Its Different Shapes, Consequences, and Remedies

Roman Matzutt, Martin Henze, Dirk Müllmann, and Klaus Wehrle

1 Introduction

Bitcoin introduced the blockchain as a digital ledger for financial transactions of its digital currency. The blockchain allows Bitcoin nodes to reach consensus about what transactions took place in the past and thereby prevents the double spending of coins by irrevocably recording all accepted transactions. The immutable and distributed ledger that is the blockchain proved highly beneficial for this application, and the technology is now being explored for various applications other than digital currencies, most notably digital notary services [6] and conflict resolution [69].

In addition to the trustworthy digital payments offered by cryptocurrencies, these applications store *nonfinancial* on the blockchain, i.e., they make use of smart contracts or augment Bitcoin transactions with arbitrary data. However, allowing any user to irrevocably store arbitrary data on the blockchain and distribute that data across the Bitcoin network also has notable downsides, especially if objectionable or even illegal content is added [32, 40, 42, 47]. Consequently, nonfinancial blockchain content has both a great potential and a great inherent risk of misuse. Blockchain users, developers, and researchers should consider such negative consequences when using, developing, or improving blockchain systems.

R. Matzutt (✉) · K. Wehrle
RWTH Aachen University, Aachen, Germany
e-mail: matzutt@comsys.rwth-aachen.de; wehrle@comsys.rwth-aachen.de

M. Henze
RWTH Aachen University, Aachen, Germany

Fraunhofer FKIE, Wachtberg, Germany
e-mail: henze@cs.rwth-aachen.de

D. Müllmann
Goethe-Universität Frankfurt am Main, Frankfurt/Main, Germany
e-mail: muellmann@jur.uni-frankfurt.de

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_10

This chapter aims to provide a holistic overview of the different shapes of nonfinancial blockchain content, the negative consequences objectionable or even illegal blockchain content can have, and to encourage researching further remedies. We first outline how blockchain content gets distributed across the blockchain's underlying peer-to-peer network and subsequently assess the technical and legal implications thereof. There, we focus on German law to show that illegal content could cause the possession of a full blockchain copy to become illegal in the future in one of the most prominently represented jurisdictions within the Bitcoin network [72]. We further provide a deep dive into the low-level methods for content insertion in Bitcoin and outline the protocols used by services and applications that build upon these methods, e.g., for vastly simplified file uploads to the blockchain.

Based on this foundation of blockchain content insertion, we discuss different lines of research for countermeasures aiming to mitigate negative impacts of blockchain content, i.e., (a) preventing unwanted content from ever entering the blockchain, (b) locally or globally pruning data from the blockchain without changing the global consensus of what is part of the blockchain, and (c) retrospectively redacting data from the blockchain. Based on this overview, we identify future research directions to strengthen the good uses of nonfinancial blockchain content while simultaneously mitigating the negative consequences of objectionable blockchain content.

Organization In Sect. 2, we reiterate the network distribution of blockchain content and its potential technical and legal consequences. Section 3 then illustrates the basic methods for content insertion and how content insertion services use these building blocks. We then provide an overview of the status quo of content insertion in Bitcoin in Sect. 4. Afterward, we present proposed mitigation strategies: We discuss pre-insertion mitigation in Sect. 5, pruning both objectionable content and also obsolete data in Sect. 6, and redactable blockchains in Sect. 7. We outline future research directions in Sect. 8 and Sect. 9 concludes this chapter.

2 Distribution of Blockchain Content and Its Consequences

We start off by reiterating the path a transaction takes from its proposal to the network to its acceptance into the blockchain. Based on this illustration, we highlight technical consequences of manipulating financial transactions to hold content as well as potential legal consequences if that content is privacy-sensitive or even illegal.

2.1 The Path of Blockchain Content and Technical Consequences

Blockchain systems establish consensus by distributing all valid data within the network so that each node (a) is aware of all information required to execute the

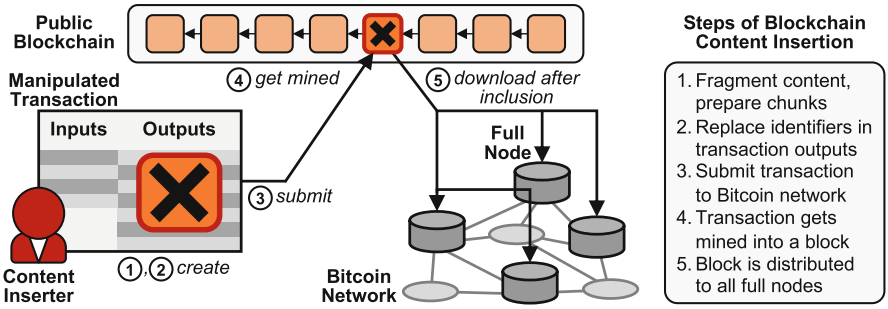


Fig. 1 Manipulated transactions get included on the blockchain and are hence distributed to the full nodes of the Bitcoin network. On the right, we detail the steps for inserting content

next state transition and (b) can independently validate which data to accept. In this section, we show how objectionable content gets distributed accordingly. We further highlight the technical consequences of both this content distribution, and naïve strategies nodes can apply locally to avoid storing such content.

Figure 1 gives a high-level overview of the flow of information within the Bitcoin network and how objectionable content gets spread accordingly. The figure further details the basic steps of blockchain content insertion, from augmenting a transaction with content through manipulation to the content's distribution to the Bitcoin nodes. Any Bitcoin user can alter their transactions to contain arbitrary content. Most commonly, users will add the content by splitting it into small chunks and inserting them into the outputs of large transactions, e.g., by manipulating standard Bitcoin transactions. As we elaborate in Sect. 3.1, the user can, for instance, replace the mutable fields of standard Bitcoin transactions with the data chunks. In this case, extracting the data again becomes easy and only requires concatenating the mutable values again. The user then submits the transaction for insertion in a future block. While the transaction is effectively distributed across the network at this stage, we focus on the case that a miner irrevocably adds the content to the blockchain in the remainder of this chapter. As Bitcoin was not designed with transaction manipulability in mind, the full nodes responsible for validating each transaction's correctness do not reject content-holding transactions or blocks. In fact, full nodes have no means to reliably identify manipulated transactions as we further discuss in Sect. 5.1. Hence, the objectionable content can enter the blockchain and is irrevocably stored once subsequent blocks have been mined on top of the content-holding block.

At this point, full nodes indefinitely store a verbatim copy of this block and the content within. Theoretically, they can choose to prune this raw block data after processing it completely. However, in this case, pruning nodes lose the ability to reconstruct their local block and transaction indexes if they experience technical issues. Instead, they rather have to download the full blockchain again. Moreover, newly joining nodes have to obtain and process a full copy of the blockchain as part of their bootstrapping process [43]. Thus, Bitcoin requires that a sufficient number

of full nodes maintain a full blockchain copy and refrain from local pruning for the system to remain operable.

The inclusion of nonfinancial blockchain content also has multiple technical consequences. Beyond forcing full nodes to store unwanted data indefinitely, the practice of manipulating standard transactions unnecessarily bloats the set of unspent transaction outputs (UTXO set), which records all transaction outputs that are not provably unspendable, including manipulated outputs. Full nodes use the UTXO set as a lookup table to improve the performance of validating pending transactions [46]. Hence, bloating the UTXO set has negative consequences for all full nodes. Finally, this form of manipulating transaction outputs costs the content inserter money, which will become inaccessible and thus *burned* due to the manipulation (cf. Sect. 3.1). In the following, we further consider potential legal consequences node operators may face because of nonfinancial blockchain content.

2.2 Considerations of Potential Legal Consequences

As we discussed in the previous section, blockchain content gets distributed to all full nodes, which constitute the network's backbone. We now consider potential legal consequences resulting from this distribution pattern regarding both *illegal blockchain content* and potential *privacy violations*.

Illegal Content We first consider the extreme case of child abuse imagery, which was already stored on Bitcoin SV's blockchain [9, 48], to discuss potential legal consequences for node operators. Courts need to rule whether such blockchain content constitutes an *accessible document* and whether a user *knowingly possesses* the content to determine whether that user becomes prosecutable. For instance, the USA [53], England [59], and Ireland [33] deem all data illegal that can be converted into a visual representation of illegal content. Here, the term *conversion* only covers creating a visual presentation, e.g., decoding an image file for displaying it. However, we expect that the interpretation of "convertibility" could be extended to also recognize blockchain content, given that its data chunks can be easily reassembled from a Bitcoin transaction (cf. Sect. 3.1).

This easy convertibility is already acknowledged, e.g., under German law. In Germany, criminal prosecution would require a statutory offense which, in our case, penalizes the users' mere possession of digital content. The German Criminal Code (StGB) sanctions the possession and the undertaking to obtain child and youth pornographic documents via §§ 184 b sec. 3 and 184 c sec. 3. Crucially, German law considers even storage devices, e.g., hard disks, to constitute "documents" (§ 11 sec. 3 StGB). Hence, possessing the storage device holding the blockchain suffices to possess its contents. Notably, these strict rulings even extend to illegal content that is only present in cache memory or RAM [17]. As a consequence, every user who maintains a local blockchain copy holding illegal content possesses prosecutable content within the meaning of the law; this still holds even though the content is typically fragmented due to its easy reassembly.

Criminal prosecution nevertheless requires that an alleged offender is also *knowingly* maintaining control over illegal content. In our scenario, where an unknown user stores illegal content on the blockchain, honest users verify incoming transactions obliviously and unaware of any illegal content. However, when such content is brought to a user's attention (e.g., as previously attempted by Interpol [32] or via media coverage [8, 9]) and the user does not immediately remove that content, the law assumes that the user *intends* to keep control over the content [66]. If full nodes were viewed as service providers, their blockchain copy could fall under punishment liberty [30, 35] as service providers have certain privileges when storing external information (§ 10 sec. 1 TMG[1]). However, punishment liberty again only applies as long as the provider has no knowledge of the illegal data or makes it inaccessible immediately, i.e., it may not apply anymore because of corresponding public reports [60, 66]. In consequence, any user who is aware of illegal content is required to delete it and thus cannot keep a full blockchain copy anymore.

Privacy Considerations In addition to illegal content, also legal content that conflicts with the European General Data Protection Regulation (GDPR) can become problematic, e.g., when considering the right to delete or correct data and the right to be forgotten (Art. 16, 17 sec. 1, 2 GDPR). The GDPR here is concerned with the processing (Art. 4 No. 2 GDPR) of personal (Art. 4 No. 1 GDPR) data. Even though public blockchains operate pseudonymously, persons can remain identifiable, which then allows linking some publicly accessible blockchain information back to them [39]. At this point, we have to ask who is the *controller* when it comes to processing blockchain data within the meaning of Art. 4 No. 7 GDPR, as the controller has to ensure compliance with the privacy rights provided by the GDPR. However, who assumes this role in our scenario is currently subject to controversy and could include, for example, the developers, miners, or network nodes [39]. While the developers define the processing rules for validating transactions, the miners and full nodes are responsible for the actual data processing. Further controversies debate whether all network nodes can be considered *joint* controllers (Art. 26 sec. 1 GDPR) [10, 39, 61]. In any case, the controller needs to ensure that the blockchain remains changeable to comply with the GDPR's defined privacy rights. Only few blockchain systems offer this property today, as we will further discuss in Sect. 7.

Summary Despite the lack of court rulings, we anticipate that the outlined scenario and the current legal perspective in jurisdictions such as the USA or England are more worrisome than reassuring regarding the prosecution of blockchain users. In conclusion, we believe that blockchain developers must keep the potential that illegal content can jeopardize the whole system in mind.

[1] Concerning the applicability of TMG and Blockchain as a Telemedium cf. Peters [57].

3 Content Insertion Methods

In this section, we illustrate the intended and unintended ways for Bitcoin users to augment their transactions with additional content. First, we discuss where content can be stored within a Bitcoin transaction or block. After presenting these methods, we present higher-level protocols and services used to further facilitate content insertion.

3.1 Low-Level Insertion of Nonfinancial Blockchain Data

We first review the technical options users have to augment Bitcoin transactions with additional content. Table 1 provides an overview of the available methods.

Identifier Manipulation Bitcoin transactions consist of inputs, which unlock funds from previous transactions, and outputs, which specify new spending conditions. Both inputs and outputs make use of Bitcoin's scripting language [14] to express and satisfy the spending conditions, respectively. While this scripting language, in theory, allows for flexible spending conditions, only a few standard script patterns based on *ownership identifiers* are used for transferring bitcoins [54]. The *pay-to-public-key (P2PK)* and *pay-to-public-key-hash (P2PKH)* patterns specify a public key or hash value thereof, and spending these outputs requires knowledge of the corresponding private key. Similarly, the *m-out-of-n-pay-to-multisig (m-n-P2MS)* pattern specifies n public keys and m corresponding private keys are required for spending the associated coins, i.e., the key holders have to agree on spending the funds. Finally, the *pay-to-script-hash (P2SH)* pattern commits to a spending condition by only publishing the hash value of the script specifying the full condition. Spending the corresponding funds then requires disclosing the script matching the hash value alongside the other values required for unlocking funds.

Table 1 Overview of content insertion methods. The last column exemplifies known applications or insertion services. Bitcoin SV has an increased `OP_RETURN` capacity

Insertion Method	Capacity	Intended	Burns Coins	Constraints	Applications / Services
Identifier Manipulation	●	no	yes	—	Satoshi Uploader, CryptoGraffiti, Apertus (discussed in Sect. 3.2)
P2SH Stuffing	●	no	no	Previous TX & Colluding miner	P2SH Injectors (Sect. 3.2)
Non-St. Outputs	●	no	depends	Colluding miner	—
Non-St. Inputs	●	no	no	Previous TX	—
Value Encoding	◐	no	no	—	Bitcoin Message [65]
OP_RETURN	○ / ●*	yes	no	—	Various [8], Bitcoin Data Protocol (Sect. 3.2)
Nonce Field	○	no	no	Miner only	—
Coinbase	○	yes	no	Miner only	Miner advertisements [9], voting on BIPs [9], CoinPrune (Sect. 6.2)

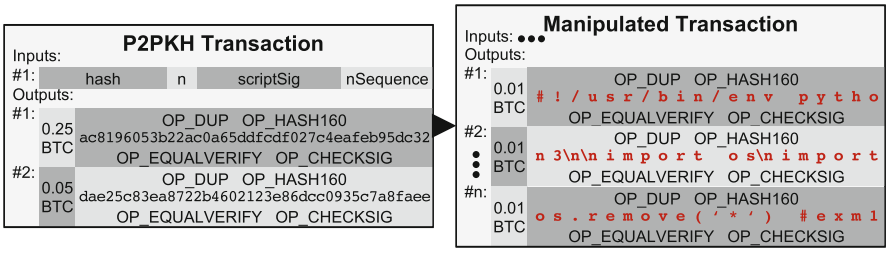


Fig. 2 The mutable identifiers defining spendability conditions of transaction outputs can be replaced with chunks of arbitrary data

While purely intended for financial transactions, all standard script patterns can be misused to store arbitrary blockchain content instead. Figure 2 illustrates this *manipulability* using P2PKH as an example; the method, however, generalizes to other patterns with varying per-byte costs [42]. Most Bitcoin transactions have two P2PKH outputs [41], which have the pattern `OP_DUP OP_HASH160 [public key hash] OP_EQUALVERIFY OP_CHECKSIG` [54]. The user can then spend coins secured by this kind of transaction output by providing a `[signature][public key]` pair. Hence, the P2PKH script pattern first matches the submitted public key against the required hash value before verifying the supplied signature, proving that the user possesses the corresponding private key and can spend the coins.

However, even though Bitcoin nodes execute these checks for each attempt to spend the coins of a P2PKH output, the nodes cannot verify that the hash value within the output script indeed corresponds to a known key pair. As illustrated in Fig. 2, a user can thus replace the 20 B-long hash value with arbitrary data. Furthermore, the user can use additional P2PKH outputs to encode more data chunks within the same transaction. By packing a single transaction with as many manipulated outputs as possible, the user can maximize the transaction's capacity for holding content. This insertion method is simple but has the drawback of *burning* bitcoins, i.e., making them inaccessible in the future: Each transaction output (except for `OP_RETURN`; see below) needs to spend a minimum amount of bitcoins, called *dust* [54]. Since the user arbitrarily manipulated the output's identifier, it is highly unlikely that a matching private key exists, and the associated bitcoins are likely lost forever [42].

Notably, this content insertion method satisfies the easy extractability of the content we discussed in Sect. 2.2. Namely, the transaction creator can entirely determine the ordering of all outputs, and concatenating all mutable fields of the transaction's output scripts is sufficient to obtain the content.

P2SH Stuffing Pay-to-script-hash (P2SH) transaction outputs only contain a commitment to a *redeem script*, which specifies the actual spending conditions of transferred funds. The redeem script should also follow a standard pattern as described above [3] and require that only corresponding input values are present. However, while full nodes discard pending transactions that violate these rules,

they accept the transaction if a miner included them in a block. Bypassing these restrictions allows for more sophisticated content insertion methods involving the redeem script. Sward et al. [65] investigated how to utilize P2SH input scripts for content insertion. A key factor here is that individual stack elements, i.e., data that is being pushed onto the stack when evaluating a script, are restricted to a size of 520 B [65]. The redeem script itself is a stack element as well and therefore also affected by these restrictions. However, content can be encoded as additional input values and then processed by the redeem script, e.g., by dropping the inputs [65], which allows for utilizing P2SH input scripts more efficiently. Using this approach further has the advantage of not burning any coins, as identifier manipulation would.

Nonstandard Transaction Scripts Like nonstandard P2SH redeem scripts, Bitcoin nodes should not relay transactions whose outputs diverge from standard patterns. However, since the same checks are in place here as well, such transactions can still occasionally enter the blockchain. This form of content insertion promises the highest capacity since the inserting user does not have to cater to any other rule, e.g., the script could only push and drop new content chunks. However, the high likelihood that the transaction will never be mined into a block makes the method cumbersome to use. Input scripts, contrarily, are not prone to the same restriction, e.g., a similar pattern to P2SH stuffing could be applied.

Encoding via Values Transactions can further be augmented with arbitrary content by encoding it using the values transferred via multiple outputs. Sleiman et al. [64] proposed using arithmetic coding to encode single letters per output, which would yield a maximum capacity of around 3 KiB for a standard transaction. Furthermore, the user needs more coins than the required dust amounts to encode the different letters in their outputs. Hence, the value fields can be used to insert additional blockchain content, and different methods than the one proposed by Sleiman et al. [64] are conceivable. Yet, this way of inserting blockchain data is more restricted than, for example, identifier manipulation, even though value encoding does not burn coins.

OP_RETURN Since 2014 [15], Bitcoin has provided a special script type for explicitly augmenting regular transactions with small chunks of arbitrary data. Starting the script of a transaction output with the `OP_RETURN` operation causes the output to be provably unspendable [15], i.e., adding data after the `OP_RETURN` operation will not bloat the UTXO set (cf. Sect. 2.1). However, since content insertion is still discouraged and `OP_RETURN` was meant as a form of compromise [15], additional rules are in place to prevent mass content insertion. Namely, a transaction may only have one `OP_RETURN` output, and the payload of this output is restricted to 83 B (formerly 80 B) [6]. The fork Bitcoin SV, for instance, allows for much larger payloads of `OP_RETURN` transaction outputs (up to 100 kB [49]), which was previously used for inserting child abuse imagery into that blockchain [9, 48].

Miner Exclusive Insertion Miners have the most control over shaping the blockchain, but they need to follow the consensus rules to prevent their blocks from being rejected by the network's full nodes. Hence, block headers are comparably

resilient to content insertion. The version field is fixed by the consensus rules, and the values of both the timestamp and target fields must lie within reasonable ranges in order to be accepted by full nodes. Contrarily, the miner is expected to randomly fill the nonce field for its proof of work (PoW). In theory, the miner could thus fix this field and solve the PoW by varying other aspects of its block, e.g., altering the block's transactions, to find a valid block nevertheless. However, this approach is tedious and yields only 4 bytes of space for arbitrary content, and a higher-capacity alternative exists: The first transaction of each block is a special *coinbase* transaction, which mints new bitcoins that the miner is free to transfer to itself. The special structure of coinbase transactions allows for inserting arbitrary data as well. The input script of the coinbase transaction contains a special coinbase field with a length of up to 100 B. The field is expected to contain the block's position within the blockchain encoded as a 4 B-integer [2]. However, the gap created by specifying a larger field length can be filled with arbitrary data. In fact, Bitcoin's genesis block already uses this method [63], and thus we consider using the coinbase field as an intended means of blockchain content insertion. In the past, the coinbase field has been used, e.g., to advertise mining pools [7] or to vote for improvements to Bitcoin's consensus rules [3, 7]. We further discuss in Sect. 6.2 how nodes can use the coinbase field to coordinate extended features such as globally pruning obsolete blockchain data.

We have illustrated the variety of available methods to insert nonfinancial data into Bitcoin's blockchain in this section. The most important among those methods are (a) identifier manipulation, as this approach allows for inserting larger content; (b) `OP_RETURN` transactions, which provide an intended method for augmenting transactions with small chunks of data; and (c) storing a small data chunk in the coinbase field, which facilitates extended coordination processes among miners. We now give an introduction to content insertion services, which build upon these methods to further facilitate blockchain content insertion.

3.2 Content Insertion Services

In this section, we present auxiliary tools and online services that facilitate the insertion of blockchain content. In particular, we discuss the technical details of how these services utilize the building blocks (cf. Sect. 3.1) and their unique features.

Satoshi Uploader This content insertion service consists of two Python scripts that were published on the Bitcoin blockchain, one for encoding a file as a Bitcoin transaction and one for extracting it later on [63]. We refer to these scripts as the *Satoshi Uploader*, as their copyright label reads Satoshi Nakamoto, although this claim is questionable [63]. The Satoshi Uploader uses 1-3-P2MS transaction outputs and uncompressed public keys, which have a length of 65 B per key, to embed the input file and its CRC32 checksum into a Bitcoin transaction. The script pads the

last chunk with zero-bytes to fill up the last public key and, if possible, reduces the number of public keys in the last P2MS output.

CryptoGraffiti *CryptoGraffiti* [25] is a web service launched in 2014, which originally was intended for parsing blockchain content and subsequently started to also offer a form to write text and files to the blockchain [20]. Initially launched on the Bitcoin (Core) blockchain, the service shifted to Bitcoin Cash in October 2017 and, as of October 2020, only operates on Bitcoin SV. Users of the service are instructed to send the funds required to engrave the content as well as a 10 % service fee to a Bitcoin address, which is then used to create the transaction [25]. Similar to the Satoshi Uploader, CryptoGraffiti relies on identifier manipulation, but the service uses P2PKH scripts instead of P2MS scripts. Each output spends the dust amount of 546 sat/B, and the profits are sent to a fixed Bitcoin address, which facilitates the detectability of CryptoGraffiti transactions. The service allows uploading files of up to 50 KiB together with optional comments.

P2SH Injectors *P2SH injectors* describe a class of services that insert content using the P2SH stuffing method we described in Sect. 3.1. As with the Satoshi Uploader, the origin of this service can be traced back to a Python script [38], but slight variations of corresponding transactions can be found on Bitcoin's blockchain.

Apertus The service Apertus [31] provides a sophisticated protocol for encoding large files across multiple transactions. Pending transactions are usually independent of each other, and a user submitting multiple transactions has no direct control over which blocks the transactions will be added to and in which order. To circumvent this limitation, Apertus uses a second layer of chunking. The file to be uploaded is stored in P2PKH outputs of multiple transactions, and a final *archive transaction* provides a lookup table of the transaction identifiers required for reassembling the original file. Notably, archive transactions can be stored hierarchically, i.e., one archive transaction can point to multiple fragments of another, larger archive. Apertus transactions can contain additional meta information, such as a digital signature, a textual comment, and an optional filename.

Bitcoin Data Protocol The *Bitcoin Data Protocol (BDP)* [67] specifies a protocol for `OP_RETURN`-based data insertion. This protocol is heavily utilized on Bitcoin SV (cf. Sect. 4.3), where an `OP_RETURN` output can hold up to 100 kB [49]. BDP uses a fixed protocol identifier and stores binary data and meta information as individual push operations in the `OP_RETURN` output script: `[ID][payload][mime type][encoding (optional)][filename (optional)]...` [67]. Other services, such as Money Button (https://www.moneybutton.com) or the blockchain-backed social network Twetch (https://twetch.app), rely on BDP and encapsulate a BDP message in their `OP_RETURN` outputs. Money Button was used to store child abuse imagery on Bitcoin SV's blockchain in the same month the full capacity of Bitcoin SV's `OP_RETURN` outputs could be used [9, 48].

In conclusion, content insertion services are as heterogeneous as the building blocks they rely on. The spectrum ranges from simple scripts for merely embedding

content into a transaction, over easy-to-use web services, to sophisticated protocols to overcome space limitations imposed by consensus rules. In the following, we investigate how these different content insertion methods have been used over time.

4 Quantitative and Qualitative Analysis

We now present to which extent the respective content insertion methods and services we presented in Sect. 3 are used in the wild. To this end, we monitored the blockchain of the original Bitcoin network, i.e., Bitcoin Core, as well as its direct active forks Bitcoin Cash and Bitcoin SV. After outlining our methodology, we first analyze the overall utilization of low-level insertion methods across blockchains and then focus on the number and types of files inserted via content insertion services.

4.1 Methodology

For our measurements, we extend the measurement framework we implemented for our previous study [42] and rely on the same methodology if not stated otherwise. With this extension, we can now analyze transactions from the blockchains of Bitcoin Core, Bitcoin Cash (forked off Bitcoin Core at block height 478,558), and Bitcoin SV (forked off Bitcoin Cash at block height 556,766) up to a block height of 640,000. While Bitcoin Core reached this block height on July 20, 2020, Bitcoin Cash reached it on June 17, 2020, and Bitcoin SV reached it on June 19, 2020, respectively. In all measurements, we only consider blocks exclusive to the respective blockchain. For instance, we ignore the shared prefix of blocks between Bitcoin Core and Bitcoin Cash when discussing Bitcoin Cash. Furthermore, we aggregate all data month-wise. We apply automated checks to identify and classify transactions based on the distinct patterns used by the content insertion services we identified in Sect. 3.2. Specifically, the MIME type field used in the Bitcoin Data Protocol (BDP) and assumptions about which data push operation of a BDP `OP_RETURN` output should contain the payload data allow us to investigate the (claimed) type of data inserted via this protocol without manual investigation.

4.2 Low-Level Content Insertion

In this section, we provide an overview of the extent of content insertion in general by first discussing its evolution on the blockchain of Bitcoin Core since our first study [42] and by then identifying similarities and differences between content insertion in Bitcoin Core, Bitcoin Cash, and Bitcoin SV, respectively.

We first compare the overall utilization of content insertion methods across the three considered blockchains. Figure 3 shows how many transactions using the

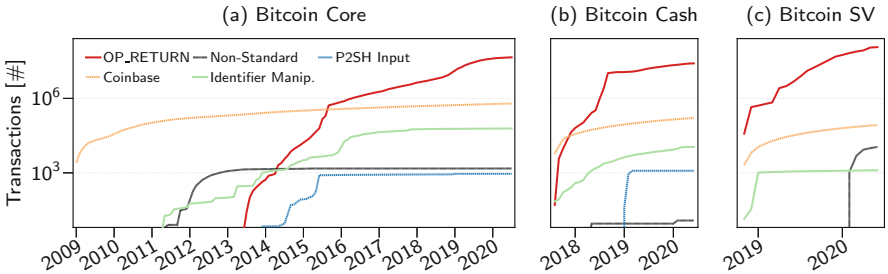


Fig. 3 Number of content-carrying transactions exclusive to the blockchains of (**a**) Bitcoin Core, (**b**) Bitcoin Cash, and (**c**) Bitcoin SV based on our content detectors

different insertion methods we discussed in Sect. 3.1 were cumulatively included on the blockchains of Bitcoin Core (Fig. 3a), Bitcoin Cash (Fig. 3b), and Bitcoin SV (Fig. 3c), respectively. This analysis continues our previous study [42], where we only considered the blockchain of Bitcoin Core up to a block height of 482,870 as of August 31, 2017. Compared to that study, the number of transactions with manipulated transaction outputs on Bitcoin Core's blockchain increased by 24.5 % from 49,342 to 61,433 instances. Contrarily, we observed only 62 additional transactions that insert content through P2SH input scripts (using P2SH injectors) and only three additional nonstandard transactions on this blockchain since our last measurement [42]. The blockchains of Bitcoin Cash and Bitcoin SV have a similar yet accelerated general development. However, we make four additional observations: First, identifier manipulation is actively being used on all considered blockchains. We observe 11,009 instances of identifier manipulation exclusive to Bitcoin Cash, i.e., this method was used on both Bitcoin Cash and Bitcoin Core to a similar extent in parallel. Contrarily, we only observed 1269 instances of identifier manipulation on the blockchain of Bitcoin SV since its inception, and utilization steeply declined since only 242 instances occurred after January 2019. Second, both Bitcoin Cash and Bitcoin SV make increased utilization of `OP_RETURN` outputs compared to Bitcoin Core. Notably, Bitcoin SV has already accumulated 2.57× the number of transactions making use of an `OP_RETURN` output compared to Bitcoin Core despite its short lifetime. Figure 4 further highlights this observation by showing the share of all transactions in each considered blockchain that have an `OP_RETURN` output. While the share of Bitcoin Core transactions holding an `OP_RETURN` output increased to 8.2 % of all transactions, both Bitcoin Cash (55.5 %) and Bitcoin SV (68.3 %) experience a vastly higher `OP_RETURN` utilization. Third, Bitcoin SV accumulated a comparably large amount of nonstandard transactions (11,059) in only one and a half years. For comparison, we observed 1500 nonstandard transactions in Bitcoin Core and only twelve nonstandard transactions in Bitcoin Cash. Fourth, we did not observe any content insertion via P2SH input scripts for Bitcoin SV.

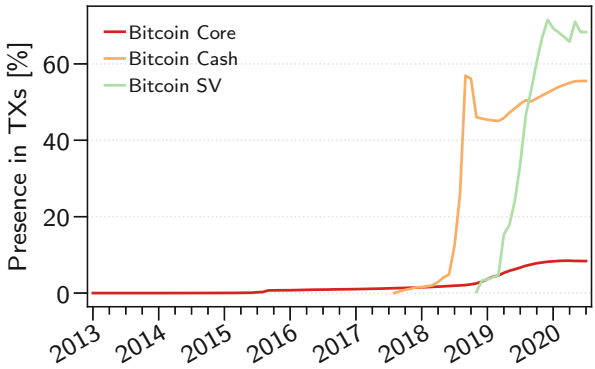


Fig. 4 The utilization of `OP_RETURN` outputs differs across Bitcoin variants

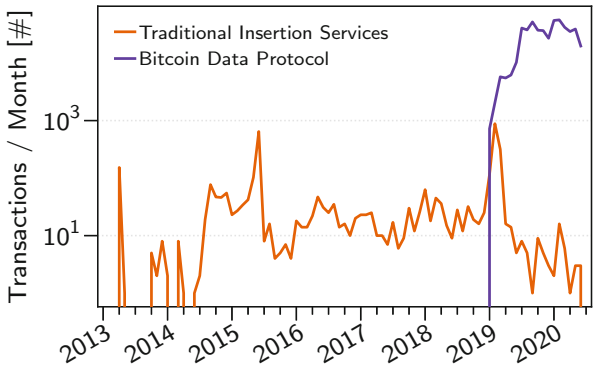


Fig. 5 BDP is now used frequently compared to traditional insertion services

In summary, content is actively being inserted via both intended (`OP_RETURN`) and unintended means (identifier manipulation) on the considered blockchains. Notably, nonstandard transactions are still being mined into blocks as of 2020.

4.3 Service-Based Content Insertion

In this section, we assess to which extent content insertion services are used to insert nonfinancial data into the blockchains of Bitcoin Core, Bitcoin Cash, and Bitcoin SV. We first give an overview of the general utilization of content insertion services across the blockchains. Then, we reconsider those content insertion services we identified in our previous study [42], i.e., the Satoshi Uploader, CryptoGraffiti, Apertus, and P2SH Injectors (cf. Sect. 3.2), which we collectively refer to as *traditional* content insertion services in the following. Finally, we focus on the utilization of the Bitcoin Data Protocol (BDP) on the blockchain of Bitcoin SV.

Overall Utilization Figure 5 provides an overview of the utilization of traditional content insertion services and BDP across the considered blockchains. We observe

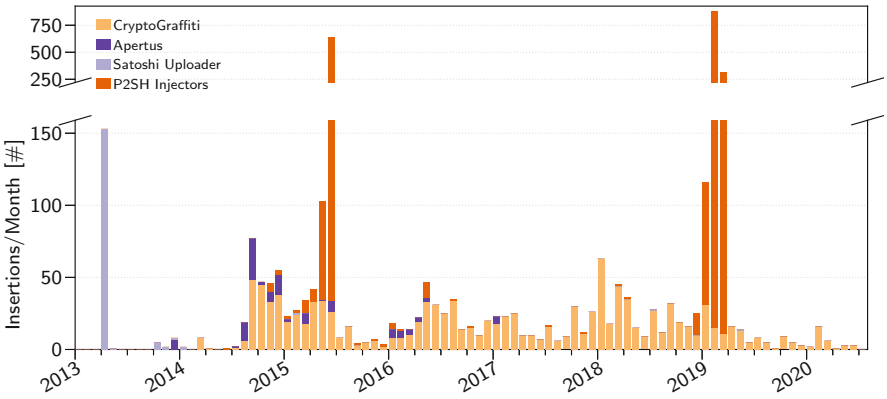


Fig. 6 This per-month overview shows the ongoing but declining (in favor of BDP) utilization of traditional insertion services

that service-backed content insertion occurs continually starting from June 2014, with the earliest utilization in April 2013. However, even though this kind of content insertion is present each month, traditional content insertion services were only rarely used. With five exceptions, all traditional content insertion services combined have only been used at most 80 times a month across all considered blockchains. We observed 25.5 insertions per month across all considered blockchains using traditional content insertion methods and a total of 3540 insertions. In stark contrast to traditional content insertion services, the `OP_RETURN`-based BDP is being heavily used on Bitcoin SV. Since its inception in January 2019, we recognized an average of 25,461.7 BDP insertions per month, accumulating to a grand total of 509,234 instances within the 83,234 blocks we considered.

Traditional Content Insertion Services We further investigate each service's contribution to the overall extent of content insertion, starting with the traditional services. Figure 6 shows the month-wise utilization of the traditional content insertion services accumulated over all considered blockchains. The overall largest contribution here comes from P2SH injectors, which were used to insert data in 2011 instances. However, their utilization is mostly concentrated in shorter bursts. The first burst occurred on Bitcoin Core during May and June 2015 (679 instances). During the second burst, we first observed 59 instances of P2SH injector utilization on Bitcoin Core in December 2018 and January 2019 and subsequently observed 1215 instances between January and March 2019 on Bitcoin Cash. Next, we observed 1245 instances where CryptoGraffiti was used to insert data. In contrast to P2SH injectors, CryptoGraffiti is being used continually since July 2014, with nine initial instances during March and April. In correspondence to the service operator's announcements (cf. Sect. 3.2), we observe a transition of CryptoGraffiti utilization from Bitcoin Core to Bitcoin Cash during October 2017 and another shift from Bitcoin Cash to Bitcoin SV during November and December 2018. Both remaining services, the Satoshi Uploader and Apertus, are less frequently

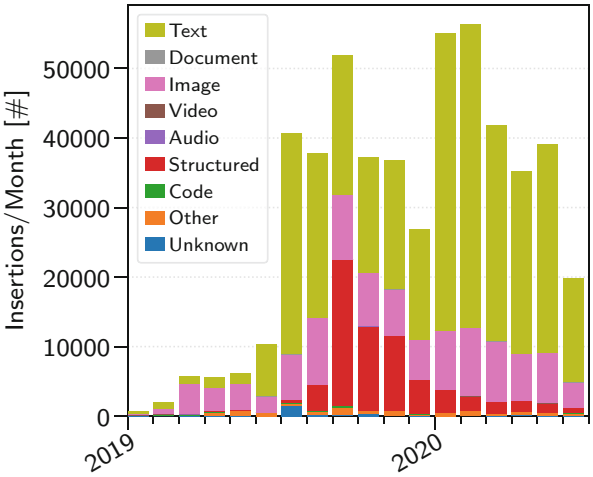


Fig. 7 Absolute numbers of BDP insertions per month and payload type

used with 167 and 117 observed instances, respectively. However, we only counted archive transactions for Apertus, and we provide more details on the occurring data fragmentation using this service in our previous study [42]. Specifically, we did not observe any new utilization of Apertus since that study. Contrarily, we observed two new instances of Satoshi Uploader transactions in July 2018 on Bitcoin Cash and in May 2019 on Bitcoin Core, respectively. In conclusion, the overall utilization of traditional content insertion decreased noticeably after March 2019, but at least CryptoGraffiti is actively being used to insert new content via identifier manipulation.

Bitcoin Data Protocol As Fig. 5 indicates, the decline in utilizing traditional content insertion roughly correlates to the increasing popularity of the Bitcoin Data Protocol (BDP) on Bitcoin SV. BDP has the advantage for content inserters of allowing them to use 100 kB-large `OP_RETURN` outputs to insert their data and meta information, such as the inserted data's MIME type, in a structured manner. Figures 7 and 8 show the absolute numbers of transactions with BDP payload per month on Bitcoin SV and highlight the relative shares of data types, respectively. We observe that the utilization of BDP is dominated by text-based formats such as plain text and HTML or Markdown documents. Collectively, text-based payloads constitute 64.38 % of all BDP instances. Next, 20.58 % of BDP instances relate to image data, with a total of 104,820 instances we observed within our considered time frame. Third, structured data, such as JSON, XML, or CSV data, makes up 12.62 % of all BDP instances. This data was inserted in a larger bulk between August and December 2019 but continues to constitute 4.25 % of BDP insertions on average. Noticeably, BDP is also being used to insert (alleged) audio (363 instances) and video data (170 instances) as well as code (1044 instances), e.g., Python, JavaScript, and Shell scripts, and other documents (325 instances) such as PDFs or Microsoft Office documents. In summary, BDP lifted blockchain data

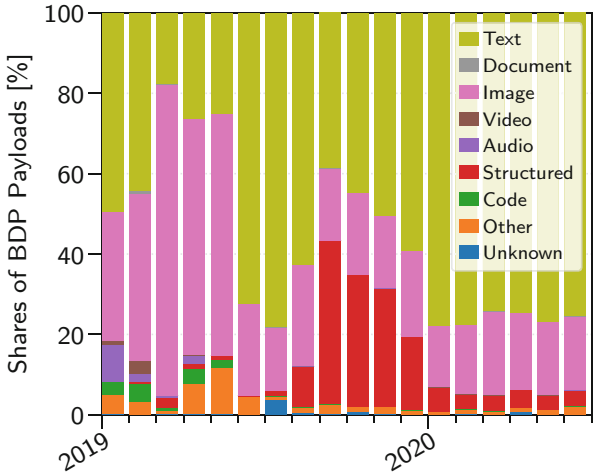


Fig. 8 Relative shares of payload types among BDP insertions per month

insertion to an unprecedented level. While most data stored on the blockchain of Bitcoin SV this way is text or structured data, we observe over 100k alleged images and other media and documents that were inserted via the protocol.

Our analysis highlights that content is actively being inserted into the blockchains of Bitcoin Core, Bitcoin Cash, and Bitcoin SV. The Bitcoin Data Protocol has elevated the phenomenon to a new level in Bitcoin SV, but also identifier manipulation is actively being used in parallel to insert data, e.g., via the CryptoGraffiti service.

5 Mitigation of Unwanted Blockchain Content Insertion

After having illustrated and quantified the methods for blockchain content insertion and the consequences of inserting illicit content, we now shift our focus to mitigation strategies. In this section, we start by exploring the available design space for hindering content-holding transactions from entering the blockchain in the first place. Namely, we discuss *filtering* content-holding transactions out, imposing *mandatory minimum fees*, *commitments* to a transaction's benignity, and alternative *balance-based blockchain models*. In the subsequent sections, we then discuss approaches that provide countermeasures against objectionable content after the fact.

5.1 On the Capability to Filter Objectionable Content

The intuitive countermeasure against blockchain content insertion would be a "content firewall," i.e., nodes actively analyze transactions for content and filter out any content-holding transactions. In the following, we highlight the fundamental

complications of this approach, namely, that *full detection of manipulation is impossible* and that *frequent rule updates are not feasible for public blockchain systems* [41].

Impossible Prevention of Content Insertion Bitcoin allows for easy identifier manipulation as full nodes have to accept transactions in good faith without knowing whether the transferred coins will, or can, be spent in the future. However, content insertion remains possible even if the full nodes had the means to reject any transaction that sends funds to unknown private keys [41]. Intuitively, a content inserter can still brute-force private keys such that the resulting on-chain identifier holds a chunk of content. By only using parts of the identifier for storing content, e.g., the first five bytes, and by again packing transactions with many outputs of this kind, the user can still encode content (albeit at a reduced capacity) [41]. This practice is already used to find the so-called vanity addresses [7], and the resulting transactions remain fully spendable. Hence, content can still be added to entirely valid Bitcoin transactions in a computationally feasible manner, even in the strictest possible scenario.

Challenges for Filtering Content Because of the impossibility of reliably preventing all content insertion, full nodes would have to inspect transactions further to identify encoded content. However, naïvely applying dedicated *content detectors* leads to the following three main problems:

First, content inserters may slightly adapt their methods to evade known content detectors, e.g., by using the last bytes of each identifier to encode content instead of the first ones. As a consequence, each node likely has to apply a large number of content detectors to vet a transaction.

Second, this flexibility is challenging regarding the coordination required between nodes. Full nodes must assume that individual content-holding transactions are mined into a block just like nonstandard transactions (cf. Sect. 4.2). In this case, full nodes must universally agree on a set of *mandatory* content detectors to prevent blockchain forks. Hence, frequent variations of insertion methods require equally frequent updates of full nodes, which can result in uncontrolled forking [41].

Finally, false positives can negatively impact the filtering process and render benign transactions invalid [41]. False positives are unlikely to result from larger transactions and could easily be resolved, e.g., by randomly reordering the outputs. However, individual Bitcoin addresses may become unusable, e.g., because their corresponding public key hash consists of printable ASCII characters and is mistaken for a text-holding transaction output [41]. Nodes can mitigate the influence of false positives by applying content detectors only to transactions with more than, for example, five outputs [41]. If the content only requires fewer outputs, the content inserter could use a single `OP_RETURN` output instead, and he would face lower costs with that approach.

In conclusion, a firewall against objectionable content would be desirable to have. However, the additional required network coordination and the high flexibility for content inserters prevent this firewall from being easily deployable.

5.2 Financially Disincentivizing Content Insertion

A different approach to detecting and discarding content-holding transactions is to deter users from inserting content. In the following, we explore the possibility to *financially disincentivize* the insertion of nonfinancial blockchain content.

The main idea here is to extend the transaction fees users pay to miners as an incentive to favor their transactions to *mandatory minimum fees* [41]. Instead of letting miners prioritize transactions paying higher fees, all full nodes would then reject transactions that underpay the mandatory fees. This proposal is motivated by the observation that, typically, many transaction outputs are required to encode objectionable content. For instance, inserting 1 kB of data using P2PKH identifier manipulation requires 50 outputs [41], while the vast majority of Bitcoin transactions have fewer outputs. Many transactions even have only up to five outputs [41]. Hence, the goal is to penalize creating very large and potentially content-holding transactions while not increasing the fees required for the vast majority of benign transactions.

Currently, the Bitcoin Core client monitors the recent per-byte fees paid by other transactions and the delay after which they were added to the blockchain to estimate the fees required for its own transactions to be included in a timely manner. Instead of this probabilistic and locally obtained per-byte fee [24], mandatory minimum fees rely on a fixed per-byte component and additionally have an increasingly penalizing per-output component [41]. The per-output component could be chosen to grow disproportionally for transactions with more outputs since most transactions only have few ($\leq$5) outputs [41]. Hence, penalizing extraordinarily large transactions suitable to contain considerable amounts of content (e.g., $\geq$50 outputs for 1 kB of data) can create an artificial barrier to inserting such content. Treating larger transactions in this special way further helps to mitigate any negative financial impact on the vast majority of benign (typically small) transactions [41]. However, a refinement of this approach that takes the fluctuation of fees into account while ensuring network-wide consensus about the mandatory minimum for transaction fees would be required for practical deployability and currently remains an open research question.

5.3 Cryptographic Commitments to Transaction Benignity

In Sect. 3.1, we have established that blockchain content insertion is currently as easy as replacing mutable identifiers within transactions with chunks of the content. We now summarize an approach [41] to prevent this simple manipulability by updating these identifiers such that alterations become detectable by any node. Hence, content inserters lose control over what data is persisted on and derivable from the blockchain beyond brute-forcing the updated identifiers (cf. Sect. 5.1).

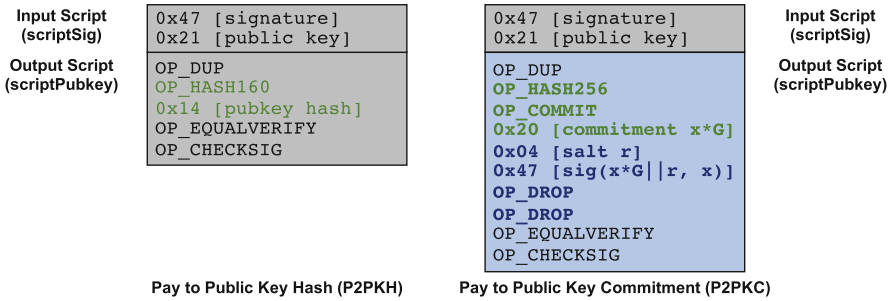


Fig. 9 P2PKC, a slight adaption of P2PKH output scripts, enables the utilization of identifier commitments on the blockchain

Namely, the approach replaces the mutable identifiers with irreversible *identifier commitments* that are *self-verifying* [41]. This means that (a) transactions only hold one-way commitments to the mutable identifiers, i.e., obtaining the original identifier from the identifier commitment is infeasible, and (b) identifier commitments are created in a way such that any tampering with them is easily detectable.

Identifier commitments $C(x)$ for a mutable identifier x are constructed as follows [41]: To commit to x, the transaction creator first computes $x \cdot G$, where G is the generator of Bitcoin's elliptic curve secp256k1. This way, other nodes cannot obtain x from $x \cdot G$ due to the hardness of the discrete logarithm problem for elliptic curves [29]. The user then makes the commitment self-verifying by interpreting x as a private key and $x \cdot G$ as the corresponding public key, which allows her to create an ECDSA signature of $x \cdot G$ that can be verified by itself. Publishing both $x \cdot G$ and the signature thus allows other nodes to verify the correctness of the commitment's signature with the commitment itself. Thereby, nodes can detect any attempts to manipulate either the commitment or the signature for inserting content there. The final identifier commitment released on the blockchain thus looks as follows:

$$C(x) \quad := \quad (x \cdot G,\ r,\ \mathrm{sig}(x \cdot G \| r, x))$$

Besides the commitment and signature, this value includes an additional salt r to mitigate rainbow attacks. The salt is derived from the previous transactions referenced by the new transaction's inputs. The salt thus satisfies a freshness property that prevents content inserters from precomputing and reusing brute-forced identifier commitments that hold content.

Figure 9 illustrates how identifier commitments can be deployed to Bitcoin with only slight adaptions. In this example, we consider *pay-to-public-key-commitment (P2PKC)* output scripts as a hardened replacement for P2PKH output scripts. Instead of the manipulable public key hash x, P2PKC output scripts contain the commitment $x \cdot G$, the salt r, and the signature $\mathrm{sig}(x \cdot G || r, x)$. Notably, the salt and signature are only required for validating the integrity of $x \cdot G$, but spendability only depends on $x \cdot G$. Hence, r and $\mathrm{sig}(x \cdot G || r, x)$ are omitted during the script

validation via the OP_DROP operation. However, P2PKC output scripts must still only be unlocked by showing possession of the right private key. The P2PKC output script uses a new OP_COMMIT operation to recalculate the commitment $x \cdot G$ from the submitted input script. After this adaption, the validation process operates like with P2PKH output scripts. The construction additionally replaces the OP_HASH160 operation of P2PKH output scripts with OP_HASH256 to utilize the order of secp256k1 better. As a consequence, $C(x)$ is robust against simple manipulation, as other nodes can easily detect alterations using the commitment's signature. Furthermore, the manipulable identifier x is only disclosed on-chain if an input script successfully spends the P2PKC output, i.e., x corresponds to a valid private key known by a user.

This construction illustrates that there are additional tools to mitigate blockchain content insertion. However, identifier commitments come at the cost of larger transaction outputs [41]. This overhead further burdens Bitcoin's blockchain, which already suffers from a size of 284 GiB as of October 14, 2020 [16]. In Sect. 6.2, we will further investigate how block pruning can vastly reduce blockchain sizes to ultimately regain capacities for deploying countermeasures against the insertion of objectionable content. First, however, we discuss how alternative blockchain designs can inherently mitigate content insertion through pruning-by-design and through forfeiting the script-based transactions of Bitcoin.

5.4 Alternative Blockchain Models

The main entry point for unwanted blockchain content in Bitcoin is its reliance on easily manipulable script-based transactions. The problem is aggravated by the fact that other nodes have to accept transaction outputs without any assertion that the output can be spent in the future. In this section, we consider the mini blockchain scheme [18] to discuss how alternatives to this script-based design could mitigate unwanted content insertion. However, this section only constitutes a high-level discussion as this design has not yet been thoroughly investigated in this regard.

The main goal of the mini blockchain scheme [18] is to create a blockchain that succinctly represents its current state without keeping track of obsoleted information. The mini blockchain scheme achieves this goal by forfeiting Bitcoin's UTXO set and instead maintains an *account tree*, i.e., the state consists only of the users' balances. Instead of storing a list of accounts, the account tree is realized as a radix trie [51], where the nodes are an account's balance, and the account's address determines the path within the radix trie required to look up the balance. Furthermore, the mini blockchain scheme removes Bitcoin's scripts and only allows for basic operations to manipulate the account tree within a transaction instead. Hence, this alternate model allows for a succinct and account-based state representation but has only limited means to define spendability conditions for a coin.

The scheme's *mini blockchain* is only a short "tail" of full blocks, which is kept to reach consensus about the longest, i.e., accepted, chain of blocks. The mini blockchain thus holds transactions only for a short time before they are pruned entirely. The scheme bootstraps new nodes from only the genesis block via an additionally maintained *proof chain*, which consists only of the chain of all, even pruned, block headers. Compared to Bitcoin's block headers, the block headers here also contain the account tree's root. This data enables joining nodes to (a) get the longest proof chain, (b) obtain and apply the account tree corresponding to this chain, and (c) obtain the remaining full blocks to gain confidence in the chain's validity and finish bootstrapping based on the recent account tree.

Revisiting the basic content insertion methods for Bitcoin (cf. Sect. 3.1), two methods are not entirely prevented by the pruning performed by the mini blockchain scheme: First, users can still manipulate identifiers, i.e., send coins to nonexistent addresses, and these fake addresses would be persisted as paths in the account tree. However, due to the account tree's structure as a radix trie, a content inserter cannot ensure that content chunks spread over different fake addresses remain linkable. The content inserter can hence insert individual paths into the account tree, but this data structure shows a larger resilience to content insertion than, for instance, Bitcoin's UTXO set. Second, using output values to encode data [64] remains possible at first glance if the content inserter sends the funds to adjacent receiver addresses, e.g., `00..00`, `00..01`, ..., `00..FF`. In that case, there is a simple extraction rule for the content. However, other users can undermine these efforts by sending coins to addresses that are suspected to be used for this kind of content insertion. As the account tree only holds the addresses' total balance, initially inserted content can, later on, be destroyed again.

This discussion illustrates that alternative designs can further improve the resilience of a blockchain against content insertion. However, further investigation of other real-world cryptocurrencies diverging significantly from Bitcoin's initial design, such as Monero [68] or Mimblewimble [58], is required.

6 Removing Blockchain Data Without Consensus Changes

We have seen in Sect. 5.4 that pruning-by-design can increase a blockchain's resilience against unwanted content insertion. We now investigate to which extent these benefits can be carried over to already established blockchain systems: First, we present a way for node operators to *locally remove objectionable content* [26] and discuss consequences for the overall network when full nodes exercise selective erasure. Second, we outline a recently proposed *block-pruning scheme for Bitcoin* [43] and how it can deal especially with blockchain content [46].

6.1 Local Erasure of Unwanted Content

Florian et al. [26] proposed enabling full nodes to locally remove content they deem objectionable using *functionality-preserving local erasure (FPLE)*. The main goal of FPLE is to minimize the impact on the erasing node's capability to validate new transactions, which might reference older transactions the node locally erased. FPLE focuses on cases where personal preferences or differences between jurisdictions cause only subsets of users to erase any content perceived as objectionable [26].

Using FPLE, full nodes keep only references to objectionable blockchain data in a local *erasure database*. Via the erasure database, nodes flag individual transaction outputs for erasure, e.g., when transactions contain both content-holding and spendable outputs. Flagging a subset of outputs X of a transaction T for erasure involves four steps [26]: First, the node creates a partially erased transaction $T' = T \setminus X$ and stores T' in the erasure database. Second, the node executes the erasure either by fully deleting all occurrences of T on its hard disk or by replacing those occurrences with T'. Third, the node only relies on T' when validating new transactions and stops relaying any unconfirmed transactions that depend on X. Fourth, the node accepts any transaction depending on X in good faith if another miner includes it in a block, assuming that other nodes did not erase X and validated the transaction properly.

This last step, however, requires further consideration since nodes that locally erase transaction outputs might lose their ability to independently validate new transactions. In the following, consider a node receiving a transaction T_2 that attempts to spend a locally erased output from a previous transaction T_1, i.e., the node cannot validate T_2 based on T_1'. If T_2 is still pending, the node discards it. However, if a (malicious) miner includes T_2 in a block and disseminates that block, the node would accept the block in good faith even if T_2 invalidly spends the output of T_1 [26]. Depending on the fraction of nodes that erased the relevant output of T_1, this scenario could either lead to a temporary blockchain fork or a successful theft of coins [26]. If only a minority accepts the block without validating T_2, the majority, which fully validates T_2, will overrule their decision. In this case, the erasing nodes will revert their wrong decision since the honest majority will not confirm the invalid block. However, if a majority of nodes validate the block based on T_1' instead of T_1, then the block will ultimately be accepted, and T_2 can steal the funds.

The authors argue that this risk of coin theft is generally acceptable since it affects (and thus potentially penalizes) the creator of a transaction holding content that gets erased by a majority of the network [26]. Namely, this scenario is most likely if (i) the content is generally condemned or illegal or (ii) the nodes follow a request to remove privacy-sensitive data [26]. The authors anticipate that nodes can nevertheless regain *trustless validation capabilities* and mitigate the potential for coin theft by hashing the manipulable identifiers of content-holding transaction outputs via a cryptographic hash function instead of fully erasing them [26]. This additional hash layer can then be locally applied again whenever the node validates a pending transaction that depends on erased outputs, but recovering the content from

the rehashed identifiers becomes infeasible. This approach is potentially applicable to the 99 % of Bitcoin's UTXO set that relies on hash-based identifiers [26].

FPLE shows that nodes can locally erase content they find objectionable in a targeted manner. However, FPLE forfeits the node's capability to independently validate all transactions and relies on a majority of nodes *not* to erase the same content. In the next section, we discuss a block-pruning scheme for Bitcoin that allows full nodes to coordinate global pruning phases to reduce the storage dedicated to obsolete information. While the approach mainly focuses on scalability, we highlight how this additional coordination can also benefit the global erasure of blockchain content based on the initial considerations by Florian et al. [26].

6.2 Bitcoin-Compatible Block Pruning

In this section, we discuss CoinPrune [43, 46], a block-pruning scheme that is designed to retroactively enable Bitcoin nodes to forget obsolete blockchain data. While CoinPrune's focus is the scalability of established blockchains with frequent and long-term utilization, pruning can also affect a blockchain's nonfinancial content. First, reducing a blockchain's storage footprint creates capacities for deploying countermeasures that come with an overhead, e.g., the self-verifying identifier commitments we presented in Sect. 5.3. Second, when nodes routinely prune obsolete data, this process may be extended by making further careful decisions about potentially unwanted content to be pruned additionally. Finally, block-pruning schemes must remain aware of higher-level application data. For instance, notary services may rely on `OP_RETURN` transaction outputs [6], which are inherently prunable.

The main goal of CoinPrune is to reduce the blockchain information that is kept by network nodes and distributed to bootstrap joining nodes to a minimum. In theory, joining nodes only need to obtain a *snapshot*, i.e., a deterministic serialization of the current UTXO set (roughly 4 GiB as of October 14, 2020 [16]) instead of a full blockchain copy (roughly 284 GiB [16]), to bootstrap. However, simply relying on a recent snapshot creates trust issues. The joining node has to trust that the snapshot comes from an honest source to prevent inconsistencies with the other nodes' states.

The mini blockchain scheme (cf. Sect. 5.4) faces a similar problem with the initial distribution of the account tree. The scheme solves this problem by referencing the account tree's root in all full blocks and has other nodes reject blocks that reference an invalid account tree root [18]. Bitcoin has no comparable feature. Furthermore, changes to Bitcoin's consensus rules can provoke heated discussions that ultimately can lead to permanent forks [50]. Hence, Bitcoin's block headers cannot easily be extended retrospectively with a commitment to a recent snapshot, similar to the mini blockchain scheme. However, the coinbase field (cf. Sect. 3.1) already provides a dedicated space in the block where miners can include additional values such as a snapshot commitment. Unfortunately, the *interpretation* of these commitments

requires further attention as CoinPrune nodes must not reject blocks based on these commitments when legacy nodes ignore them at the same time.

Recently, *velvet forks* have been proposed to address these kinds of deployability issues and to allow for a *gradual deployment* of new features [36, 73]. Velvet forks introduce new features in a way that legacy nodes are not affected by the rule changes, i.e., (i) updated blocks remain valid to legacy nodes and (ii) upgraded nodes do not discard legacy blocks [73]. CoinPrune uses a velvet fork to gradually deploy its block-pruning scheme by storing snapshot commitments in the coinbase field, i.e., without changing the block structure for legacy nodes. Furthermore, CoinPrune nodes do *not* reject blocks with invalid or missing commitments, but they look for redundant *mutual reaffirmations* of the snapshot commitment. Namely, CoinPrune nodes coordinate the periodic creation of new snapshots in *pulses* (e.g., every 10,000 blocks). During a short *reaffirmation window* following the pulse, CoinPrune miners commit to their most recent snapshot by adding a corresponding cryptographic identifier to the coinbase fields of their blocks. CoinPrune nodes explicitly *tolerate invalid information*, i.e., invalid snapshot commitments, to avoid diverging from legacy nodes. Joining nodes rather look for the snapshot with the most reaffirmations. Assuming that enough honest Bitcoin nodes support CoinPrune, the pulse's true snapshot will accumulate reaffirmations the fastest [46]. This way, CoinPrune enables Bitcoin users to seize the efficient bootstrapping of the mini blockchain scheme. Namely, joining nodes now only have to obtain the chain of all block headers, a recent snapshot, and the full blocks following the snapshot to validate the snapshot's reaffirmations and to finish synchronizing beyond the snapshot's state. Thereby, all CoinPrune nodes can completely prune the historic blocks preceding the snapshot without hurting the network, and they can reduce their storage footprint by 303 GiB for Bitcoin's blockchain up until a block height of 640,000 (June 17, 2020), which reduces initial synchronization times from seven hours to below one hour [46].

Finally, CoinPrune also directly deals with nonfinancial blockchain content in two regards: First, the scheme's pulses provide an additional synchronization point where CoinPrune nodes derive a snapshot from the current UTXO set. Hence, additional considerations beyond the mere serialization become possible. To this end, CoinPrune combines the double-hashing approach proposed by Florian et al. [26] (cf. Sect. 6.1) to *obfuscate* objectionable transaction outputs while remaining capable of validating new transactions spending obfuscated transaction outputs. The nodes only store obfuscated UTXOs and locally rewrite the UTXOs' output scripts during the validation, similar to the P2PKC output scripts we discussed in Sect. 5.3. Specifically, CoinPrune provides the option to locally obfuscate almost all entries of the UTXO set regardless of whether or not the entries contain any content and allows distributing snapshots of obfuscated UTXO sets while retaining full transaction verifiability for all joining nodes [46]. Second, the scheme acknowledges the benefits of storing small data chunks via `OP_RETURN` transaction outputs for providing additional services on top of Bitcoin's blockchain. In a purely UTXO-oriented pruning scheme, all `OP_RETURN` transaction outputs would be pruned and, thereby, break such services. Hence, CoinPrune additionally maintains an

application data storage that contains all `OP_RETURN` payloads in conjunction with meta-information about which blocks or transactions they were part of, which also allows pruning nodes to up look this data.

In conclusion, pruning blockchains promises vast savings regarding storage requirements and improved synchronization processes. Block-pruning schemes should further remain aware of blockchain content insertion to enable extended mitigation strategies and to respect higher-level applications. Finally, the storage savings can enable further changes that are currently prohibited by the blockchain's growth.

7 Retrospective Removal of Unwanted Blockchain Content

Up until now, we discussed smaller tweaks to existing blockchains that aim to mitigate the impact of inserting objectionable content. However, none of the presented approaches globally challenge the immutability property of a blockchain, i.e., once objectionable content enters the blockchain, it resides there indefinitely. In this section, we present alternative blockchain designs that center around the retrospective removal of such content. These *redactable blockchains* extend traditional blockchains with a moderation capability to alter or remove past blockchain transactions. After presenting the initial proposal to enable blockchain redactions by a single moderator or a fixed moderator group, we discuss implications for the permissionless setting, and we briefly outline GDPR-centered blockchain redactions.

7.1 A Cryptographically Redactable Blockchain

In 2017, Ateniese et al. introduced the concept of *redactable blockchains* [5], i.e., an alternative blockchain design that allows redacting transactions retrospectively. In traditional blockchain systems, altering the blockchain after the fact is made impractical through interlinking the blocks and the difficulty to (re)establish these links. For instance, an adversary had to create a proof of work for the altered block and all its successors in Bitcoin. Realizing a redactable blockchain requires weakening this property to carry out the redactions.

To this end, Ateniese et al. [5] replace the traditional cryptographic hash function used to chain blocks together with a *chameleon hash function (CHF)* [4, 5, 37]. CHFs behave as traditional cryptographic hash functions but are parametrized with an asymmetric cryptographic key pair. Everyone can compute a valid hash value using the public key, but knowledge of the private key enables a party to *efficiently compute a collision* for this hash value later on. In the context of blockchains, this means that a party in possession of the CHF's private key can replace a block anywhere in the blockchain without having to recompute all subsequent blocks.

This approach allows establishing a blockchain *moderator*. By additionally relying on secure multiparty computation [11, 62], the private key can further be distributed among a fixed *moderator group*. Whenever objectionable content is brought to the attention of this moderator or the moderator group, the block can be updated to delete the objectionable content, and it can then be distributed to other network nodes.

Before we give a design overview of the redactable blockchain proposed by Ateniese et al. [5], we first provide more background on how CHFs are defined in the context of redactable blockchains. Given a group $\mathbb{G}$ for which the discrete logarithm problem is hard and generator g, the blockchain moderator creates a key pair (k, g^k) and shares g^k with the miners. Miners use a *hashing algorithm* $\mathcal{H}(\cdot)$ to compute the redactable hash value $\mathcal{H}(M, g^k) = (h, \xi)$ for a given message (or block) M. Here, h is the message's hash value and ξ is a random *check value*. Anybody can now use a *verification algorithm* $\mathcal{V}(M, (h, \xi), g^k)$ to ascertain the correctness of h. Knowing the secret trapdoor key k, the blockchain moderator can furthermore use a *collision-finding algorithm* $\mathcal{C}(M', (M, h, \xi), k) = \xi'$ to obtain a new check value ξ' such that $\mathcal{V}(M', (h, \xi'), g^k)$ still yields that h is correct.

We show how CHFs integrate with the blockchain based on the following notation [27]. A block within a blockchain BC is a tuple $B = (s, x, ctr)$, where $s \in \{0, 1\}^\kappa$ is the reference to the block's predecessor; $x \in \{0, 1\}^*$ is the payload, i.e., the set of transactions in the block; and $ctr \in \mathbb{N}$ is the nonce that is gradually increased during the mining process and, for simplicity reasons, is assumed to always start at $ctr = 1$. Further, let the tip of BC, i.e., the most recently mined block, be Head(BC) with identifier s'. The authors further consider the current difficulty level $D \in \mathbb{N}$ and a maximum number $q \in \mathbb{N}$ of allowed tries per block. The model then distinguishes an *inner hash function* $G(\cdot)$ and an *outer hash function* $H(\cdot)$ and defines a newly mined block to be valid if the following conditions hold:

$$H(ctr, G(s, x)) = s' \wedge H(ctr, G(s, x)) < D) \wedge (ctr < q) \stackrel{!}{=} 1$$

Note that the outer hash function only covers ctr and the inner hash value $G(s, x)$. Hence, by replacing $G(\cdot)$ with a CHF, the blockchain moderator can exchange x with a redacted set of transactions x' and compute a collision for the inner hash function without altering the outer hash value. This way, the blockchain remains intact despite the redaction. The block model is extended to account for the CHF, i.e., a block is now a tuple $B = (s, x, ctr, (h, \xi))$, where $\mathcal{H}((s, x), g^k) = (h, \xi)$ is the initial inner hash value. Additionally, the block validity check for a redactable blockchain has the following adapted conditions:

$$(H(ctr, h) = s') \wedge (H(ctr, h) < D) \wedge \mathcal{V}((s, x), (h, \xi), g^k) \wedge (ctr < q) \stackrel{!}{=} 1$$

In summary, now the inner hash value (h, ξ) must be a valid chameleon hash value, and h must yield a valid block in the non-redactable model. If the blockchain moderator now computes a collision $\mathcal{C}((s, x'), ((s, x), h, \xi), k) = \xi'$ based on

a redaction x', then only the CHF verification step for other nodes changes to $\mathcal{V}((s, x'), (h, \xi'), g^k)$, but all other conditions, especially the block interlinking, remain the same.

The redactable blockchains due to Ateniese et al. [5] showcase how novel blockchain systems can be extended to allow for moderation and the removal of objectionable content. However, single blockchain moderators conflict with the decentralized approach of permissionless blockchains, such as Bitcoin. The authors address this problem by presenting a decentralized redaction protocol based on secure multiparty computation, in which multiple blockchain moderators have to cooperate to redact a transaction. While providing a step in the right direction, the scalability of this distributed approach remains questionable due to the performance limitations of secure multiparty computation [74]. Furthermore, blockchain moderators can arbitrarily alter blocks, which means that they are additionally responsible for manually keeping the transaction graph intact, i.e., never redact spent transaction outputs. Finally, CHF-based redactable blockchains do not allow for joining nodes to learn whether a block has been redacted in the past since they remain oblivious to whether they are served the initial check value ξ or a check value ξ' resulting from computing a collision. Next, we thus consider an alternative approach to redactable blockchains that is tailored to the permissionless setting.

7.2 Voting-Based Redactions in the Permissionless Setting

Deuber et al. propose a *voting-based* redaction scheme to prepare redactable blockchains for the permissionless setting where all nodes are considered equal [23]. In the following, we give an overview of how Deuber et al. realize their voting-based redaction scheme based on transparently reaching consensus about accepted redactions instead of relying on dedicated blockchain moderators.

In contrast to the moderator-driven approach by Ateniese et al. [5] (cf. Sect. 7.1), blockchain users *report* transactions they deem to contain objectionable content to the miners, who then engage in a voting period to determine whether to accept or reject the user's report. Users trigger the voting by proposing a *candidate block*. The candidate block is a valid replacement block that redacts content from an already mined block according to a predefined *blockchain policy*. As proposed by the authors, the voting is performed as a simple majority vote over a period of 1024 blocks, i.e., roughly one week of real-world time in the case of Bitcoin [23]. Hence, miners can vote to accept proposed candidate blocks during the voting period. The corresponding candidate block gets accepted by all nodes if at least 50 % of the miners' votes will have approved it by the end of the voting period.

The scheme does not enact the new block by relying on cryptographically replacing the block within the blockchain as the moderated redactable blockchain would (cf. Sect. 7.1). Instead, the checks performed when validating the blockchain are extended to account for potential redactions. To this end, the scheme again extends the block structure, i.e., a block now consists of $B = (s, x, ctr, y)$, where

$y = G(s, x) \in \{0, 1\}^{\kappa}$ is the inner hash value of the block in its initial state. The backlink s' of a block's successor is then computed analogously to traditional blockchain systems in conjunction with the initial inner hash value, i.e., $s' = H(ctr, G(s, x), y)$. If the block is now redacted, i.e., replaced with the candidate block $B^* = (s, x^*, ctr, y)$, the backlink $s' \stackrel{!}{=} H(ctr, G(s, x), y)$ breaks with high probability as $G(s, x^*) \neq G(s, x)$. However, as a fallback, a validating node can substitute $G(s, x^*)$ with the initial inner hash value and check $s' \stackrel{?}{=} H(ctr, y, y)$ instead. If B^* is a valid candidate block and this fallback check succeeds, then the validating node knows that B^* is a redacted version of a formerly valid block since the block was part of the blockchain in its initial state. By extending the initial inner hash value y to a vector y^* of former inner hash values, this scheme also allows for tracking accepted redactions over time in case multiple transactions are redacted as a result of different voting processes. Furthermore and in contrast to the moderator-based approach, Deuber et al. [23] define explicit redaction policies to prevent constructing inconsistent states via an accepted redaction. These policies enforce that (i) candidate blocks can only remove data from a transaction, (ii) the removed outputs must be provably unspendable (e.g., `OP_RETURN` transaction outputs), and (iii) no protocol-specific messages, such as other miners' votes, are redacted [23].

This consensus-based approach to redactable blockchains increases the overall transparency and does not rely on especially trusted parties, which makes it a promising approach for the targeted permissionless setting. However, the required additional considerations trade off the desired trustlessness with increased delays and a reduced scope for possible redactions. On the one hand, the voting period introduces an inherent delay. The proposed voting period of 1024 blocks would, for instance, incur a delay of one week when applied to Bitcoin. During this period, identified illegal content would still be distributed within the blockchain network. On the other hand, the policy for valid redactions is rather conservative to avoid any inconsistencies or ripple effects caused by accepted redactions. As a consequence of this design decision, only provably unspendable transaction outputs can be redacted. However, as we discussed in Sect. 4.2, content is actively being inserted using identifier manipulation. We further established in Sect. 5.1 that nodes cannot reliably determine that a transaction output is indeed manipulated and becomes unspendable as a result. Hence, further improving the reaction times and efficiency of redactions in the permissionless setting is required.

7.3 Redactions for Enhanced Privacy

Up until now, we considered blockchain redactability from a global perspective: the blockchain was rewritten because either a blockchain moderator deemed the edit necessary to protect the network from objectionable content or the network nodes reached consensus among themselves to execute such edits. However, in cases where personal data is also involved, the individual user may request their data to be

removed after it was included on the blockchain, e.g., in the context of the GDPR's "right to be forgotten" (cf. Sect. 2.2). In both previously presented approaches, a user would have to file an editing request with the blockchain moderators or the network nodes to exercise this right. We now present another approach [22] that gives users more direct control over their personal data on the blockchain.

Derler et al. [22] achieve this goal by revisiting chameleon hash functions (CHFs, cf. Sect. 7.1) and extending them with *ciphertext-policy attribute-based encryption (CP-ABE)* [13, 28]. Using CP-ABE, users can decrypt an encrypted message based on policies specified during the encryption step instead of having to get access to a single decryption key [22]. Each user is assigned a set of *attributes* where each attribute corresponds to a secret key. The message creator specifies the *access policy* as a Boolean formula over the available attributes obliviously of the users' assigned attributes [22]. In the context of CHF-based redactable blockchains, the gain promised by integrating CP-ABE is that *users can efficiently redact individual transactions* based on an access policy specified by the transaction creator.

However, this technical integration bears further challenges. For instance, traditional CHFs are fixed, i.e., the same key pair is used for all hash values and their collisions. Each policy requires a different secret key for computing collisions to enforce a fine-granular per-transaction access control [22]. The authors realize this property by integrating CP-ABE not with CHFs directly but with special *chameleon-hashes with ephemeral trapdoors (CHETs)* [19], which require both a long-term secret key and a per-hash-value secret key for computing a collision [22]. Based on CP-ABE and CHETs, the authors then construct *policy-based chameleon-hashes* to be able to implement the possibility of rewriting the blockchain similar to Ateniese et al. [5] (cf. Sect. 7.1) but now on a per-transaction level and according to fine-granular access policies. For efficiency reasons, the scheme expresses *monotone access policies* using monotone span programs [34], which allow positively specifying groups of users that are eligible to redact a transaction based on their attributes and the `and` and `or` operators [22].

This construction allows for fine-grained control over who can rewrite which transaction in a blockchain system. A common use case for such access policies is a mutual agreement between cooperating partners, e.g., when a customer uses a blockchain-backed service. If the customer, later on, revokes the right to process personal data, e.g., by exercising the GDPR's "right to be forgotten" and that data was previously stored on-chain, then the customer can now enact this desire and rewrite the transaction holding the personal data. However, this scenario is orthogonal to the setting we established in Sect. 2, namely, that individual users may deliberately add, and have [9, 48] deliberately added, objectionable or even illegal content to a blockchain with the intent of jeopardizing the system. For such cases, the malicious user must not be allowed to implement a policy that prevents any moderation by other parties. Thus, proper fallback mechanisms are required to hinder objectionable content insertion in blockchains that rely on policy-based redactability.

8 Discussion and Future Research Directions

In this chapter, we discussed blockchain content insertion and presented recent countermeasures to mitigate its negative consequences. However, while the presented approaches tackle blockchain content insertion from different angles, our overview also identifies research questions that remain open. In this section, we thus summarize our findings and highlight further issues that future research should consider to sensibly counteract blockchain content insertion. Corresponding approaches should prevent the insertion of unwanted, objectionable, or even illegal content to the best extent possible while keeping other factors, such as the approaches' efficiency, the trustlessness of permissionless blockchain systems, and the value added by benign blockchain-backed services in mind.

Deeper Understanding of Implications In Sect. 2, we provided a general framework for blockchain content distribution and potential legal consequences. Thereafter, in Sects. 3 and 4, we gave an overview of how content can be and has been inserted into blockchains. We focused on permissionless and Bitcoin-like blockchains used to create ledgers for cryptocurrencies. Further research into understanding the implications and the extent of blockchain content insertion is required to establish a more solid foundation for required remedies, ensure that they are effective, and determine legal obligations. A first research open research question is thus the following: *To which extent are conceptually different blockchain systems prone to unwanted blockchain insertion?* Examples of relevant blockchain systems are privacy-focused cryptocurrencies [58, 68], scaling blockchains [18], or smart contract-based blockchain systems, such as Ethereum [70], that allow for storing arbitrary data by design. Furthermore, the methods applied to insert blockchain content require a deeper understanding. For instance, steganographic approaches started to utilize Bitcoin as an oblivious medium for exchanging messages that cannot be tampered with [71]. Hence, another question for future research is the following: *How can we improve the detection of blockchain content?* The more content researchers can unveil, the better we can understand (all) applications, benign and malicious, of blockchain content insertion. Finally, up until now, we only project past court decisions and views on how general law could be applied in the context of blockchain content insertion and distribution (cf. Sect. 2.2). To conceptualize more sustainable blockchain systems despite the possibility of inserting unwanted and objectionable content, researchers need a dependable legal framework to identify both actionable challenges and potential non-challenges that allow for steering future research efforts. An important question thus remains: *What are the true consequences of illegal and privacy-sensitive blockchain content for honest node operators in permissionless blockchain systems?*

Improve Prevention Mechanisms The mitigation schemes we presented in Sect. 5 are capable of reducing the amount of inserted unwanted blockchain content. Yet, rejecting transactions early on promises to remain the most efficient means of avoiding negative consequences as the immutability of a blockchain does not have

to be weakened by the respective countermeasures. Future endeavors into this area could hence address the following research question: *How can full nodes detect unwanted content more reliably before it enters the blockchain?* Improving the detection rate of unwanted content before it enters a blockchain would reduce the number of cases in which special needs for content removal are required, such as the redactable blockchains we presented in Sect. 7. For instance, tweaking and thoroughly analyzing the deterring effect of mandatory minimum fees (cf. Sect. 5.2) or the costs having to brute-force non-manipulable identifiers (cf. Sect. 5.3) remain open questions as of now. Other works have further questioned the efficiency of these mitigation schemes and advocate for revisiting the applicability of *notice-and-take down* approaches for blockchains instead [60]. In pursuing an increased resilience of blockchain systems against unwanted blockchain content, there remain further open research questions regarding the incorporation of external services. For instance, designs for non-cryptocurrency blockchain systems now regularly rely on external storage systems to store application-relevant data and only insert pointers to such data into the blockchain [45, 55, 56, 75]. In this regard, the interplanetary file system [12] was proposed as a promising candidate to facilitate the storage of files that are to be referenced on a blockchain, and first applications started to rely on it [1, 52]. In anticipation of an increasing need for such applications, another research question is thus the following: *How can different applications be integrated and coexist, e.g., via standardized message formats?* External services can further aid the detection of objectionable content. For instance, Cremona et al. [21] proposed establishing a decentralized lookup table of hash values of illegal content such as child abuse imagery,[2] which could help to reject such transactions before they enter the blockchain. A general research question thus remains: *How can (decentralized) third-party services improve a blockchain's data quality without violating the trustlessness property?*

Swift and Trustless Redactions We provided an overview of different approaches to redactable blockchains in Sect. 7, which all had a different objective. Ateniese et al. [5] initially established cryptographic means to redact objectionable content anywhere in the blockchain. Deuber et al. [23] focused on increasing the trustworthiness of redactions in the permissionless setting. Finally, Derler et al. [22] realized fine-grained per-transaction redactability by cryptographically enforcing access policies. All presented approaches have their individual advantages and disadvantages. Per-transaction access policies help to keep the original data owner in control even when using a blockchain-backed service. However, the approach does not deter malicious transaction creators, who would insert objectionable blockchain content to jeopardize the whole system, even though we have experienced this behavior in the past [9, 48]. The initial redactable blockchain, as proposed by Ateniese et al. [5], explicitly enables a dedicated blockchain moderator to remove any objectionable blockchain content to counteract this malicious behavior. The

[2] A centralized related service is, for example, Microsoft's PhotoDNA service, URL: https://www.microsoft.com/en-us/photodna.

idea of having only one or few permanent blockchain moderators may be sensible for permissioned blockchains controlled by a known set of stakeholders, but it is detrimental to the idea of "trustless" permissionless blockchains where no node has elevated privileges by design. Deuber et al. [23] transfer the idea of redactable blockchains to the permissionless setting, but they ensure the trustworthiness of redactions through a lengthy voting phase. Hence, even if illegal blockchain content has been identified, which will result in an obvious redaction, this content will reside on the blockchain and will be distributed to joining nodes for another week. Hence, an open research question remains: *How to design a redactable blockchain that allows for a swift yet trustworthy response to identified illegal blockchain content?* Another area for future research is the definition of policies for redacting blockchain content. The approach of Ateniese et al. [5] enables the moderators to operate without any oversight. Theoretically, this approach can even create inconsistent states when redactions are not planned carefully (cf. Sect. 7.1). Contrarily, Deuber et al. [23] presented explicit policies for valid redactions. However, these policies are restrictive to avoid inconsistencies (cf. Sect. 7.2), to a point where identifier manipulation cannot be redacted (cf. Sect. 7.2) even though this method is an actively exploited entry point for blockchain files (cf. Sects. 3 and 4). Another open problem thus is as follows: *What are adequate redaction policies to counteract all objectionable content without risking the blockchain's integrity?*

Foster Benign Applications Throughout this chapter, we have concentrated on objectionable blockchain content and its negative consequences. While we argue that protecting blockchain-based systems against such consequences is imperative, novel applications, such as digital notary services [6], conflict resolution services [69], bootstrapping services [44], or even higher-level coordination between blockchain nodes [43], all rely on inserting application-specific data into a permissionless blockchain to leverage its trustlessness property. When investigating further remedies that prevent unwanted blockchain content from being inserted or that reduce the blockchain's size, researchers should keep these benign blockchain-backed services in mind. These services oftentimes rely on the intended insertion methods provided by the coinbase field and `OP_RETURN` transaction outputs (cf. Sect. 3.1). However, there are also arguments for utilizing the unintended insertion methods, such as identifier manipulation. For instance, while this medium can be used to spread illegal content, whistleblowers and politically prosecuted journalists could use it as a means for censorship-resistant publication [42]. Hence, future research efforts could ask: *How can we provide a (moderated) medium for benign blockchain-backed applications services while deterring the spreading of objectionable or even illegal data over the same medium?* Future designs should refrain from using unintended content insertion methods for that purpose but strive to create an additional channel that can properly be moderated by design, in a trustworthy manner, and that integrates well with the underlying blockchain's design principles. While the decision of Money Button to use large `OP_RETURN` fields for content insertion instead of identifier manipulation (cf. Sect. 3.2) already follows the latter

advice, the near-immediate abuse of this feature underpins the necessity for enabling a proper moderation or oversight of such channels.

9 Conclusion

In this chapter, we have systematically analyzed how nonfinancial content enters the blockchains of Bitcoin-based cryptocurrencies, how such content is subsequently distributed through the peer-to-peer network maintaining the blockchain, and what consequences such content distribution can have for blockchain users. Based on this analysis, we discussed recently proposed countermeasures to (a) mitigate the insertion of new blockchain content, (b) deal with excessive blockchain data and unwanted blockchain content after the fact with minimal adaptions to existing systems, and (c) extend blockchain designs to implement moderation processes that allow for retroactively redacting objectionable blockchain content. While these approaches provide promising ideas, we identified further future research directions in this field. These research directions engulf generating a deeper understanding of the implications of blockchain content insertion, improving mechanisms that prevent objectionable content from ever entering the blockchain, researching means to respond swiftly and in a trustworthy manner if unwanted content still enters the blockchain, and fostering benign applications that rely on storing data on permissionless blockchains.

References

1. M.S. Ali, K. Dolui, F. Antonelli, IoT Data Privacy via Blockchains and IPFS, in *IoT* (ACM, New York, 2017)
2. G. Andresen, Block v2, Height in Coinbase (2012). URL https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki, accessed 2021-09-11
3. G. Andresen, Pay to Script Hash (2012). URL https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki, accessed 2021-09-11
4. G. Ateniese, B. de Medeiros On the Key Exposure Problem in Chameleon Hashes, in *SCN* (2005)
5. G. Ateniese, B. Magri, D. Venturi, E. Andrade, Redactable Blockchain—or—Rewriting History in Bitcoin and Friends, in *IEEE EuroS&P* (2017)
6. M. Bartoletti, L. Pompianu, An analysis of Bitcoin OP_RETURN metadata, in *IFCA BITCOIN Workshop* (2017)
7. M. Bartoletti, B. Bellomy, L. Pompianu, A journey into bitcoin metadata. J Grid Comput. **17**, 3–22 (2019)
8. BBC News (2018) 'Child porn links could make Bitcoin blockchain illegal'. URL https://www.bbc.com/news/technology-43485572, accessed 2021-09-11
9. BBC News, Child abuse images hidden in crypto-currency blockchain (2019). URL https://www.bbc.com/news/technology-47130268, accessed 2021-09-11
10. H. Bechtolf, N. Vogel, Datenschutz in der Blockchain—eine Frage der Technik: technologische Hürden und konzeptionelle Chancen. ZD (69) (2018), p. 66

11. M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, in *ACM STOC* (1988)
12. J. Benet, IPFS—Content Addressed, Versioned, P2P File System (DRAFT 3). White paper (2014)
13. J. Bethencourt, A. Sahai, B. Waters, Ciphertext-Policy Attribute-Based Encryption, in *S&P* (IEEE, New York, 2007)
14. Bitcoin Project, Script. URL https://en.bitcoin.it/wiki/Script, accessed 2021-09-11 (2010)
15. Bitcoin Project, Bitcoin Core version 0.9.0 released – OP_RETURN and data in the block chain (2014). URL https://bitcoin.org/en/release/v0.9.0#opreturn-and-data-in-the-block-chain, accessed 2021-09-11
16. Blockchaincom, Blockchain Charts (2011). URL https://www.blockchain.com/charts, accessed 2021-09-11
17. C.M. Bron, Germany–Judicial and Legislative Developments on Internet Child Pornography (2010). Tech. rep., URL https://merlin.obs.coe.int/newsletter/download/154/pdf/en, accessed 2021-09-11
18. J.D. Bruce, The Mini-Blockchain Scheme. White paper (2014)
19. J. Camenisch, D. Derler, S. Krenn, H.C. Pöhls, K. Samelin, D. Slamanig, Chameleon-Hashes with Ephemeral Trapdoors, in *PKC* (Springer, Berlin, 2017)
20. M. Chesnokow, CryptoGraffiti: Permanently Preserve Images on the Blockchain (2016). URL https://news.bitcoin.com/cryptograffiti-images-blockchain, accessed 2021-09-11
21. K. Cremona, D. Tabone, C. De Raffaele, Cybersecurity and the Blockchain: Preventing the Insertion of Child Pornography Images, in *CyberC* (IEEE, New York, 2019)
22. D. Derler, K. Samelin, D. Slamanig, C. Striecks, Fine-Grained and Controlled Rewriting in Blockchains: Chameleon-Hashing Gone Attribute-Based, in *NDSS* (2019)
23. D. Deuber, B. Magri, S.A.K. Thyagarajan, Redactable blockchain in the permissionless setting, in *IEEE S&P* (2019)
24. J. A. Donet Donet, C. Pérez-Solà, J. Herrera-Joancomartí, The Bitcoin P2P Network, in *IFCA FC* (2014)
25. E. Erstu, Cryptograffiti.info (2014). URL http://cryptograffiti.info, accessed 2021-09-11
26. M. Florian, S. Henningsen, S. Beaucamp, B. Scheuermann, Erasing data from blockchain nodes, in *IEEE EuroS&PW* (2019)
27. J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: analysis and applications, in *EUROCRYPT* (2015)
28. V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in *CCS* (ACM, New York, 2006)
29. D. Hankerson, A.J. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography* (Springer, Berlin, 2004)
30. E. Hilgendorf, B. Valerius, *Computer- und Internetstrafrecht*, 2nd edn. (Springer, Berlin, 2012), pp. 60 et seq.
31. HugPuddle, Apertus—Archive data on your favorite blockchains (2013). URL http://apertus.io, accessed 2021-09-11
32. INTERPOL, INTERPOL cyber research identifies malware threat to virtual currencies (2015). URL https://www.interpol.int/News-and-Events/News/2015/INTERPOL-cyber-research-identifies-malware-threat-to-virtual-currencies, accessed 2021-09-11
33. Irish Office of the Attorney General, Child Trafficking and Pornography Act, Section 2. Irish Statue Book (1998), pp. 44–61. URL http://www.irishstatutebook.ie/eli/1998/act/22/enacted/en/pdf
34. M. Karchmer, A. Wigderson, On span programs, in *Proceedings of the Eigth Annual Structure in Complexity Theory Conference* (1993)
35. KG Berlin, Beschl. v. 25.08.2014 - 4 Ws 71/14 (2014)
36. A. Kiayias, A. Miller, D. Zindros, Non-interactive proofs of proof-of-work. *IACR Cryptology ePrint Archive 2017/963* (2017)
37. H. Krawczyk, T. Rabin, Chameleon Signatures, in *Internet Society NDSS* (2000)

38. A. Le Calvez, Non-standard P2SH scripts (2015). URL https://medium.com/@alcio/non-standard-p2sh-scripts-508fa6292df5, accessed 2021-09-11
39. M. Martini, Q. Weinzierl, Die Blockchain-Technologie und das Recht auf Vergessenwerden—Zum Dilemma zwischen Nicht-Vergessen-Können und Vergessen-Müssen. NVwZ **36**17, 1251–1259 (2017)
40. R. Matzutt, O. Hohlfeld, M. Henze, R. Rawiel, J.H. Ziegeldorf, K. Wehrle, POSTER: I Don't Want That Content! On the Risks of Exploiting Bitcoin's Blockchain as a Content Store, in *ACM CCS* (2016)
41. R. Matzutt, M. Henze, J.H. Ziegeldorf, J. Hiller, K. Wehrle, Thwarting unwanted blockchain content insertion, in *IEEE BTA Workshop* (2018)
42. R. Matzutt, J. Hiller, M. Henze, J.H. Ziegeldorf, D. Müllmann, O. Hohlfeld, K. Wehrle, A quantitative analysis of the impact of arbitrary blockchain content on bitcoin, in *IFCA FC* (2018)
43. R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, K. Wehrle, How to securely prune bitcoin's blockchain, in *IFIP NETWORKING* (IFIP, New York, 2020)
44. R. Matzutt, J. Pennekamp, E. Buchholz, K. Wehrle, Utilizing public blockchains for the sybil-resistant bootstrapping of distributed anonymity services, in *ACM ASIACCS* (2020)
45. R. Matzutt, J. Pennekamp, K. Wehrle, A secure and practical decentralized ecosystem for shareable education material, in *ICOIN* (IEEE, New York, 2020)
46. R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, K. Wehrle, CoinPrune: shrinking bitcoin's blockchain retrospectively. IEEE TNSM **18**(3), 3064–3078 (2021)
47. E. McReynolds, A. Lerner, W. Scott, F. Roesner, T. Kohno, Cryptographic currencies from a tech-policy perspective: policy issues and technical directions, in *IFCA FC* (2015)
48. Money Button, Against illegal content on the blockchain (2019). URL https://blog.moneybutton.com/2019/01/31/against-illegal-content-on-the-blockchain, accessed 2021-09-11
49. Money Button, How We Added Support for Giant OP_RETURN Data in Money Button (2019). URL https://blog.moneybutton.com/2019/01/26/how-we-added-support-for-giant-op_return-data-in-money-button, accessed 2021-09-11
50. D. Morgan, The Great Bitcoin Scaling Debate—A Timeline (2017). URL https://hackernoon.com/the-great-bitcoin-scaling-debate-a-timeline-6108081dbada, accessed 2021-09-11
51. D.R. Morrison, Patricia–practical algorithm to retrieve information coded in alphanumeric. J. ACM **15**(4), 514–534 (1968)
52. N. Nizamuddin, K. Salah, M. Ajmal Azad, J. Arshad, M. Rehman, Decentralized document version control using ethereum blockchain and IPFS. Comput. Electr. Eng. **76**, 183–197 (2019)
53. Office of the Law Revision Counsel of the United States House of Representatives, U.S. Code, Title 18, Chapter 110, §2256 (1996)
54. K. Okupski, Bitcoin Developer Reference. White paper (2014)
55. J. Pennekamp, F. Alder, R. Matzutt, J.T. Mühlberg, F. Piessens, K. Wehrle Secure end-to-end sensing in supply chains, in *CPS-Sec* (IEEE, New York, 2020)
56. J. Pennekamp, L. Bader, R. Matzutt, P. Niemietz, D. Trauth, M. Henze, T. Bergs, K. Wehrle, Private Multi-Hop Accountability for Supply Chains, in *BIoTCPS* (IEEE, New York, 2020)
57. A. Peters, Sabotage von Blockchains durch Einschleusung strafrechtsrelevanter Inhalte?, in *Rechtsfragen digitaler Transformationen*, ed. by J. Taeger, 1st edn., OlWIR (2018), pp. 392 et seq
58. A. Poelstra, Mimblewimble. White paper (2016)
59. Protection of Children Act, Chapter 37, Section 7 (2015). http://www.legislation.gov.uk/ukpga/1978/37
60. M. Schellekens, Does regulation of illegal content need reconsideration in light of blockchains? Int. J. Law Inf. Technol. **27**(3), 292–305 (2019)
61. J. Schrey, T. Thalhofer, Rechtliche Aspekte der Blockchain. NJW (1434) (2017), p. 1431
62. A. Shamir, How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
63. K. Shirriff, Hidden surprises in the Bitcoin blockchain and how they are stored: Nelson Mandela, Wikileaks, photos, and Python software (2014). URL http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html, accessed 2021-09-11

64. M.D. Sleiman, A.P. Lauf, R. Yampolskiy, Bitcoin message: data insertion on a proof-of-work cryptocurrency system, in *2015 International Conference on Cyberworlds (CW)* (2015)
65. A. Sward, I. Vecna, F. Stonedahl, Data Insertion in Bitcoin's Blockchain. Ledger 3 (2018)
66. G. Taylor, concepts of intention in German criminal law. Oxf. J. Leg. Stud. **24**(1), 99–127 (2004)
67. User "unwriter", B://—Bitcoin Data Protocol (2019). URL https://github.com/unwriter/B, accessed 2021-09-11
68. N. van Saberhagen CryptoNote v 2.0. White paper (2013)
69. E. Wagner, A. Völker, F. Fuhrmann, R. Matzutt, K. Wehrle, Dispute resolution for smart contract-based two party protocols, in *ICBC* (IEEE, New York, 2019)
70. G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow paper (2014)
71. M. Xu, H. Wu, G. Feng, X. Zhang, F. Ding, Broadcasting steganography in the blockchain, in *Digital Forensics and Watermarking* (2020)
72. A. Yeow, Bitnodes: Global Bitcoin Nodes Distribution (2018). URL https://bitnodes.earn.com/dashboard/?days=730, accessed 2021-09-11
73. A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, W.J. Knottenbelt, A wild velvet fork appears! inclusive blockchain protocol changes in practice, in *IFCA BITCOIN Workshop* (2018)
74. J.H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, K. Wehrle, Secure and Anonymous Decentralized Bitcoin Mixing. Futur. Gener. Comput. Syst. **80**, 448–466 (2018)
75. G. Zyskind, O. Nathan, et al., Decentralizing privacy: using blockchain to protect personal data, in: *S&P Workshops* (IEEE, New York, 2015)

Blockchain-Based Distributed and Secure Digital Forensic Investigation Systems

Mauro Conti, Gulshan Kumar, Chhagan Lal, and Rahul Saha

1 Introduction

With the rapid digitalization of our everyday activities and the usage of cutting-edge features of Information and Communication Technology (ICT) in business activities and government services, many individuals' lifestyle has significantly improved. However, it also makes people more dependent on communication technologies, smart appliances, Cyber-Physical Systems (CPSs), Internet of Things (IoT) services, and cloud computing services. This high dependence between the ICT and the physical world has resulted in a rapid increase in threats and cybersecurity issues (e.g., identity fraud, cyberharassment, sensitive data leakage, distributed denial of service, and advanced malware targeting smart devices). Therefore, the prompt identification of cyber activities (or criminals) and the collection and management of high-quality evidence, which is later needed in the courtroom, are of great significance. Unfortunately, the DFs procedure exhibits various challenges such as efficient identification and collection of evidence, custody of evidence chain, secure and privacy-preserving evidence management, and integrity and suitability of evidence presented in a court of law. Moreover, extracting evidence from cloud services, mobile devices, smart wearable devices, pervasive Internet access, and

M. Conti
Department of Mathematics, University of Padova, Padova, Italy
e-mail: conti@math.unipd.it

G. Kumar · R. Saha
Department of Mathematics, University of Padova, Padova, Italy

School of Computer Science and Engineering, Lovely Professional University, Bengaluru, India

C. Lal (✉)
Department of Intelligent Systems, Delft University of Technology (TU Delft), Delft, Netherlands

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_11

new digital wallet platforms spans the investigations to international borders and is subjected to heterogenous jurisdictions and legal procedures. Finally, the availability of a large amount of data in heterogeneous formats at various platforms, usage of highly different ICT technologies, and sophisticated modern software or hardware architectures add new challenges in effective DFs investigation.

To address different challenges caused by the abovementioned new technologies (i.e., IoT, CPS, and ICT) and to provide the essential features (i.e., tamper-resistant, traceability, auditability, and privacy preservation) in an evidence management system, recently, the researchers are envisioning the usage of blockchain technologies in DFs domain. Blockchain is a distributed and append-only data structure with inherent features such as data integrity, immutability, transparency, traceability, relieve single point-of-trust, and privacy-preserving. These unique features of blockchain make it a promising technology for building a decentralized and secure digital evidence management system. Moreover, blockchain's decentralized nature is a good match for the requirements (e.g., integrity and provenances) of evidence collecting across jurisdictional borders. However, the researchers proposing blockchain to improve the forensics system should ensure that the proposed framework is practical. Its deployment is feasible concerning the computation costs, computation overhead, storage requirements, and usability. For instance, storing data directly to the blockchain is not efficient. Other options like offchain storage with InterPlanetary File System (IPFS) should be explored. Moreover, the framework should support certain features such as witness and juror privacy, access control for evidence storage/retrieval for the chain of custody assurance along the forensic workflow, and lightweightness needed in forensic investigation, but are not inherent in blockchain technology.

In recent years, many blockchain-based forensics investigation frameworks have been proposed to investigate and exploit the potential of blockchain technology for various areas of DFs. For instance, authors in [1–3] present blockchain-based forensic applications for vehicular networks. Some other researchers like [4–6] and [7] propose blockchain-based distributed forensic schemes for IoT networks. Similarly, authors in [8, 9] and [10] present blockchain-based forensic techniques for cloud computing services. Furthermore, authors in [11] and [12] use blockchain to secure the Chain of Custody and digital evidence preservation framework and in [13] for photo forensics. All the abovementioned research works show promising results concerning blockchain usage's feasibility to improve the DF investigation process. However, the research on blockchain-based forensics is still in its early stages. It is necessary to understand the key advantages, drawbacks, and challenges associated with it before such a framework is deployed in a real work forensics domain. To this end, this chapter has the following contributions.

- We survey the state-of-the-art research on blockchain-based DF schemes, and we provide the key benefits along with the challenges and issues that need to be addressed before its usage in the real world.
- We present a generic blockchain-based DF investigation framework and discuss its different components along with their working methodology. We provide

details about how a blockchain-based DF framework can efficiently cover the whole lifecycle of DF that includes its various phases such as evidence identification and collection, evidence processing, evidence presentation in a courtroom, the decision of a court trial, and criminal imprisonment-related information.
- Finally, we present the open issues and challenges in the usage of blockchain in DF investigation along with possible solutions to address them. Moreover, we provide future research directions that need attention from the research community working in this area.

The rest of the chapter is arranged as follows. In Sect. 2, we present the essential background needed to understand the basic concepts of digital forensics and blockchain technology. In Sect. 3, we present the design and working methodology of our proposed blockchain-based generic DF investigation framework. Section 3 also presents a survey on the state of the art on blockchain-based DF solutions, along with their benefits and drawbacks. In Sect. 4, we discuss challenges and future research directions in blockchain-based DF solutions. Finally, we conclude our work in Sect. 5.

2 Overview of Digital Forensics and Blockchain Technology

In this section, we present a brief overview of digital forensics and blockchain technology. Please note that in this chapter, we survey only the blockchain-based DF solutions. Comprehensive details about major digital forensics areas (such as cloud forensics, network forensics, mobile device forensics, and IoT forensics) and blockchain technologies are out of the scope of this chapter. We direct the interested readers to the following high-quality survey articles that are specifically dedicated to these areas [14–17].

2.1 Digital Forensics

Digital Forensics (DF) is considered an important branch of forensic science. It concerns the extraction and investigation of artifacts (sometimes also called evidence) found in one or more electronic devices. The aim is to use the artifacts to understand and retrace the chain of events that must have led to the generation of these artifacts in the first place. In particular, the DF investigation life cycle consists of a series of steps that include identification, extraction, examination, storage, analysis, and presentation of evidence in a court of law [18]. These steps should be performed by taking into account the predefined guidelines, scientifically proven methodology, and tools that could help the presenter reconstruct events based on the evidence. Thus, it ensures the security and integrity of the custody of the evidence

chain. To keep pace with the rapid advances in electronic devices and information technology, the DF investigators and analysts continuously work on developing new tools and standards to ensure the correct and timely extraction of electronic data.

We subdivide the DFs into four key domains: mobile forensics, cloud forensics, streaming services and devices forensics, and IoT forensics. Below we provide a brief overview of these forensics areas and their key features and importance in the DF investigation process.

- *Mobile forensics:* The fundamental principle on which the DFs operate is that evidence extraction, preservation, and processing have to be done correctly to ensure that it is admissible in a court of law. Present-day mobile devices exhibit high computational and communication capabilities to keep up with the rapidly growing application requirements. This enabled mobile devices (e.g., smartphones, PDAs, and tablets) to execute highly sophisticated tasks up to the level where these devices could replace the conventional desktop systems and laptops. Due to their mobile nature and small size, these are more prone to be targeted by adversaries to perform malicious activities. The investigation for the incidents related to mobile devices resulted in the emergence of mobile forensics, which is considered a DF subdomain. During DF investigation, when the evidence is stored on mobile devices (e.g., smartphones and tablets), then various techniques could be used for its extraction and analysis [19]. The choice of a data extraction technique depends on several factors such as the type of mobile device and its underlying operating system and security configurations [20]. The whole forensic process can be carried out in the following steps: seizure, extraction, and analysis.
  The broad range of usage (e.g., PDA, calendar, audio/video calling, navigator, and messaging) of mobile devices in its owner's daily life makes it one of the most important sources of evidence during the DF investigation. The fact that the mobile device could be in continuous possession of its owner makes it wealthy of evidence whose usage could be wide-reaching. For instance, the data extracted from a mobile device could reconstruct most aspects of its owner's daily life. A non-comprehensive list of evidence extracted from a mobile device includes [21] user call history, contact information from phonebook, text and voice-mail messages, image/audio/video files, Internet browsing history, events stored in a calendar, past geo-location and GPS data, WiFi connection information (including access points to which the devices connected in the past), and stored documents like Word/Excel/presentation files and other user-created data. In mobile devices, the DF mainly focuses on memory modules since they are the key source of electronic data that later could be part of the evidence. The data available in the memory of a device results from its usage, i.e., the interaction between a device and its owner and between a mobile device and other devices through some networking infrastructure or the Internet.
  Due to the increase in the number of sensors, communication technologies, and mobile apps and large storage in today's mobile devices, the functionalities supported by these devices have greatly increased. It further enables creating,

transferring, and storing huge amounts of sensitive information and personal data on these devices. This information and its associated metadata can be a valuable source of digital forensic evidence in different types of investigations (e.g., civil, criminal, and military). Cybercriminals are continuously trying to exploit these devices when performing unlawful activities to access this rich set of data. In the recent past, mobile devices are well known for having weak security against an array of threats. To defeat the advanced defense techniques, the adversaries combine technical exploitation (e.g., banking Trojans, information stealers, and botnets) and social engineering tactics (e.g., TimpDoor, Gustuff, and XcodeGhost). Due to these mobile devices' low security, the evidence collected from them could be questioned in court as the suspect could use the "the Trojan did it" line for her defense. In some cases, it could be possible that the suspect is telling the truth about her mobile device being attacked. In literature, a large number of data acquisition algorithms and evidence extraction process schemes exist. However, this heterogeneity coupled with the rapidly changing technology landscape contributes to multiple challenges to the DF investigation process. Authors in [22, 23] provide a detailed review on state-of-the-art mobile device forensic techniques along with a discussion on future research opportunities that should be explored to envision more efficient solutions in the future.

- *Cloud forensics:* Over the years, cloud computing has emerged as one of the most significant technology in ICT services. The heterogeneous services offered by cloud service providers (CSPs) not only support numerous new technologies and provide economic benefits, but it has also completely changed how various computing activities are carried out in today's market. Still, several cloud customers avoid moving their IT requirements to the cloud platform, as they are concerned about the security and privacy threats against the cloud. The easy application interface (through virtualization services) that CSPs provide to its customers to hide the underlying complexities makes the system non-transparent to its customers, thus making it untrustworthy for few customers. The distributed nature and heavy use of virtualization technology in cloud computing also make it challenging to perform efficient digital forensic investigations. Moreover, the unique challenges such as cross-border jurisdiction, dependence on CSPs, multi-tenancy, and data duplication make it harder to locate, identify, and separate the possibly compromised targets for DFs.
  Since the design of cloud systems has hardly considered the issues of DFs like evidence preservation and integrity, investigators face challenges from various view-points (e.g., technical, legal, and organizational) while extracting evidence from these systems. The technical issues concern the standards and tools required to perform the forensic investigation in a cloud platform. It includes processes like data collection, real-time forensics, evidence segregation, and proactive actions, while the legal issues regard with the development of regulations and contracts to ensure that forensic activities do not breach laws and regulations in the jurisdictions where the data resides or is collected, simultaneously ensuring the confidentiality of co-tenants who share the same

infrastructure. Finally, the organizational issues concern organizational aspects of the forensics, which mainly include various system entities such as CSPs, customers, advisers, incident handlers, and system objects (e.g., binding service level agreements, policies, and guidelines).

In a cloud environment, it is also possible that the evidence resides on the client (aka customer) side instead of the cloud servers. This is specifically true in software-as-a-service (SaaS) and platform-as-a-service (PaaS) scenarios, where investigators' access to evidence data and information might be minimal or entirely absent. In such scenarios, the most crucial step is timely access to the evidence because the suspect could destroy the data and key evidence. Another key challenge in cloud forensics is getting physical access to a cloud environment because of its hardware components' geographically distributed nature in different jurisdictions. Moreover, the gathered data for evidence purposes could be available in encrypted form, which is not useful for analysis until the correct decryption keys are found. Although there are various research efforts to improve security and privacy for cloud services, very few works focus on developing theoretical and practical solutions for cloud forensics.

- *Streaming services and devices forensics:* Recently, there has been a rapid increase in cybercrimes that use live video streaming services and social networking platforms to broadcast illegal content, such as child pornography, hate speeches, illegal streaming of copyrighted content, and broadcasting live videos of vandalism. One key point that forensic investigators have to consider in streaming services is the *time factor* because it is quite possible that as the time between reporting an illegal stream and investigator to review it increases, the possibility of finding any traces of the content that was streamed deceases. Such time constraints are not applied to the other types of forensics and are only true for streaming services-related crimes, making such crimes prove in a court of law very challenging. In particular, the main problem associated with streaming illegal content is that there is a possibility that this content is not ever being stored on any device. Therefore, the law enforcement authorities face content unavailability, and thus, there is no content to analyze and use as evidence. Nowadays, the social live streaming services (e.g., TikTok and LiveMe) are very popular with young users. Thus, it is essential to design and implement effective, real-time techniques to detect and report any deviant behaviors to keep the users safe. However, there is a lack of effective solutions to detect or prevent such deviant behaviors that led to problems, such as the exploitation and grooming of young users by radical organizations or pedophiles. Moreover, the area of streaming services is still quite new for forensic investigators, and there are no efficient tools to fight the cybercrimes that exist on these platforms.

  Another issue that comes into the domain of streaming services forensic is related to copyright infringement in streaming applications and devices. Nowadays, streaming multimedia services such as YouTube, Netflix, and Amazon Prime dominate traditional multimedia. These new services are available with a paid subscription, and it provides premium multimedia content to billions

of users worldwide. The content could be made available on many devices (including mobile devices such as smartphones and tablets), making it easy to copy. There are many applications and services (e.g., torrents) that make this copied content for free. Currently, there is a huge focus on streaming content on smart devices by publishing apps on Android or Google Play stores. The idea is that smart devices, such as smartphones, tablets, but most importantly TVs, will install a streaming app through which the users can get access to the multimedia content. Many users that do not want to pay for the content move toward illegal content providers; these providers either capitalize through monetization with embedded ads, or they could place some malicious code when the user installs their app, thus turning the user devices into bots that they can later use for any malicious purpose. Presently, despite the industry need, the related work is limited, and toolkits to handle such crimes are not available for investigators.

Some of the streaming devices, such as CCTV cameras installed at public places and security buildings, also store the video for few hours (i.e., before it gets overwritten). In such cases, identifying the source of the video presented in the court of law (as a piece of potential evidence) to prove a crime is a major component of video forensics, for instance, if a video camera captures a suspect's activities and the recorded video is used in the court to provide an alibi to the suspect. It is essential to provide proof that the evidence is extracted from the claimed surveillance camera. Also, every year, in the movie industry, significant revenue loss occurs due to illegal recording in movie theaters followed by illegal distribution. The video source identification could help to track down the device and person who have committed the piracy crime in the first place. Moreover, the source identification could help to regulate individual video sources. Even though source identification research efforts are still at early stages, the most straightforward way for source identification is to embed the digital watermarks while recording the video. However, this technique incurs a high computational cost, and it requires modification in recording devices, thus not suitable for most off-the-shelf cameras.

- *IoT Forensics:* The recent technological and hardware advances have enabled the rapid deployment of millions of smart and adaptive devices (all interconnected through the Internet) in various critical domains such as home automation, intelligent transportation systems, healthcare, and environmental control. As a result, today, we see and use several IoT applications in our daily life activities. The usages of these applications have greatly improved the users' lifestyle, but it also brings a whole new set of challenges for DF investigators. A relatively new and less-explored subdomain of DF is IoT forensics. The investigators deal with resource-constrained IoT devices to identify and extract digital data in a legal and forensically sound way. The heavy use of IoT devices (e.g., sensors) makes IoT data a rich source of evidence for crimes committed in these ecosystems. However, the investigators have to face several diverse challenges, starting from the heterogeneity of devices, and nonstandard data storage structures, to the multi-tenant cloud platforms and the resulting cross-border investigation warrants. Moreover, the memory limitation (due to the

small size) of collection devices may cause the evidence data to be frequently overwritten, and some devices cannot report evidence data on time. Therefore, considering its volatile nature, the digital information must be extracted and analyzed using the correct set of tools and techniques that ensure the Chain of Custody's integrity for the collected and presented evidence.

Most IoT devices have inbuilt sensors or actuators that generate data, depending upon the type of device, sometimes autonomously (e.g., periodically, depending upon room temperature) and sometimes in response to human actions (e.g., motion detection, door opening). The active and continuous data generation nature of these devices makes them capture traces of the surrounding environment or user activities. Later, these traces can be used as evidence in investigations, thus making them adequate digital witnesses. However, to use IoT devices as sources of evidence, the investigators should handle the quantity of data generated, the network scalability, and the heterogeneity of devices and communication protocols used. For instance, a network's devices could use end-to-end encryption-enabled data communication protocol to ensure the communicated data's privacy. Still, the use of encryption makes the extraction and analysis of data challenges for investigators. User privacy is essential in IoT scenarios where the devices handle users' sensitive data, such as smart medical devices (e.g., fitness trackers or remote health monitoring systems), which deal with personal data that includes users' medical files prescriptions and current health status. When such IoT systems are connected with cloud platforms that store the sensitive data generated by these devices, the cloud service providers have to sign confidentiality agreements with their customers. This will cause difficulties for investigators when they need access to the data stored in the cloud because the CSPs could deny access to the shared memory, as it might reveal data of customers who are not involved in the investigation. Particularly in a multi-tenancy context, there are minimal techniques to create a forensic image without violating legal or ethical considerations.

2.2 Blockchain Technology

Blockchain gained sudden popularity after its implementation as a key underlying technology in the Bitcoin cryptocurrency system. Blockchain is an open, decentralized database (also known as the distributed ledger) that stores financial and non-financial transactions. With the help of hashing, the transactions (and blocks) are chained together to keep track of the stored data, and it uses digital signatures to keep track of the data owners (but not their actual identities). Since the ledger is distributed, each participant in the blockchain network can keep a local copy of the global blockchain. The blockchain uses cryptography techniques and digital signatures to prove identity and authenticity and enforce read/write access rights. To add a new block in the blockchain, the participants have to go through a consensus mechanism, and the block is only added if it is approved by majority.

Depending upon the accessing of blockchain and its maintenance procedures, it could be implemented with the following modes of operation: public (also known as permissionless) and private (also known as permissioned). In a permissionless blockchain, anyone can access the ledger information (i.e., join the network and read or send transactions) and participate in the consensus mechanism (i.e., maintain the ledger). Most cryptocurrencies (e.g., Bitcoin and Ethereum) and few open-source blockchain platforms are public blockchain systems. In permissioned blockchain, participants have restrictions on read or write operations, and only a pre-defined set of participants can take part in the consensus mechanism. The specific type of blockchain system where the ledger keeps a record of transactions coming from different organizations is termed a consortium blockchain. In particular, the permissioned blockchain can be implemented as public and private permissioned systems. Since both systems are permissioned, only the authorized entities can perform various operations on the blockchain depending upon their granted permissions. But there is one key difference between these two, which concerns the access to reading operation on data stored at blockchain. Anyone who shares the ledger can perform the read operation in permissioned public while restricted in permissioned private.

A BC can use Smart Contracts (SC), which are securely stored on the BC and are executed manually (via a transaction invoking a function in it) or automatically (when a precondition evaluates to true). Specifically, it allows for decentralized automation by facilitating the verification and enforcement of conditions written in the underlying contract [24]. In this way, SCs serve as agreements or a set of policies that supervise a transaction. For instance, an SC can define a set of rules for an individual's travel insurance, which triggers the contract execution when a traveling carrier (such as flight or train) is experiencing delay by more than a fixed amount of time. In particular, an SC consists of instructions or operations written in special programming languages (e.g., Solidity and Go Lang). It gets executed upon the fulfillment of predefined conditions. The key property that makes SCs of great use in many real-world applications is their ability to eliminate the requirement of a trusted third party in multiparty interactions [25]. It is achieved to enable parties to participate by performing secure peer-to-peer transactions over BC without placing their trust in outside parties, which are generally used to ensure that all parties fulfill the contractual obligations. Examples of application areas where SCs could provide benefits are insurance, real estate, supply chain management [26], data management and sharing, access control and identity management [27], and voting. Moreover, with the advancements in the Internet of Things (IoT) domain, SCs could also play a vital role in machine-to-machine communications [28]. Currently, the largest BC platform for SCs deployment is Ethereum [24]. It uses Solidity, a high-level scripting language that is specifically designed to write SCs [29]. Solidity is influenced by common programming languages, including Python, C++, and JavaScript. Thus it provides features such as inheritance, libraries, and user-defined types, which help the developers to depict complex business logic using them.

3 Blockchain-Based Digital Forensics Techniques

In this section, first, we provide the benefits that blockchain and smart contracts provide in the domain of digital forensics (please refer to Fig. 1). Next, we present the possible design of a generic blockchain-based digital forensics (BCDF) architecture and discuss its working methodology by considering the involved entities and components. The interaction between the different entities and components is also provided to describe the workflow in the architecture. As per our knowledge, all the BCDF architectures available in the state of the art can be derived from this architecture. Finally, we provide a review of the state-of-the-art BCDF solutions by classifying them into IoT, cloud, and other categories. The benefits and challenges of the reviewed research works are also provided to highlight the advances and research gaps in digital evidence management by envisioning blockchain technology.

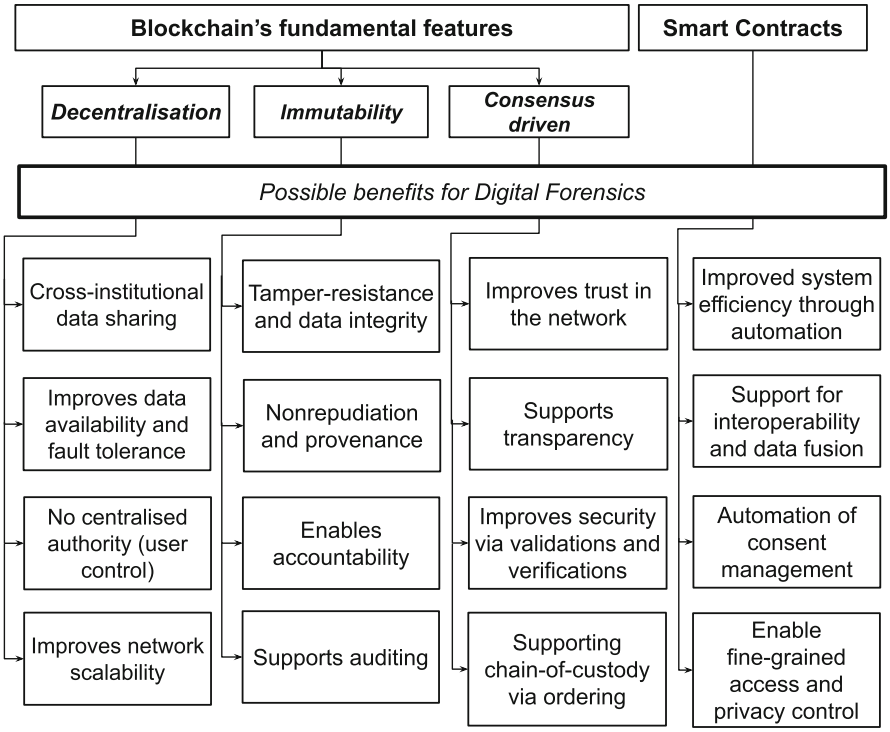


Fig. 1 Benefits for digital forensic domain envisioned by the usage of blockchain technologies and smart contracts

3.1 Blockchain-Based Digital Forensics Architecture

The generic architecture of a BC-based digital forensics system might consist of various entities such as a trusted authority, a police department, a court, a prison, victims, prosecution lawyers, defense lawyers, police investigators, crime scene analysts, witnesses, monitoring devices, a judge, a jury (with jurors), a hospital, a repair factory, and an insurance company as depicted in Fig. 2. There could be other specific entities involved that assume the role of a user or miner. All the entities shown in Fig. 1 are the users of the system. Simultaneously, some of them are also miners, i.e., they run the consensus protocol and execute smart contracts to govern the various functionalities of the system and maintain the distributed ledger. Next, we briefly describe the role of these entities in the system.

- Police Investigator: It collects evidence from different sources like a victim, witness, and crime scene analyst. Once collected, the evidence details are documented, a hash of the document is created, which is uploaded on the distributed ledger, while the evidence is stored in the police department. The hash will allow others to verify the evidence at a later stage of the investigation. If the verification process fails, then this mismatch can be reported to the TA, who can identify the true identities of the entities that have accessed the data after its original store operation.

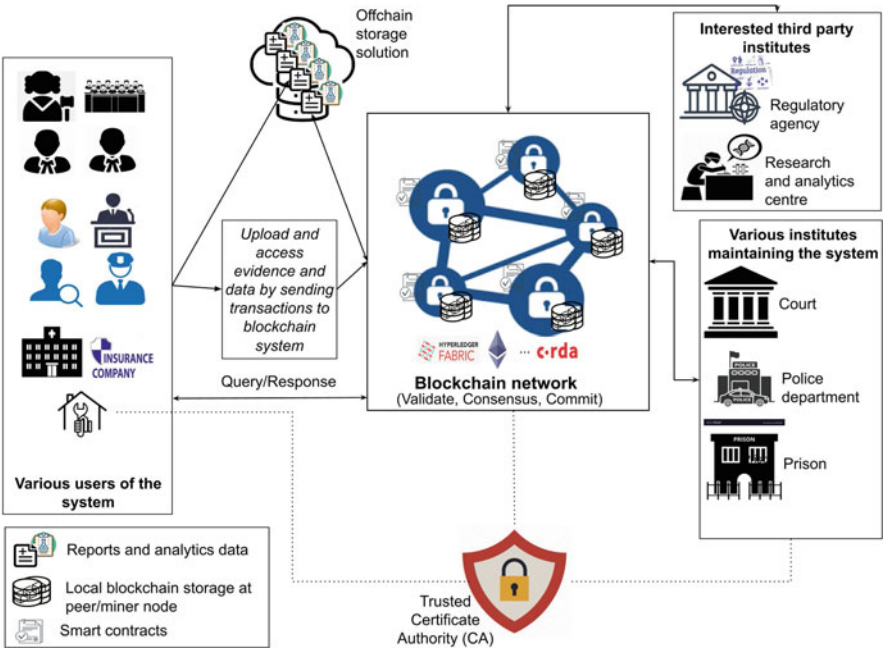


Fig. 2 Generic blockchain-based digital forensic architecture

- Victim: It shares evidence with the police investigator supporting its allegations.
- Witness: It acts as an important source of critical information related to a crime or crime scene. A police investigator can use the information to identify a criminal or collect evidence related to a crime.
- Monitoring Device: Such surveillance devices are usually deployed by individuals or by government organizations at different locations in a city. When a crime is committed within the proximity of such a device, the police investigators can seize them as evidence for further investigation of the crime.
- Crime Scene Analyst: At a crime scene, there could be evidence that is hard to extract due to its nature (e.g., biometric traits). A specialized team collects such evidence (usually termed forensic experts). They collect and analyze evidence and send their analysis results to police investigators.
- Prosecution Lawyer: During court trials, the charges are filed by the prosecution lawyer against the criminal. A copy of these charges is uploaded to the ledger.
- Defense Lawyer: When the criminal is presented before the court, his/her lawyer pleads on his/her behalf and uploads the corresponding data to the ledger.
- Juror: These are the selected individuals that cast their vote directly on the ledger. To ensure the safety of jurors, their identity and vote might need to be kept confidential.
- Judge: Based on the votes extracted from the ledger, the judge will provide a decision for a specific trial. A hash of the result report will also be uploaded to the ledger.
- Police Department: It acts as one of the consensus nodes in the consortium blockchain system, and it stores all the evidence related to specific cases.
- Court: The trials are carried out, and it has the responsibility of preserving court data by uploading it on the blockchain system.
- Prison: Based on the data received from the blockchain system, it verifies the prisoners' sentence conditions and keeps the prisoners. It might also upload some details about the prisoners it keeps.
- Trusted Authority (TA): It is a government organization that initializes the whole system. It also generates the crypto material and performs the key and identity management.
- Hospital: In case if a patient is part of some crime investigation, then it treats the wounded and uploads their treatment reports on the ledger.
- Repair Factory: In case if a vehicle or device is evidence in an ongoing criminal investigation, then it repairs the damaged objects (e.g., mobile devices and vehicles) and uploads the repair reports.
- Insurance Company: Based on the result of a court trial or company policies, it compensates different entities and uploads the corresponding insurance reports.

Figure 2 shows the various entities and their possible interactions in a blockchain-based digital forensic (BCDF) architecture. In particular, these entities can be broadly classified into three categories. First category entities are the **users of the system** that submit and access (read, write, and update) data to and from

the blockchain system by generating corresponding transactions. The transactions submitted by these users are validated by the second category of entities which includes various **institutions** that are part of the consortium blockchain. These institutions also maintain the blockchain system and the distributed ledger via consensus nodes. A transaction is only added in the ledger if most consensus nodes agree on it or the transaction satisfies some predefined set of policies inserted via smart contracts. The third category of entities is the **interested third parties** that might need the data from the blockchain for different purposes. For instance, a government committee could require access to the blockchain system to audit it, or a research organization could ask for access to specific information or dataset for analysis purposes that could lead to innovations.

Ideally, a BCDF framework aims to satisfy an essential set of requirements that include the following: (i) secure exchange of forensic data (i.e., ensure storage and communication security), (ii) data integrity (i.e., ensure that the data, in rest or in fly, have not been modified or tampered with), (iii) data provenance (i.e., ensure that there exists a secure chain-of-custody that records all events related to data, starting from its generation to its deletion), (iv) subject/owner consent (i.e., ensure that the data owner, which could be an individual or an entity acting on behalf of several individuals, provides explicit consent or contractual terms for data usage), (v) privacy (i.e., the access of data is restricted to authorized parties only), and (vi) audit trail (i.e., by simply analyzing the chain-of-custody, the auditors can produce a trail of events associated with a particular data object).

Once collected, the secure storage and adequate cross-institutional sharing of forensic records are essential for several reasons, such as investigating the ongoing crime(s) and ensuring security and privacy of the data in a way that it remains admissible in court. Therefore, the right choice of data storage structure (e.g., centralized or distributed) and secure storage techniques (e.g., encryption/decryption and access control structure) depend on the requirements (e.g., S&P, scalability, usability, and compliance management) of the target law system that might vary based on the country.

Figure 2 depicts that the collected forensic data is stored in an offchain storage system, and its corresponding meta-data is stored in the distributed ledger. Offchain data can be characterized as non-transactional data whose efficient storage in the blockchain is difficult due to its large size. The data needs to support update and deletion operations that are not possible if stored on the blockchain (due to its immutability feature). Moreover, there are many reasons, such as confidentiality and scalability, which make a choice and usage of offchain storage solutions one of the main considerations when designing a blockchain-based system. Since the forensic data is too large (e.g., images and videos), it is sensitive. Therefore, it is impractical to store it directly on the blockchain. This is why a majority of existing BBHC solutions have proposed the use of offchain storage solutions for storing medical data, and only the meta-data about the offchain data is added in the distributed ledger [30–32]. The meta-data could hold information such as the hash of the offchain stored data, data ownership information, and a pointer to data location. Specifically, the information stored in the form of meta-data depends on

the target application requirements. For example, suppose data integrity needs to be supported in the system. In that case, the hash value of the offchain data objects should be uploaded to the ledger. Later, when a user accesses the data object, it can compare the hash values to prove that the retrieved object is the same as the one stored initially. Moreover, other S&P measures can be applied to protect the stored data from different types of threats at this stage. For example, instead of sharing the actual data path, the requestor will be provided with a temporary path where the data is made available by copying it from the actual path. In this way, the requestor will not be able to access the future updates (i.e., updates made on the data record after the requestor's recent access period) on the forensic records without transacting a new access transaction on the blockchain. The above process will also help provide a one-way linking between the data requestors and data objects. It will always involve the blockchain network while an entity is trying to access an object. Hence, it will ensure that all events related to data objects are recorded in the ledger. This will further help to perform complex querying and monitoring of the data access-related events. For instance, a data owner can query the blockchain about all the operations performed on its data over a period.

Finally, in any blockchain-based system, the storage and access management of data can be governed by using various functions defined in smart contracts, and these contracts can be securely deployed and executed on the blockchain network nodes. Similarly, the core business logic associated with a target use-case can be implemented using smart contracts. A layer consisting of and governed by smart contract components can provide an interface for facilitating interactions between the blockchain ledger and the front-end application users (e.g., investigators, analysts, regulators, and auditors). All these interactions will be recorded on the immutable ledger to provide a chain-of-custody, which can be used by an institute's information technology (IT) department to monitor security and quality parameters and by regulators and auditors to check if all the actions performed related to forensic data comply with the predefined regulation policies.

3.2 State-of-the-Art Proposals

In this section, we review some important proposals in the direction of blockchain-based forensics proposals. We have further categorized the proposals based on IoT forensics, cloud forensics, and others.

3.2.1 Blockchain-Based IoT Forensics

The increasing number of devices in IoT is also increasing the proliferation of cyber threats. Various intrusion detection systems are configured to detect those insecurities. IoT forensics takes a step ahead of solutions for intrusion detecting and

objectifying the process for investigators to study heterogeneous devices and tracing the evidence.

A blockchain-based investigation framework for IoT is shown in [4]. This considers all communications of IoT devices as transactions in the blockchain. It is divided into three layers: cloud, blockchain, and IoT devices. The devices' transactions help in the strong chain-of-custody (CoC) process. A permissioned blockchain-based IoT forensics framework is shown in [7]. It uses smart-contract based CoC to enhance the transparency and traceability of the evidence. Moreover, it ensures the integrity, authenticity, and non-repudiation properties of the collected evidence. It uses one-time signatures based on the Merkle tree for the preservation of privacy. Another IoT forensic solution specifically for the smart home is researched [33]. It uses a private forensic evidence database, where the captured evidence is stored. It uses a permissioned blockchain, ensuring integrity, authentication, and non-repudiation. This evidence preservation model also works as cyber-attack detection for Small Office Home Office (SOHO) networks. IoT forensic chain (IoTFC) has been shown in [31]. IoTFC ensures traceability and tracks the provenance of evidence items. The blocks are made of the information of evidence identification, preservation, analysis, and presentation. Auditing of the evidence is made trusted due to the use proof of existence. IoTFC also ensures privacy.

A secure blockchain-based Digital Evidence Framework (Block-DEF) has been proposed by the researchers [34]. It uses two stages of evidence storage: original evidence is stored in a secured storage, and evidence or meta-information is stored in the blockchain. A lightweight blockchain is used, and a name-based Practical Byzantine Fault Tolerance (PBFT) consensus is proposed. It uses multi-signature for evidence submission and retrieval. A solution for IoT installations called Probe-IoT uses a public digital ledger [35]. It collects interactions from all the IoT layers and stores them as transactions in public, distributed, and decentralized blockchain networks. In the same direction of work, Forensic Investigation Framework (FIF-IoT) uses a public digital ledger to validate criminal events in an IoT-based environment [36]. The use of blockchain eliminates a centralized control and avoids single-point failure.

Software-defined networks (SDNs) are also enablers for IoTs. A forensic architecture for such an SDN-based IoT is researched in [37]. It uses CoC in the blockchain. Users are validated with Linear Homomorphic Signature (LHS). Voice over Internet Protocol (VoIP), File Transfer Protocol (FTP), and HyperText Transfer Protocol (HTTP), these three types of traffic patterns are considered for transactions. Neuro Multifuzzy classifiers are used to classify maliciousness. IoT functioning also extends to vehicular networks, and therefore, forensics processes are also important in such networks. One such solution for connected vehicles is shown in [1]. It uses a permissioned blockchain framework. Vehicular Public Key Infrastructure (VPKI) provides membership establishment, and privacy is ensured by using pseudonym certificates. It shows a fragmented ledger that stores various information related to vehicles. Autonomous vehicles are the present and future of vehicular networks. Such vehicles are making a way to tamper with the insurance liability model. This problem is addressed by a permission blockchain-based solution [38]. It integrates

the concerned entities in the liability model and provides untampered evidence for liability attribution and formal judgment of any dispute. Another forensic framework for Internet-of-Vehicle (IoV) is studied [2]. The solution, Trust-IoV, provides a secure provenance system for the evidence to provide integrity.

The popularity of IoT infrastructure has given rise to the number of devices connecting to this network technology. We can easily predict that the volume of data will become enormous with more heterogeneity of the devices in the near future. IoT forensics, therefore, becomes a significant concern shortly. Various challenges and opportunities are observed, which shows that IoT forensics is in infancy stage [40–42]. As wireless networks are the core components of IoT and having vulnerabilities of crime activities, network forensics is also integrated with IoT forensics. Such feasibility of methods and opportunities is addressed in [43]. Whatever the methods of forensics are available, one major concern is legal aspects. As the devices or the networks in IoT deal with the users' data privacy, the forensics operations must keep the data privacy intact. Moreover, there are various reasons applicable for which the investigators' privacy or the evidence is required at times. Such legal aspects are discussed in [44]. Recently, authors in [39] propose an IoT forensics system consisting of multiple blockchain networks. It aims to resolve a possible dispute by verifying the authenticity and integrity of data collected from different IoT nodes. The collaboration of multiple blockchains is envisioned to build an improved (i.e., more secure and immutable) but affordable system. The performance analysis on three popular blockchain networks by considering the actual prices shows a significant operational cost reduction. In Table 1, we provide the list of some major state-of-the-art efforts, which uses blockchain-based schemes for addressing various challenges in IoT forensics. The table includes the benefits and the research gaps (derived from their limitations and issues) of the blockchain-based IoT forensics proposals.

3.2.2 Blockchain-Based Cloud Forensics

When we talk about Internet of Things (IoTs), cloud storage also greatly concerns forensic interpolation. The multi-tenant environment and dynamic cloud location also provide hinges back to the forensic applications. Various challenges and opportunities in cloud forensics are explained in [14, 45, 46]. We have reviewed some of the explicit works related to cloud forensics.

Blockchain application in cloud storage forensics is shown in [9]. The authors discuss research done in cloud storage forensics and how it contrasts with distributed blockchain-based cloud forensics. Specifically, the authors shed light on how blockchain technology is integrated and implemented with cloud forensics and what possible benefits and challenges it brings to forensic investigators. Some potential research directions to address the identified challenges are also presented. STORJ is used for the discussion as a technology example. Evidence reconstruction is not addressed and can be explored further. Jurisdictional borders are posing severe challenges on cloud forensics due to their distributed nature and require multiparty

Table 1 Benefits and challenges in IoT forensics proposals

Reference	Benefits	Research gap
Ryu et al. [4]	Strong CoC as all the transactions are in blockchain; integrity and transparency for all stakeholders of devices	Forensic processes are not included efficiently; privacy is not addressed
Le et al. [7]	Privacy, traceability, reporting of attack	Reliability and scalability with heterogeneous devices
Brotsis et al. [33]	Cyber-Trust platform relies on advanced intrusion detection tools, CoC process	Scalability, heterogeneity of the devices not supported well
Li et al. [31]	Data provenance architecture and assurance of examination operations, CoC, privacy	Validation incomplete, performance need to be measured on heterogeneity of devices
Tiana et al. [34]	Lightweight, PBFT consensus	The mapping between the storage and evidence information needs to be secure
Hossain et al. [35]	It ensures integrity, confidentiality, anonymity, and non-repudiation of the evidence	Analysis process and CoC are not addressed
Hossain et al. [36]	Collection of evidence and tamper-resistant evidence storage	Evidence processing and CoC are not addressed
Pourvahab and Ekbatanifard [37]	CoC, SDN-based IoT	Load balancing, scalability, applicable only for CoC
Cebe et al. [1]	It helps in trustless, traceable, and privacy-aware post-accident analysis, minimal storage, and processing overhead	Fragments mapping, regulation enforcement, availability of forensic critical data
Oham et al. [38]	Untampered evidence for liability attribution	The connection between transactions of entities and liability, accidental scenarios need to be studied
Hossain et al. [2]	Secure evidence provenance system	CoC is not considered and therefore manipulation with evidences can be exploited
Mercan et. al. [39]	Low cost evidence verification system	CoC is not considered, not evaluated against any threat model

cooperation. Besides, trustworthiness is missing in the interaction of records for different stakeholders in clouds. To handle such issues, a process of provenance has been proposed in [8]. It uses proof of existence and privacy preservation. Moreover, group signatures are also used to provide anonymity. Some other approaches of securing audit trails in cloud forensics are mentioned in [10]. Some possible solutions that it encompasses include the use of distributed ledger with a cloud service provider, Hashgraph, and offchain state agreement solutions.

A recent development for cloud forensics is shown in [47]. It uses a multi-granularity privacy leakage forensics method. It combines continuous RAM mirroring technology and dynamic taint analysis to assist the forensics investigation.

This solution suffers from a lack of attribution on the malicious features. Forensic aware secure logging as a service is researched in [48]. It handles multi-stakeholder collusion problems. The intrinsic immutability of blockchain ensures the integrity of the logs. Confidentiality and access control are provided by this solution. However, the auditability increases by this approach, the service providers' indulgement cannot be ignored completely.

3.2.3 Other Blockchain-Based Forensics Proposals

Digital evidence related to videos and photos is crucial in the investigation process and is also prone to manipulation. Therefore, such evidence must be appropriately preserved. This section discusses some of the significant contributions for multimedia forensics, preservation of evidence, and other applications.

While considering the video as evidence, the source identification and its attribution are of great importance. These videos also suffer from blurring due to packet loss in a wireless environment. A significant solution to handle the mentioned problems has been shown in [49]. It uses wireless channel signatures and selective frame processing into source identification to detect the wireless camera spoofing attack. Another video-based forensic approach is shown in [50]. It uses two phases: offline and online. During the offline phase, raw video collection is executed. These are inserted into the ingestion service. These services run in parallel to provide various video analytics tasks. Visual index and Evidence Graph (EG) are generated. The Complex Query

Formulator (CQF) is used for multiple searches during the online phase. Additionally, the inference and recommendation modules use the knowledge accumulated in the EG. A distributed ledger-based CCTV forensics is shown in [51]. Block creation is configured on the cameras themselves. Video segments are input to blocks via the embedding of a watermark into the video stream. Metadata is used for cross-reference of the video segment. Previously, a solution for video forensics for investigating child pornography has been shown in [52]. It is mostly focused on peer identification and reducing or disallowing the peers for sharing such offensive content. A blockchain-based approach for photo forensics is researched in [13]. It helps in solving the problems of photo-faking, photo-tracing problems, and copyright disputes. It combines blockchain with image authentication technologies. A new mining protocol is used for verification. It uses a user-defined access control mechanism.

Mobile devices are essential for evidence collection. The data storage, network connection, hardware identity, and other such information are very helpful for the investigators to initiate a direction of queries. However, mobile forensics must be transparent and distributed as every minute detail of such devices can lead to some critical evidence. Open research possibilities are existing in this domain as shown in [15] and [53]. Whatever the sources of evidence are, they must be properly preserved along with the accountability of chain-of-custody. In this sense, preservation of the evidence is also important as any dispute can invalidate them.

Such preservation system of Bitcoin is shown in [54]. It uses lightweight digital evidence-preservation architecture with proof of existence. Evidence chain interacts with the original blockchain. The extended chains lie on the top of the main chain. Smartphone-based evidence preservation and its integrity verification have been researched [55]. It uses the spatio-temporal properties of digital evidence. Public blockchain uses ledger entries for evidence file's hash and location certificate. Anonymity is ensured for the human witness. Blockchain is queried for integrity checking of the required evidence. Electronic Evidence Preservation Model (EEPM) uses the blockchain [12]. It is segregated into four-part data: complete evidence information, user data information, platform business data, and log data. Blockchain uses the hashed values of the evidence, and original information are stored centrally for security. The integrity verification for blockchain-based preservation models has been explained in [56]. It uses verification of time-based existence of evidence with Open Time Stamps (OTS) service. Another recent evidence management approach with consortium blockchain is shown in [30]. It monitors the evidence flow along with the prosecution trials. The approach uses short randomizable signatures and ciphertext-policy attribute-based encryption for access control. Secure voting is a designed method to protect privacy. Blockchain-based Framework for auto-Insurance Claims and Adjudication (B-FICA) is shown in [38]. It uses permissioned blockchain with two partitions. Multi-signed transactions use proof-of-execution of instructions. The consensus is lightweight, and validation protocol prevents evidence modification.

Some of the other proposals for different aspects of digital evidence and their varsity of the sources are also important to consider. For example, log files are an excellent source of evidence in digital forensics. Therefore, their security is also a concern. Toward this direction, a blockchain-based solution has been researched in [57]. Encryption techniques are used for the information before sending to the blockchain. Scientific reasoning and its confidence level must be high while presenting evidence in front of the juries. A blockchain-based solution for this direction is shown in [58]. It provides a digital evidence chain, the Digital Evidence Inventory (DEI), confidence level rating with Forensics Confidence Rating (FCR) structure, and time-based exploration of evidence using a Global Digital Timeline (GDT). The evidence lifecycle passes through various intermediate stages, and different personnel acquire them at different timestamps for examination. These transactions need to be properly accounted for to avoid any dispute. Thus, chain-of-custody (CoC) is very important and needs to be transparent. Blockchain's transparency of transactions assists CoC in this regard. For example, blockchain-based chain-of-custody (B-CoC) ensures auditable integrity of the collected evidence and traceability of owners [59]. Another example, called forensic chain, uses five components: participants, front-end, core modules, blockchain network, and distributed storage [11].

Recently, authors in [30] propose *LEChain*, a framework for digital evidence management based on blockchain technology. LEChain supports the supervision of the entire evidence lifecycle, i.e., from evidence collection and access for the period of investigation to jury voting in the court trials. It also manages all

the court data (e.g., votes and trial outcomes). To ensure security and privacy preserving the execution of evidence flow, it proposes various techniques. For instance, (i) to protect the witness privacy, it uses short randomizable signatures to anonymously authenticate witnesses' identities; (ii) to control access to evidence, it uses ciphertext-policy attribute-based encryption; and (iii) to protect juror privacy, it designs a secure voting approach. Also, it builds a consortium blockchain to record evidence transactions. Finally, a formal analysis of the security and privacy of LEChain is presented, and it is evaluated for computational costs and communication overhead by implementing a prototype based on a local Ethereum test network. Finally, we use Table 2 to summarize all the above discussed blockchain-based solutions that have been proposed in the literature to address different challenges in various application domains. For each of these proposals, the table includes the application domain in which it has been applied, the benefits that was targeted, and the limitations of these proposals.

4 Discussions and Future Research Directions

This section discusses the key limitations while using blockchain for digital forensic management and possible strategies to address them. Also, we present an analysis of future research challenges of digital forensics.

There are two design choices related to the use of blockchain platforms (i.e., public or private). Depending upon the type, each type has its set of challenges that vary concerning the target application scenario. For instance, public blockchains face limitations such as scalability, performance, and cost issues. In this regard, public blockchains are mainly used for cryptocurrencies and to commit small pieces of data (i.e., hashes) for verifiability purposes, while in private blockchains, the performance and scalability challenges are overcome due to more efficient consensus mechanisms and a reduced number of participants. Nevertheless, both public and private blockchains require proper data management and architectural designs to provide security and privacy guarantees.

With blockchain usage, one of the biggest and most-heated debates that persist even today is the issue of scalability with which data storage is strongly related. The storage techniques used in a blockchain-based system can be classified as onchain or offchain. Both have their pros and cons. Depending upon the type of storage solution integrated into the system, several system performance parameters will be affected, including data availability, security and privacy, and access latency. Since various entities exist at different layers (e.g., application, network, and device layers) in the BCDF system, integration of different technologies might be required. In such a system, designing an architecture that supports better modularity will improve its overall productivity. Various blockchain platforms that provide support to build such modular architectures make them a more practical design choice for BCDF solutions.

Table 2 Benefits and challenges in other proposals

Reference	Application	Benefits	Research gap
Chen et al. [49]	Video	Real-time video forensics, prevention of video spoofing	Not transparent and distributed, prone to influence-based attacks
Axenopoulos et al. [50]	Video	Accuracy of object and event identification, interference, and recommendation module	Coordination between offline and online phases, prone to manipulation in offline phase
Kerr et al. [51]	Video	Trustworthiness of video evidence protection through watermarks	The validation of watermarks is questionable as watermarks can be cloned, central point failure of integrity
Hurley et al. [52]	Video	Peer detection and disallow	Online and offline peer behavior mapping
Zou et al. [13]	Photo	Prevention of photo faking and authentication, user-defined access control	Performance and attack analysis need to be explored
Wang et al. [54]	Evidence preservation	Audit transparency, anonymity, scalability, less complex	The connection of original chain and extended chain need to be optimized
Samanta et al. [55]	Evidence preservation	Integrity verification of the evidences, anonymity of the human witness	Application specific, location attestation can be manipulated with fake coordinates if not properly managed
Xiong et al. [12]	Evidence preservation	Hashed information checks integrity	Unreliability for hash-based attacks, non-validated
Weilbach et al. [56]	Evidence integrity	Reliability	Timestamp accuracy
Li et al. [30]	Evidence management	Privacy, access control, complete monitoring of evidence transactions	Data leakages attacks are not considered
Oham et al. [38]	Auto-insurance	Need-to-know basis information sharing, auditability, and reliability	Validation of attestation protocols for correctness, reliability, and integrity of data submitted by vehicles
Louis et al. [57]	Log files	Confidentiality of proprietary information of organizations	Single user control, privacy need to be addressed
David Billard [58]	Evidence management and processing	Digital evidence chain, confidence level for scientific reasoning, time-based audits of evidences	The cooperation and coordination among these three prime modules of operations
Bonomi et al. [59]	CoC	Auditability, integrity of evidences, and traceability of owners	Validators' validation is incomplete
Lone et al. [11]	CoC	Integrity and tamper resistance of CoC	Optimization required for end-to-end services
Meng et al. [30]	CoC	Fine-grained access control, witness, and jury identity privacy	Data leakage attacks from internal adversaries

Next, we present next-generation digital forensics research challenges, which are as follows.

- *Evidence tokenization:* During the investigation of a crime scene, various pieces of evidence are analyzed by multiple law enforcement persons for knowledge extraction that could help understand the events of the crime under investigation. It is also possible that multiple peoples process the same evidence to extract different knowledge about the case. For example, a laptop involved in a crime must be analyzed by different members, where one member may study the content available on the laptop. At the same time, another may process its hard-drive or file system to recover deleted files, and a third one might check its physical details. Therefore, it is required to divide such evidence into several artifacts to enable parallelism. Here, the challenge is how to store these artifacts in the blockchain. One existing solution is to use tokens. Therefore, a significant challenge is to design a mechanism that enables the storage of tokenized artifacts in the blockchain during the period of a digital forensic investigation.
- *Large-scale data management:* As mentioned before digital forensics is data-driven domain because for each case there might be a need to collect and store a large volume of data (e.g., log files, images, and videos). Moreover, the requirement of storing the raw data forces us to store the whole data without any modifications that can be done to reduce its size. Therefore, designing a system with an adequate offchain storage solution (e.g., IPFS, Storj) ensures features such as security, privacy, availability, and scalability remain a challenge.
- *Support for complex querying and understandable forensic outcomes:* The users should be able to query the blockchain system for specific tasks. For this purpose, the blockchain system needs to integrate a querying engine that interacts with the blockchain to provide replies. Moreover, the query outcome should be simple to understand for the user. Furthermore, the system should support automation, and the reports/outcomes generated should be admissible and understandable in court trials. Thus, efforts are required to provide a link between forensic sound procedures and their understandable explanation.
- *Interoperability and cross-border jurisdictions:* The use of international standardized flows and proper data management and sharing agreements will enhance the fight of cybercrime. Nowadays, international collaborations already exist between governments of different countries. Nevertheless, further development of blockchain-based solutions will serve as a ground truth platform for standardized solutions, enabling global interoperability. Moreover, since multiple institutions and heterogeneous devices exist from which evidence is collected and generated during a crime investigation, this data may be of a different format. Therefore, solutions are required that ensures that the data is submitted/inserted to/in the blockchain in a homogeneous format that is understandable by all its users. One possible way to support data interoperability is to use smart contracts. Suppose a contract checks the input data against a predefined data structure when it receives a request from an external entity to

store or update forensic data. In that case, it can be used to enforce a predefined data entry format. Such a contract will ensure that the data models have a homogeneous data storage and access structure across all the institutions that generate data. This homogeneity will make access to different data storage systems interoperable, and it will allow efficient data sharing across multiple institution systems.

5 Conclusions

In this chapter, we investigated research challenges in the management of digital forensics investigation systems. To address the identified challenges, we explore the possible usage of blockchain technology along with smart contracts. We provide a list of benefits that these new technologies can provide in the digital forensics domain. Based on the state-of-the-art efforts studied, it is safe to say that the existing blockchain-based solutions for digital forensics are at their early stages. Specifically, no complete solution covering most of its challenges and tested in a real-world environment or trails exists. We also present designing a generic blockchain-based digital forensics architecture and discuss its methodology along with the directions that could address some of the existing challenges in this domain. The aim of the framework is to show how blockchain and smart contract technologies can be integrated in the present digital forensics system to improve it. Finally, we also conclude that there are various challenges in adopting blockchain technology in this domain, and the same should be researched to check its feasibility for digital forensic applications.

Acknowledgments This work is partially supported in part by EU LOCARD Project under Grant H2020-SU-SEC-2018-832735.

References

1. M. Cebe, E. Erdin, K. Akkaya, H. Aksu, S. Uluagac, Block4Forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles. IEEE Commun. Mag. **56**(10), 50–57 (2018)
2. M. Hossain, R. Hasan, S. Zawoad, Trust-IoV: a trustworthy forensic investigation framework for the internet of vehicles (IoV), in *2017 IEEE International Congress on Internet of Things (ICIOT)* (2017), pp. 25–32
3. C. Oham, S.S. Kanhere, R. Jurdak, S.K. Jha, A blockchain based liability attribution framework for autonomous vehicles, in *CoRR*, vol. abs/1802.05050 (2018). http://arxiv.org/abs/1802.05050
4. J.H. Ryu, P.K. Sharma, J.H. Jo, J.H. Park, A blockchain-based decentralized efficient investigation framework for IoT digital forensics. J. Supercomput. **75**, 4372–4387 (2019)
5. S. Rathore, B.W. Kwon, J.H. Park, Blockseciotnet: Blockchain-based decentralized security architecture for iot network. J. Netw. Comput. Appl. **143**, 167–177 (2019)

6. I. Kotsiuba, A. Velykzhanin, O. Biloborodov, I. Skarga-Bandurova, T. Biloborodova, Y. Yanovich, V. Zhygulin, Blockchain evolution: from bitcoin to forensic in smart grids, in *2018 IEEE International Conference on Big Data (Big Data)* (2018), pp. 3100–3106
7. D. Le, H. Meng, L. Su, S.L. Yeo, V. Thing, Biff: A blockchain-based IoT forensics framework with identity privacy, in *TENCON 2018–2018 IEEE Region 10 Conference* (2018), pp. 2372–2377
8. Y. Zhang, S. Wu, B. Jin, J. Du, A blockchain-based process provenance for cloud forensics, in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)* (2017), pp. 2470–2473
9. J. Ricci, I. Baggili, F. Breitinger, Blockchain-based distributed cloud storage digital forensics: Where's the beef?. IEEE Secur. Priv. **17**(1), 34–42 (2019)
10. M. Westerlund, M. Neovius, G. Pulkkis, Providing tamper-resistant audit trails for cloud forensics with distributed ledger based solutions, in *International Journal On Advances in Security*, vol. 11 (2018)
11. A.H. Lone, R.N. Mir, Forensic-chain: Blockchain based digital forensics chain of custody with poc in hyperledger composer. Digit. Investig. **28**, 44–55 (2019)
12. Y. Xiong, J. Du, Electronic evidence preservation model based on blockchain, in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, ser. ICCSP '19. (ACM, New York, 2019), pp. 1–5. http://doi.acm.org/10.1145/3309074.3309075
13. R. Zou, X. Lv, B. Wang, Blockchain-based photo forensics with permissible transformations. Comput. Secur. **87**, 101567 (2019)
14. M. Bharat, S. Gaurav, C. Raymond, C. Mauro, G. Manoj, A systematic survey on cloud forensics challenges, solutions, and future directions, in *ACM Computing Survey* (2019), pp. 38
15. B. Konstantia, C. Tiago, M. Edmundo, S. Paulo, Current and future trends in mobile device forensics: a survey, in *ACM Computing Survey* (2018), pp. 31
16. M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, E.K. Markakis, A Survey on the Internet of Things (IoT) Forensics: challenges, approaches, and open issues, in *IEEE Communications Surveys Tutorials* (2020), pp. 1191–1221
17. J. Hou, Y. Li, J. Yu, W. Shi, A survey on digital forensics in Internet of Things, in *IEEE Internet of Things Journal* (2020), pp. 1–15
18. T. Mehrdad, H. Elaheh, S. Sajjad, *Forensically Ready Digital Identity Management Systems, Issues of Digital Identity Life Cycle and Context of Usage* (Inderscience Publishers, New York, 2017)
19. R.P. Ayers, S. Brothers, W. Jansen, Guidelines on mobile device forensics, in *Special Publication (NIST SP) - 800-101 Rev 1* (2014). https://doi.org/10.6028/NIST.SP.800-101r1
20. ENISA, Mobile threats incident handling (Part II). Accessed: 05/06/2019 (2015).
21. Infosec Institute, Guidelines on mobile device forensics. Accessed: 05/06/2019 (2015). https://resources.infosecinstitute.com/category/computerforensics/introduction/mobile-forensics/the-mobile-forensics-process-steps-types
22. M. Chernyshev, S. Zeadally, Z. Baig, A. Woodward, Mobile forensics: advances, challenges, and research opportunities. IEEE Secur. Priv. **15**(6), 42–51 (2017)
23. K. Barmpatsalou, T. Cruz, E. Monteiro, P. Simoes, Current and future trends in mobile device forensics: A survey. ACM Comput. Surv. **51**(3), 46:1–46:31 (2018)
24. V. Buterin, *White Paper, A Next-Generation Smart Contract and Decentralized Application Platform* (2014), pp. 1–36. https://cryptorating.eu/whitepapers/Ethereum/Ethereumwhitepaper.pdf
25. D. Macrinici, C. Cartofeanu, S. Gao, Smart contract applications within blockchain technology: a systematic mapping study, in *Telematics and Informatics* (2018), pp. 2337–2354
26. D. Shakhbulatov, J. Medina Chirinos, Z. Dong, R. Rojas-Cessa, IEEE open journal of the computer society, in *How Blockchain Enhances Supply Chain Management: A Survey* (2020)
27. M. Kuperberg, Blockchain-based identity management: a survey from the enterprise and ecosystem perspective. IEEE Trans. Eng. Manag. **67**(4), 1–20 (2019)

28. Y. Hanada, L. Hsiao, P. Levis, Smart contracts for machine-to-machine communication: possibilities and limitations, in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)* (2018), pp. 130–136
29. D. Chris, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners* (2017)
30. M. Li, C. Lal, M. Conti, D. Hu, LEChain: A blockchain-based lawful evidence management scheme for digital forensics. Futur. Gener. Comput. Syst. **115**, 406–420 (2021)
31. S. Li, T. Qin, G. Min, Blockchain-based digital forensics investigation framework in the internet of things and social systems, in *IEEE Transactions on Computational Social Systems*, vol. 6(6) (2019), pp. 1433–1441. https://doi.org/10.1109/TCSS.2019.2927431
32. D. Thomas, C. Fran, P. Constantinos, *SoK: Blockchain Solutions for Forensics* (2020), pp. 1–16
33. S. Brotsis et al., Blockchain solutions for Forensic Evidence preservation in IoT environments, in *2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France* (2019), pp. 110–114. https://doi.org/10.1109/NETSOFT.2019.8806675
34. Z. Tiana, M. Lia, MeikangQiub,_, YanbinSuna,_, Shen Sua, Block-DEF: a secure digital evidence framework using blockchain, in *Procedia Computer Science* (2019), pp 1–16
35. M. Hossain, R. Hasan, S. Zawoad, Probe-IoT: A public digital ledger based forensic investigation framework for IoT, in *IEEE INFOCOM 2018—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI* (2018), pp. 1–2. https://doi.org/10.1109/INFCOMW.2018.8406875
36. M. Hossain, Y. Karim, R. Hasan, FIF-IoT: A forensic investigation framework for IoT using a public digital ledger, in *Proceedings of the 2018 IEEE International Congress on Internet of Things (ICIOT) (2018)*. Part 2018 IEEE World Congress Service (2018), pp. 33–40
37. M. Pourvahab, G. Ekbatanifard, An efficient forensics architecture in software-defined networking-IoT using blockchain technology, in *IEEE Access*, vol. 7 (2019), pp. 99573–99588. https://doi.org/10.1109/ACCESS.2019.2930345
38. C. Oham, R. Jurdak, S.S. Kanhere, A. Dorri, S. Jha, B-FICA: BlockChain based Framework for Auto-Insurance Claim and Adjudication, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada* (2018), pp. 1171–1180. https://doi.org/10.1109/Cybermatics_2018.2018.00210
39. S. Mercan, M. Cebe, E. Tekiner, K. Akkaya, M. Chang, S. Uluagac, A Cost-efficient IoT Forensics Framework with Blockchain, in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada* (2020), pp. 1–5
40. F. Servida, E. Casey, IoT forensic challenges and opportunities for digital traces. Digit. Investig. **28**(Supplement), S22–S29 (2019)
41. M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, E.K. Markakis, A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues. IEEE Commun. Surv. Tutorials **22**(2), 1191–1221 (2020). https://doi.org/10.1109/COMST.2019.2962586
42. J. Hou, Y. Li, J. Yu, W. Shi, A survey on digital forensics in Internet of Things. IEEE Internet Things J. **7**(1), 1–15 (2020). https://doi.org/10.1109/JIOT.2019.2940713
43. S. Khan, A. Gani, A.W.A. Wahab, M. Shiraz, I. Ahmad, Network forensics: review, taxonomy, and open challenges. J. Netw. Comput. Appl. **66**, 214–235 (2016)
44. M.M. Losavio, K.P. Chow, A. Koltay, J. James, The Internet of Things and the Smart City: Legal challenges with digital forensics, privacy, and security. Secur. Priv. **1**, e23 (2018). https://doi.org/10.1002/spy2.23
45. S. Khan, A. Gani, A.W.A. Wahab, M.A. Bagiwa, M. Shiraz, S.U. Khan, R. Buyya, A.Y. Zomaya, Cloud log forensics: foundations, state of the art, and future directions. ACM Comput. Surv. **49**(1), Article 7, 42 pages (2016). https://doi.org/10.1145/2906149
46. A. Pichan, M. Lazarescu, S.T. Soh, Cloud forensics: technical challenges, solutions and comparative analysis. Digit Investig **13**, 38–57 (2015)
47. D. Zou, et al., A multigranularity forensics and analysis method on privacy leakage in cloud environment. IEEE Internet Things J. **6**(2), 1484–1494 (2019). https://doi.org/10.1109/JIOT.2018.2838569

48. S. Rane, A. Dixit. BlockSLaaS: blockchain assisted secure logging-as-a-service for cloud forensics, in *Security and Privacy. ISEA-ISAP 2019*, ed. by S. Nandi, D. Jinwala, V. Singh, V. Laxmi, M. Gaur, P. Faruki. Communications in Computer and Information Science, vol. 939 (Springer, Singapore, 2019). https://doi.org/10.1007/978-981-13-7561-3_6
49. S. Chen, A. Pande, K. Zeng, P. Mohapatra, Live video forensics: source identification in lossy wireless networks, in *IEEE Transactions on Information Forensics and Security*, vol. 10(1) (2015), pp. 28–39. https://doi.org/10.1109/TIFS.2014.2362848
50. A. Axenopoulos, V. Eiselein, A. Penta, E. Koblents, E. La Mattina, P. Daras, A framework for large-scale analysis of video ïn the Wildẗo assist digital forensic examination, in *IEEE Security & Privacy*, vol. 17(1) (2019), pp. 23–33. https://doi.org/10.1109/MSEC.2018.2875851
51. M. Kerr, F. Han, R. v. Schyndel, A blockchain implementation for the cataloguing of CCTV video evidence, in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand* (2018), pp. 1–6. https://doi.org/10.1109/AVSS.2018.8639440
52. R. Hurley, S. Prusty, H. Soroush, R.J. Walls, J. Albrecht, E. Cecchet, B.N. Levine, M. Liberatore, B. Lynn, J. Wolak, *Measurement and Analysis of Child Pornography Trafficking on P2P Networks International World Wide Web Conference Committee (IW3C2)*. WWW 2013 (2013), Rio de Janeiro, Brazil
53. M. Chernyshev, S. Zeadally, Z. Baig, A. Woodward, Mobile forensics: advances, challenges, and research opportunities. IEEE Secur. Priv. **15**(6), 42–51 (2017). https://doi.org/10.1109/MSP.2017.4251107
54. M. Wang, Q. Wu, B. Qin, et al., Lightweight and manageable digital evidence preservation system on Bitcoin. J. Comput. Sci. Technol. **33**, 568–586 (2018). https://doi.org/10.1007/s11390-018-1841-4
55. P. Samanta, S. Jain, E-Witness: preserve and prove forensic soundness of digital evidence, in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)* (Association for Computing Machinery, New York, 2018), pp. 832–834. https://doi.org/10.1145/3241539.3267720
56. W.T. Weilbach, Y.M. Motara, Applying distributed ledger technology to digital evidence integrity. SAIEE Africa Research Journal **110**(2), 77–93 (2019). https://doi.org/10.23919/SAIEE.2019.8732798
57. L.M. Shekhtman, E. Waisbard, Securing log files through blockchain technology, in *Proceedings of the 11th ACM International Systems and Storage Conference (SYSTOR '18)* (Association for Computing Machinery, New York, 2018), p. 131. https://doi.org/10.1145/3211890.3211921
58. D. Billard, Weighted forensics evidence using blockchain, in *Proceedings of the 2018 International Conference on Computing and Data Engineering (ICCDE 2018)* (Association for Computing Machinery, New York, 2018), pp. 57–61. https://doi.org/10.1145/3219788.3219792
59. S. Bonomi, M. Casini, C. Ciccotelli, B-CoC: A blockchain-based chain of custody for evidences management in digital forensics. Tokenomics (2019)

Part IV
Applications

Supply Chain Management Using Blockchain

Christopher Klinkmueller, H. M. N. Dilum Bandara, Xiwei Xu, and Qinghua Lu

1 Introduction

Over the last century, industrial production has seen an enormous shift in how goods are created. Instead of sourcing, manufacturing, and selling materials and goods locally or regionally, organizations have started to focus on core competencies and have increasingly participated in global markets. Consequently, in many industries, such as manufacturing, retail, agriculture, and pharmaceutics, goods and services are not delivered by individual organizations anymore. Instead, organizations collaborate in *supply chains* to collectively produce and distribute goods. To this end, organizations that participate in a supply chain establish complex, multi-party arrangements in which organizations take over different responsibilities and accountabilities. Additionally, organizations need to deliver specified service levels while satisfying regulations and logistical constraints. The complexity of these arrangements further increases in international supply chains, as different legislations in different jurisdictions need to be accounted for. Moreover, to ensure that these arrangements are implemented and that goods are produced and distributed to the right participant at the right time and costs in the correct quality and quantity, organizations need to communicate and exchange required information. For example, carriers need to know when to pick up goods, manufacturers require information to schedule their production, temperature-sensitive goods need to be

C. Klinkmueller
BPMotion, Sydney, Australia
e-mail: christopher.klinkmueller@bpmotion.com.au

H. M. N. Dilum Bandara · X. Xu (✉) · Q. Lu
CSIRO Data61, Eveleigh, NSW, Australia
e-mail: HMN.DilumBandara@data61.csiro.au; Xiwei.Xu@data61.csiro.au; Qinghua.Lu@data61.csiro.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_12

monitored throughout the supply chain, and customs inspections are informed by shipment documents.

In recent years, blockchains have been proposed as a highly promising technology to address challenges arising from the complexity of supply chain arrangements and the associated need for information exchange. That is because, when used as a communication and integration platform, the flexible structure of nodes and smart contracts in a blockchain network can accommodate the complex, dynamic structure of business relationships and operations in a supply chain. Further, blockchains provide a means to drive supply chain transformation by enhancing trust, improving efficiency, and enabling regulatory compliance [1].

Data management and *business process management* (BPM) are believed to benefit from the blockchain technology [2]. On the one hand, blockchain technology ensures data immutability, consistency, and availability. These core features support the enhancement of supply chain visibility and traceability; management of origin and ownership data; as well as auditing processes, fairness, transparency, and trust. For instance, confidence in supply-chain-related documents and certificates could expedite customs and biosecurity processes, could reduce risk and insurance costs, and could be used as leverage in trade finance. The timely flow of goods and information could enable real-time payments between parties along many points in the supply chain. Moreover, applied cryptography enables participating parties to issue and digitally verify certificates related to goods and qualifications.

On the other hand, by offering the opportunity to automate business processes, blockchains can contribute to efforts regarding the reduction of errors, communication complexity, and risks for cyber-attacks. This way, blockchains can ultimately improve supply chain governance, coordination, and contract management. That is, blockchain technology provides a level playing field for supply chain participants due to the disintermediation and supports equal rights for all participants. Due to open real-time sharing of business data, automatic payment processing with real-time regulatory enforcement, and reduced regulatory burden, supply chain efficiency can be significantly improved. Moreover, blockchains could ease monitoring and auditing of supply chain activities, while enforcing compliance on the fly for regulatory bodies and businesses. That is because essential business data are securely and reliably stored on the blockchain as permanent evidence and they are manipulated only through smart contracts.

With that in mind, it is not surprising that the use of blockchains for supply chain management is an extremely active area of innovation and technology development where many organizations have started to adopt blockchains. For example, Everledger[1] uses blockchain technology to create a secure digital record for diamonds and colored gemstones, where their origin, characteristics, and ownership are stored. Walmart has tested blockchain technology for its supply chain management in a pilot project on tracking pork in the USA and China using Hyperledger Fabric [3]. Similarly, Maersk—the globally operating shipping

[1] https://www.everledger.io/.

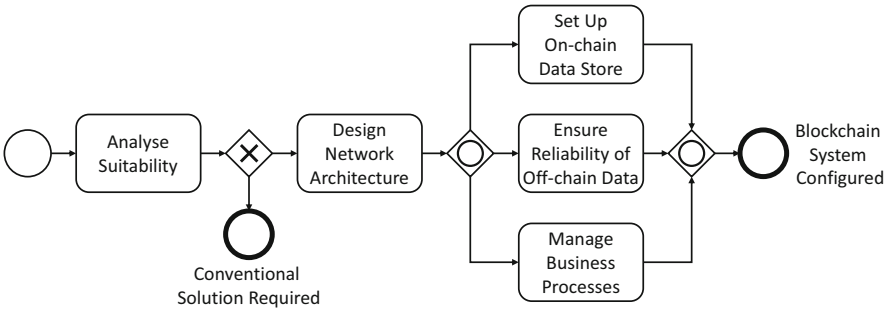


Fig. 1 Key steps for developing a blockchain-based supply chain management solution

company—in collaboration with IBM did a proof of concept for the tracking of containers on their journey from Africa to Europe [4]. Provenance[2]—a platform for creating transparency of products—conducted a pilot project in which fish caught in Indonesia was tracked on the blockchain [5]. Lockheed Martin is working on a blockchain-based solution for ensuring data integrity along its development processes to minimize security risks [6]. Lastly, modum[3] uses blockchain technology to monitor and control the safe delivery of pharmaceutical drugs [7]. Overviews of blockchain projects and initiatives in the supply chain industry are provided in [8–10].

Considering the interest blockchain technology has gained in the supply chain management community, in this chapter, we provide an introduction to approaches and solutions that enable organizations to set up and maintain blockchain-based solutions. In this regard, we focus on the key steps necessary to develop and configure the blockchain components. Figure 1 outlines the idealized development process and the activities that we discuss in this chapter.

First, a suitability analysis needs to be conducted to determine whether a blockchain-based solution is the right fit for the respective supply chain context. We discuss the fundamental questions that such a suitability analysis needs to answer in the context of supply chain management in Sect. 2. Should a blockchain solution be required, the next step is to design the network architecture. We outline basic architecture options in Sect. 3. Once the network architecture was decided and set up, applications on top of this architecture can be developed. In this regard, we focus on applications that support the two key areas that blockchain is considered to enhance in supply chain management [2]: data management and BPM. Regarding the former, we introduce an approach to establishing an on-chain data store in Sect. 4. While blockchain technology offers data immutability, integrity, and availability for data managed on-chain, blockchain-based solutions typically send and receive data to and from off-chain components. Therefore, we also elaborate

[2] https://www.provenance.org/.

[3] http://modum.io.

on mechanisms to ensure the reliability of off-chain data in Sect. 5. With respect to BPM, we summarize techniques for enforcing, implementing, monitoring, and analyzing business processes in Sect. 6.

2 Blockchain Suitability Analysis

Blockchain technology does not constitute a universal solution whose application makes sense in every supply chain setting. By contrast, blockchain's specific properties render it useful in some contexts and limit its applicability in others. Before designing and implementing a blockchain-based system, it is necessary to assess blockchain's suitability against the characteristics of the use case that it is intended to be used for. With that in mind, we present a guideline for supply chain practitioners to perform such an evaluation and to determine whether the blockchain fits a specific use case or not. Figure 2 shows the suitability framework from [11] that could guide the analysts through a series of six questions and assists them in deciding whether a conventional *information system* (IS) or a blockchain-based solution should be preferred. We outline each of the questions in more detail below.

Is Multi-Party Required? The blockchain can be viewed as a mechanism for integrating individual stakeholders. Supply chains are highly complex networks in which many organizations jointly realize a service. That is, supply chains involve a broad range of participants, such as producers, factories, transport providers, wholesalers, distributors, retailers, and consumers. Moreover, in international supply chains, stakeholders additionally include authorities like customs and biosecurity. Hence, as long as the supply chain scenario's scope is on multi-party integration, the answer to this question is *Yes*, and blockchains are in principle applicable to the scenario.

Is a Trusted Party Required? One of the key benefits of blockchain technology is that it provides a reliable, transparent, and immutable platform for storing data and executing custom source code while being operated by a network of independent hosts. Thus, in situations where supply chain functionality can be automated, blockchain technology can take over activities that are otherwise performed by trusted parties. This, for example, can include activities that are typically performed by a supply chain coordinator like the automated recording or transmission of data, the implementation of transparent auctions or marketplaces that facilitate the trading of goods and materials, or the coordination of activities in the supply chain. If it is conceivable to transfer such tasks from a trusted party to a blockchain solution, the answer is *No*. However, in some scenarios, a trusted party cannot be replaced, for example, aspects of coordination which require a trusted party. Moreover, in a port, customs or biosecurity authorities are regulatory bodies that have to release containers for import based on an assessment of relevant shipping documents and the containers. For legal reasons, the coordination of the assessment process, as

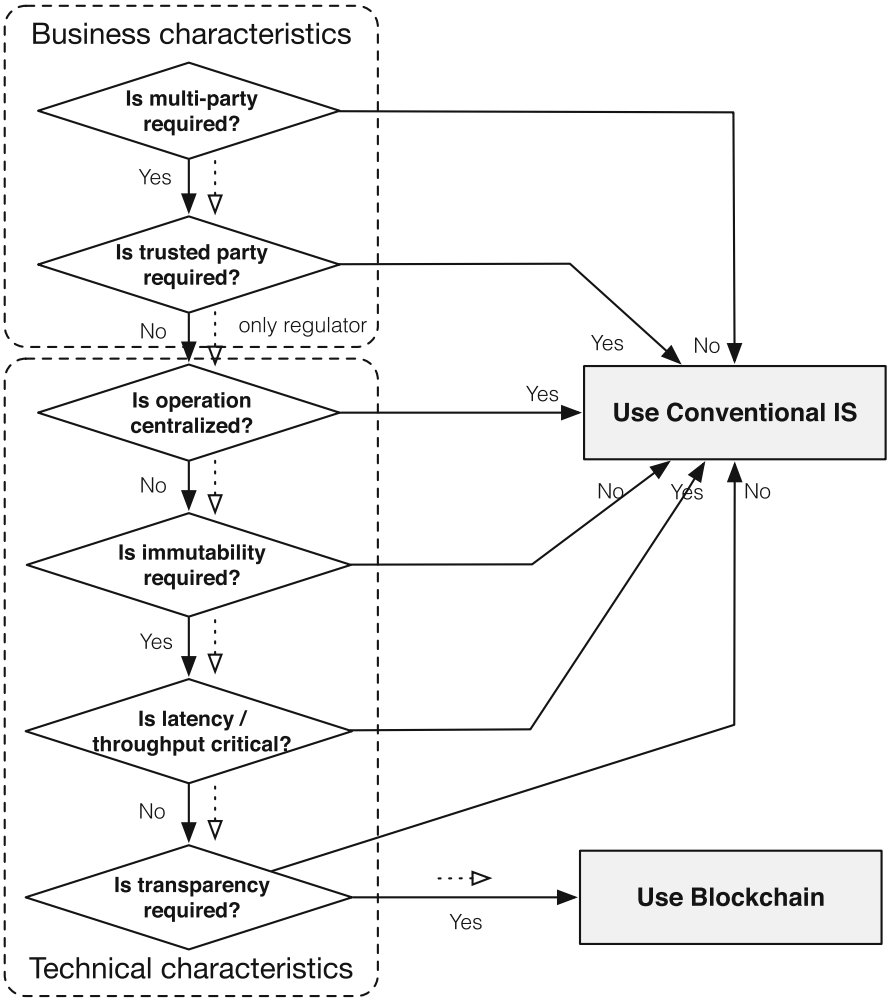


Fig. 2 Suitability analysis of applying blockchain in supply chains

well as decision-making and data publishing, has to stay with the authorities, and a blockchain will not replace them.

Is the Operation Centralized? Typically, individual participants operate their own ISs, which in the context of a supply chain are then integrated to varying degrees: from supply chains with no digital integration, over loosely integrated supply chains that, for example, rely on machine-readable barcodes to enable tracking and tracing, through to full system integration with digital message exchanges. In such a setting where the operations are not centralized, blockchain can act as an integration technology to record, coordinate, and communicate information and activities.

Is Immutability Required? Another advantage of using a blockchain is that it provides a non-modifiable record of transactions. That is, the current state of the data can be changed, e.g., when a supply chain item is transferred from one party to another or when the invocation of a smart contract enforces a quality requirement, while the history of those changes is recorded in the underlying data structure and cannot be removed or altered without consensus. It is thus often referred to as an immutable data store. Consequently, in scenarios that require immutability, the answer to this question is *Yes*. For example, this is the case when the history of business transactions or information about the chain of custody is needed to enable traceability of goods and materials to support recalls of products or to detect and control fraud and substitution among others.

Is Latency/Throughput Critical? The central mechanism of blockchain technology that ensures the immutability of data modifications is the consensus algorithm. It allows a network of nodes to synchronize the modifications to the data and to verify the validity of a set of proposed modifications. The synchronization of modifications, however, comes at the cost of limited transaction throughput. This primarily pertains to the writing of data to the blockchain, whereas reading the data from the blockchain is typically not restricted by the consensus algorithm. Thus, blockchain technology is not a good fit in scenarios where transaction latency and throughput are critical, e.g., when location and state data are streamed from IoT devices at a high rate. On the contrary, scenarios that require an infrequent exchange of data, for instance, when certificates of organizations are stored or when purchase orders are exchanged, are not subject to those restrictions. Note that the frequency and volume of data updates can be controlled through mechanisms like data aggregation or selection, e.g., when purchase orders are transmitted in bulk and only contain limited information. Further strategies for managing the throughput are outlined in Sect. 5. In conclusion, to answer this question, a thorough evaluation of the volume and frequency of data that needs to be recorded must be conducted. If the volume and frequency are low or can be reduced to an acceptable level, the answer to this question is *No*, and *Yes* otherwise.

Is Transparency Required? In a blockchain network, data are generally transparent and visible to all nodes that participate in the network and all devices with access to those nodes, respectively. This might pose a risk in situations where sensitive data should only be visible to selected participants in the supply chain and where the answer is *No*. However, in scenarios where data transparency is desired, and the answer is *Yes*, blockchain can be used as a communication platform, for example, when information from all participants is required to support logistics planning and identify and respond to problems. Moreover, controlled confidentiality that is required for open supply chain infrastructure can be supported by pairwise-party ledgers or by combining conventional information exchange technologies with hashed data on blockchains to ensure integrity and authorization. Additionally, in vertically controlled supply chains, confidentiality can be managed using a private blockchain. Note that Sect. 3 discusses blockchain architectures that limit the

visibility of certain data and, hence, support scenarios requiring a trade-off between transparency and confidentiality.

In the remainder of this chapter, we assume that a suitability analysis was carried out and that the assessment suggested blockchain as a means for implementing the supply chain scenario requirements.

3 Multi-Party Integration

Blockchain technology essentially provides a trustable integration platform that can establish a communication and coordination platform for the various organizations that participate in a supply chain, such as farmers, producers, carriers, and warehouses. Consider the example of a supermarket chain that wants to provide end-to-end food traceability to its customers, but the ISs that hold the relevant data reside at the individual participants and are only connected to the immediate one-step-up and one-step-down transacting parties. These data silos exacerbate the establishment of a unified view of a food item's history, as traceability breakdowns at each IS's boundary. To overcome this problem, the supermarket chain can attempt to integrate the ISs and make relevant data available on a blockchain, providing a transparent and non-reputable record of the food item's history to its customers. Moreover, this record can reduce regulators' efforts to investigate the compliance of the item flow with the regulation of each relevant jurisdiction. However, the supermarket chain also needs to ensure data and performance isolation across the different parties to enhance privacy and performance. In this section, we discuss different blockchain architecture designs that the supermarket chain can implement to balance non-functional properties like transparency, immutability, trust, privacy-preservation, and performance. While the supermarket scenario serves as an illustrative example, our discussion is general and, without the loss of generality, is applicable to other multi-party integration scenarios.

3.1 Requirements

The list below shows the key functional requirements for the traceability of food items and a unified blockchain-based system for multi-party integration of supply chain data in general:

- Providing the entire traceability history: The integrated system must provide functions for obtaining the entire data history of food items currently fragmented over the IS of the individual participants.
- Writing data to and executing functionality on the system: The involved parties must be able to write relevant data to the system and execute functionality deployed on the system with low cost and overhead.

- Providing external auditing/verification for independent regulators: Independent agencies shall be able to access data for auditing of each party.

In addition to these functional requirements, a range of properties need to be taken into account when designing a network architecture. In this section, we consider the following core properties:

- On-chain data integrity: The degree to which data pulled from each IS must be accurate and consistent
- Data confidentiality: The degree to which a party should be prevented from accessing data added by other parties unless there is a business requirement to do so
- Performance isolation: The degree to which one party's workload does not affect the read and write performance for other parties
- Availability: The likelihood for supply chain parties to be able to read or write data when desired
- Cost: The expenses associated with hosting, maintaining, and operating the blockchain solution
- Configuration flexibility: The extent to which communication and coordination functionality can be implemented on a point to point basis, i.e., the extent to which functionality does not need to be standardized across the entire supply chain

3.2 Solution Designs

Below we introduce three basic architectural designs for blockchain-based multi-party integration. An in-depth discussion of the properties of these architectures is presented in the next subsection.

3.2.1 Design 1: Using a Public Blockchain

The most straightforward design is to store data from all the supply chain parties on a public blockchain. This provides a trusted and neutral data infrastructure for all the parties, including the integrator and regulators. As shown in Fig. 3, anyone on the Internet can access the data stored on the public blockchain by either directly querying data from a node or by using a set of smart contracts such as the registry contracts discussed in Sect. 4. The communication between the IS of each supply chain party with the blockchain and the smart contracts deployed on the blockchain is implemented using the blockchain platform's API, e.g., REST-based *blockchain triggers* can be developed [12]. The *blockchain trigger* is responsible for sending blockchain transactions, querying smart contract states, and listening to transaction progress and smart contract events. Each supply chain party runs its own blockchain

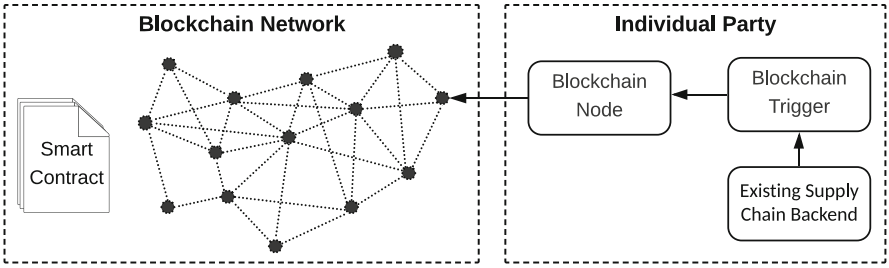


Fig. 3 Design alternative using a public blockchain

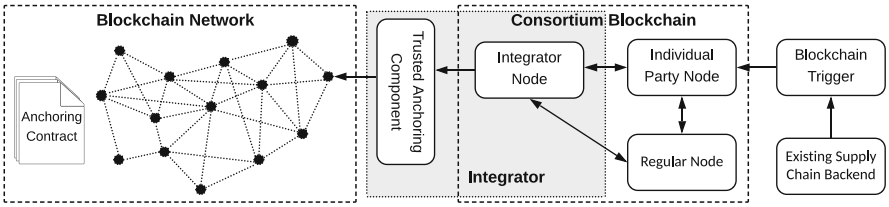


Fig. 4 Design alternative using a consortium blockchain anchoring to a public blockchain

node/client and hence sees all data stored on-chain. This also allows the integrator and regulators to query the data, as it is publicly accessible.

3.2.2 Design 2: Using a Consortium Blockchain

Due to its full replication across all parties, the public blockchain has very limited data storage capacity. Storing large volumes of data within a transaction may also be impractical due to the limited block size. Moreover, it is technically difficult to achieve data confidentiality on a public blockchain, as everyone can join the network and obtain access to the data on the blockchain. Thus, instead of directly storing raw data on the blockchain, a design option is to store a smaller representation of the data on the blockchain. A basic solution is to keep the raw data off the blockchain and then store a hash value (also called the digest) of the raw data on the blockchain [13, 14]. The hash value is generated using a consistent hash function, which maps arbitrary size data to a fixed size datum. While the hash value is stored on the blockchain, the raw data could be stored on a conventional data storage like a relational or a distributed database [15], the cloud, or a *peer-to-peer* (P2P) storage [16]. Other hash-based data structures, like the Merkle tree, can also be used to represent a large collection of data [17] using a single datum. The process of taking a unique representation of raw data and storing it in the blockchain is referred to as *proof of existence*.

Another design alternative based on proof of existence and a consortium blockchain is illustrated in Fig. 4. The consortium blockchain hosted by supply chain parties and regulators is used to store the raw data, whereas hashes or Merkle

roots of raw data are stored on the public blockchain. The consortium blockchain is viewed as a component that is *off* the public blockchain (aka., *off-chain*). Because a relatively small number of nodes host the consortium blockchain, it cannot achieve the same security level of a public blockchain. Therefore, to align with a public blockchain's security level, an anchoring mechanism is used in this design.

The supply chain integrator provides the *trusted anchoring component*. This component regularly queries the Merkle root of the consortium blockchain and then stores the Merkle root's value on the public blockchain, together with the block number and block hash of the consortium blockchain at the time of anchoring. The anchoring data are written into the *anchoring contract* on the public blockchain. The anchoring component is trusted by all supply chain parties involved. Its behavior can be verified anytime by comparing the anchored points from the anchoring contract on the public blockchain to the consortium blockchain's Merkle roots at the anchored time. In this design, the regulator also runs a node on the consortium blockchain to access the data for auditing purposes; hence, it can verify the data on the consortium chain like any other member.

3.2.3 Design 3: Using Pairwise Blockchains

While a consortium blockchain has better privacy properties than a public blockchain, data are still visible to all consortium members. Even if data are encrypted, secondary information such as transacting parties and the volume of transactions can be derived. This is unacceptable in a business confidential setting. However, supply chain data are often visible between a party and its one-step-up or one-step-down neighboring party. Therefore, another design option is to rely on a private pairwise ledger between a party and its one-step-up or one-step-down neighbor to gain most of the blockchain properties without losing privacy.

Figure 5 illustrates a third design alternative that maintains a collection of pairwise private blockchains among the supply chain parties. Each pairwise blockchain is privately hosted by a party, its one-step-up or one-step-down neighbor, and the supply chain integrator, such as the supermarket chain in the food item traceability scenario. Each party sends write requests to the corresponding *blockchain trigger*

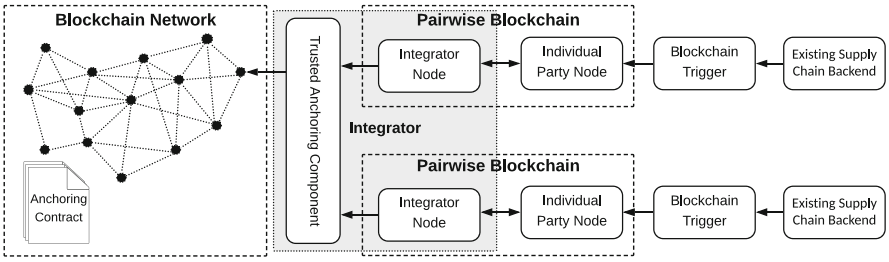


Fig. 5 Design alternative using pairwise blockchains anchoring to a public blockchain

component. The blockchain trigger provides functionality to write data to the blockchain and to communicate with the *trusted anchoring component*. The parties trust both the anchoring component and the blockchain trigger in the pairwise ledger.

Similar to the second design, the third design allows anyone to verify the integrity of a pairwise blockchain at a given time by anchoring the consensus state of each blockchain at periodic intervals to the public blockchain. Such an anchoring scheduler can be configured as agreed between the integrator and parties to the pairwise blockchain. The trusted anchoring component needs to connect to each pairwise blockchain using the respective *integrator node* and to the public blockchain. A Merkle tree T is used to integrate the latest global state from all the pairwise blockchains. Every leaf node of T represents the root of each pairwise blockchain's Merkle tree R_x. The newly created Merkle tree T is stored on each pairwise blockchain, and the root of T is stored on the public blockchain through the anchoring component. Individual parties rely on the integrator to handle the anchoring process. This leads to the design decision that only the integrator is hosting the anchoring component. Each party can continuously monitor that the integrator is performing the anchoring process correctly. In other words, the trust in the integrator only extends to it performing the anchoring.

The architecture can also be applied to store data with different characteristics to improve flexibility and reduce cost, e.g., based on geographic zones of transactions [18]. One of the variants for the third design is to grant the write permission of each pairwise blockchain to the corresponding parties. The integrator only provides the anchoring component. In this case, the integrator still needs to access each blockchain to get its latest global state. Other variants include alternative data structures for the dedicated pairwise chain, which could be a side-chain linked with the public blockchain or a shard [18] of the public blockchain.

3.3 Property Analysis

We now evaluate the three design alternatives against the properties from Sect. 3.1.

On-chain Data Integrity Data integrity could be achieved by all three design alternatives. In Design 1, each party holds a local copy of the public blockchain, through which they can access the data stored on the blockchain. In Design 2 and 3, data integrity is guaranteed by anchoring to a public blockchain. Design 2 stores the Merkle root of the consortium blockchain to the public blockchain, while Design 3 stores the Merkle root of the Merkle roots of each pairwise blockchain on the public blockchain.

Data Confidentiality To achieve data confidentiality, data can be encrypted before storing it on the public blockchain in Design 1 or on the consortium blockchain in Design 2. By contrast, in Design 3, access to data can be configured at a more fine-grained level. That is, only the immediate transacting parties and the integrator are

allowed to join a pairwise blockchain to access data. Hence, privacy is enhanced without the need to encrypt data. However, in this design, the integrator is a trusted party that needs to access each pairwise blockchain.

Performance Isolation In Design 1 and 2, parties with higher transaction volume might affect performance for lower-throughput parties, as all the data are written to a single blockchain. Also, the performance on a public blockchain might be further affected by the workload of other applications owned by parties that are not part of the supply chain and deployed on the same blockchain. In Design 3, transactions on one blockchain would not affect others, as each pairwise blockchain has its own blockchain trigger to write data to the corresponding blockchain. Lastly, the trusted anchoring component from Design 2 and 3 does not affect the write performance of the consortium or a pairwise blockchain, respectively.

Availability Design 1 can achieve high availability because it uses a public blockchain deployed on a large-scale P2P network. By contrast, the number of nodes in the consortium blockchain in Design 2 and the pairwise blockchains in Design 3 is much smaller than Design 1. Therefore, the consortium and pairwise blockchains' availability in these designs is not as good as in the first design. An option for Design 2 and 3 to enhance the availability is to add more nodes.

Cost Design 1 is associated with the highest costs because writing data to or executing functionality on the public blockchain typically requires the supply chain parties to compensate the nodes in the network in the form of transaction fees. Here, the costs are positively correlated to the volume of data stored on and the complexity of the computations executed on the blockchain. By contrast, the cost for anchoring to the public blockchain in Design 2 and 3 is minimized and constant, as only the Merkle root of the consortium blockchain (in Design 2) or the combined root of the Merkle roots of all pairwise blockchains (in Design 3) are written to the public blockchain at a predetermined interval. However, there are additional infrastructure costs, as the integrator needs to host one node for the consortium blockchain in Design 2 and at least one node for each pairwise blockchain in Design 3. Moreover, the costs for maintaining availability and maintenance might increase with a growing number of tenants.

Configuration Flexibility Both Designs 2 and 3 are blockchain platform agnostic. Permissioned blockchains with different consensus algorithms can be used for the consortium or pairwise blockchains. Also, both designs can anchor to any public blockchain that can store a small byte array in the form of a transaction memo. Hence, support for smart contracts is not essential. In Design 2, each party needs to agree on using the same blockchain platform, consensus algorithm, and configuration, whereas in Design 3, each pairwise blockchain could choose its preferred blockchain platform (e.g., one set of parties can use Hyperledger Fabric, while another party uses Ethereum), consensus algorithm, and configuration (e.g., inter-block time). The only component that all participants need to agree on is the anchoring protocol. By contrast, Design 1 requires all participants to agree on the applications and data formats that are hosted on the blockchain. Moreover, the

participants have to rely on the properties of the public blockchain and only have limited power to influence its governance.

4 Tracking Supply Chain Items

The need to track and trace items along the entire supply chain has become increasingly important, as it is an essential prerequisite to provide traceability, chain of custody, and provenance information. In the context of supply chains, *traceability* can be defined as the ability to identify and trace the history of an item from its materials and parts to its production process, distribution, storage, location, and application. *Chain of custody* is a form of traceability that focuses on the chronological documentation (time, location, party) of the sequence of custody, control, transfer, analysis, and disposition of an item. *Provenance* provides information about an item's origin or source and is typically proven through laboratory testing and certification.

There are many use cases to which this information is relevant, as the following list of examples outlines. First, supply chains need to be resilient to counterfeiting to avoid introducing and disseminating fake products. Moreover, consumers are starting to demand more transparency related to the products they purchase [19]. For example, increased awareness regarding food quality has led to a growing interest in the origin of ingredients and food items, production processes, and adherence to sustainable and ethical practices. Furthermore, to swiftly contain and respond to food-related outbreaks, supply chains need to understand where infested food items originated from.[4] Lastly, stakeholders also want to know the status of an item at every point along the supply chain to align their production practices.

These use cases necessitate that the (i) data about items are captured in the supply chain management systems; (ii) events such as the sequence of custody, location, and time of an item are recorded; and (iii) auxiliary data like accolades gained by items or their manufacturer are attached. Ultimately, this information enables the tracing of items forward or backward from any point along the supply chain and to derive useful business insights. With that in mind, we discuss how a blockchain-based solution can provide trusted, highly available, and fast-tracking of supply chain items.

4.1 Blockchain Registries

A *registry* is a list of records maintained by a trusted authority and is expected to have high data integrity, availability, transparency, immutability, and the ability to

[4] https://neurochaintech.io/supply-chain-traceability-romaine-lettuce/.

connect with other registries [20, 21]. Blockchain-based registries are a solution for storing and publishing traceability, chain of custody, or provenance information without the need for a trusted authority while achieving the above properties [21]. Additionally, blockchain can be used as a unified infrastructure that enables multiple registries to interact with each other easily.

In fact, the establishment of such registries has emerged as a central use case for blockchain. The first blockchain system, Bitcoin, was designed as a distributed ledger, i.e., a registry for cryptocurrency. This limited use case was later expanded by second-generation blockchains which supported the storing of arbitrary data and the execution of custom programs. Ethereum, for example, has even standardized contracts that capture registry information. That is, the Ethereum Request for Comment[5] (ERC) includes a set of standard development interfaces that smart contracts must comply with when storing tokens, i.e., digital representations of real-world entities. Here, ERC-20[6] regulates the handling of fungible tokens that represent interchangeable entities. By contrast, ERC-721[7] defines how non-fungible tokens are managed, where each token represents a unique entity. In the context of supply chains, non-fungible tokens can be used to represent items, transportation units, assets, and certificates among others. While cryptocurrencies are a well-established kind of fungible tokens, they might also be used to represent quantities of fuel, grain, or other types of bulk goods in supply chains. Another example of a blockchain-based registry is Namecoin,[8] which is a domain name registry that shares the same network with Bitcoin. Further, Abscribe[9] is an artwork registry that enables artists to maintain ownership of their digital artwork.

In supply chains, registries are typically used to capture information about items. In this regard, basic properties of items include information on whether they are considered as individuals or groups of items (e.g., a single diamond vs. a box of apples); as low, medium, or high value (e.g., from potatoes to diamonds); and perishable, one-time use, or as long-lasting (e.g., apples, surgical masks, and jet engines). Each item or group of items on a supply chain is represented using a unique ID. The primary stakeholders involved in a supply chain item tracking system are the product or brand owners, clients, end consumers, and supply chain parties. *Clients* are organizations that buy items from the *product owner*. Once an item is purchased, the ownership is transferred from the product owner to the client. *End consumers* are interested in making purchase decisions; hence, they want to verify the authenticity and production practices of an item, e.g., origin of a wine bottle and when it was bottled. End consumers may access this information via an application provided by the customer or product owner. Change of ownership may also be recorded at the time of purchase, especially for high-value or long-

[5] https://docs.ethhub.io/built-on-ethereum/erc-token-standards/what-are-erc-tokens.

[6] https://eips.ethereum.org/EIPS/eip-20.

[7] https://eips.ethereum.org/EIPS/eip-721.

[8] https://namecoin.org/.

[9] https://www.ascribe.io/.

lasting items. Supply chain *parties* include farmers, carriers, and warehouses. They generate various events as an item flows through the supply chain. While only the product owner should be able to register the ID of an item, other stakeholders should be able to add event and auxiliary data. In summary, this leads to the following list of basic functional requirements that blockchain-based registries must implement:

- Unicity: Ability to register a unique identifier (ID) for each supply chain item
- Record event data: Ability to record supply chain event data, e.g., location and time that one party handover an item to another party
- Record auxiliary data: Ability to record additional data such as expiry date and accolades gained by the item or its manufacturer
- Query data: Ability to search event and auxiliary data of an item towards the origin or consumer from any point on the supply chain

While there is out-of-the-box support for blockchain-based registries, the establishment of registries that deliver these requirements typically warrants additional development efforts based on various ad hoc techniques. That is because standards like ERC-20 or ERC-721 abstract from the specific use case and only define how tokens and ownership information are managed, but not how related information is handled. Hence, the standardized token registries need to be adapted, causing additional development efforts and challenges. Here, the need for developers to gain a deeper understanding of how a particular blockchain platform operates and to master the respective smart contract programming language to implement registries that provide the aforementioned functionality further exacerbates the problem. Considering the broad applicability of registries in supply chains and the efforts required to set them up, we outline an approach to develop such registries in the next subsection. In this regard, we only focus on the functional requirements and abstract from non-functional requirements such as scalability or availability. To implement those non-functional requirements, developers can rely on any of the design options from Sect. 3.

4.2 Registry Architecture

A generic design for blockchain-based registries for item tracking systems is put forward in Fig. 6. In this design, all item-related data is stored in three registries where each of the blockchain-based *registries* is a list of records with high data integrity, availability, and the ability to connect with other registries [21]. The three contracts—*ID*, *ID event*, and *ID auxiliary data* registries—implement the functional requirements. That is, unique IDs of supply chain items are recorded in the ID registry smart contract, which provides the functionality to add, update, and query an item. IDs from this registry establish the existence of the respective item. Similarly, ID event and ID auxiliary data registries record supply chain events and auxiliary data of items, respectively. Anyone with access to the blockchain can read the unique ID information stored on the blockchain using the registry smart

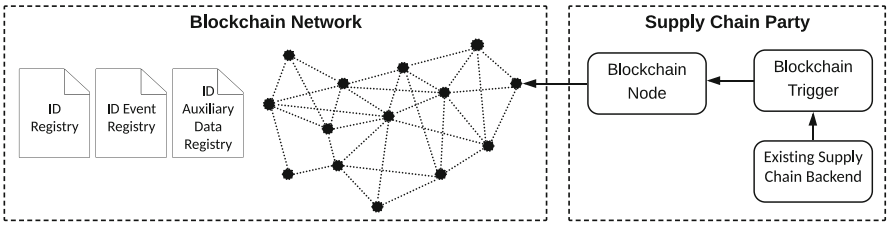


Fig. 6 Architecture of blockchain-based item tracking systems

contracts. Moreover, this data can be jointly queried via the blockchain trigger, so that backend systems can extract the information specific to a certain use case. Separating the item data into these three registries enforces a strong cohesion where data and management functionality that address similar concerns are grouped into smart contracts with clear and unified purposes.

The existing supply chain management system's backend communicates with the registry smart contracts via the API exposed by the *blockchain trigger* [12]. For example, when the supply chain backend makes an API request to create a new unique ID, the *blockchain trigger* first queries the *ID Registry* smart contract to check if the ID already exists. If not, it sends a transaction to register the new ID. Otherwise, it reports the existence of the ID to the backend. Next, it listens to the transaction's progress, i.e., it waits for the inclusion of the transaction into a block. Once the transaction is confirmed, it responds to the backend via the API.

4.3 Model-Driven Registry Development

While the basic architecture for registry design provides a generic pattern for dividing registries into modules, it does not explain how the registry functionality is adapted to the specific scenario. To addresses this challenge, we designed and developed a tool called Regerator [21], which could generate the three registries in Fig. 6. Regerator adopts principles from *model-driven development* (MDD) and *model-driven engineering* (MDE). That is, it provides templates for the developers to customize registry data models and then transforms these registry models into smart contracts following the basic design from Fig. 6.

MDD is a software development methodology that uses models at various abstraction levels to address software development complexity. Domain-specific MDD can help map the model of the problem domain to the design of the software solution [22]. MDE is a particular case of MDD in which models are used as specifications from which source code or other artifacts like database schemas can be generated. Depending on the use case, various concepts can be captured in models and in general models hide implementation details, providing a means to specify a software configuration at a higher level of abstraction. In the context of

blockchain-based applications, MDD and MDE are of particular importance for the following reasons [23]. First, MDE tools can implement best practices and generate well-tested code, including smart contracts, thereby minimizing vulnerable code. Second, models can avoid lock-in to specific blockchain technologies because they can be platform-agnostic, and an MDE tool might be able to produce artifacts for multiple blockchain platforms. Third, models are easier to understand than code, thus improving developer productivity. It is easier to check the correctness of a model, and tools can ensure that the deployed code has not been modified after its generation from the model. Fourth, it can facilitate communication with domain experts, as domain experts can look at the model to understand how their ideas are represented in the system.

Regerator relies on MDE and allows users to specify registry models that can then be translated into code for blockchain-based registries. A registry model consists of three parts, including basic information, basic operations, and advanced operations. *Basic information* refers to the properties of the registry and includes registry name, type, and user-defined attributes. The registry type can be either single or distributed. The single registry is suitable for simple registries such as the registries in Fig. 6. A simple registry holds all data records in a single data store or smart contract, respectively. The distributed registry is designed for registries with complex operations (e.g., record-level access control), which manages each record as a separate smart contract and stores pointers to them. Regarding user-defined attributes, users can specify the attribute name and type. *Basic operations* are the operations that can be performed on an individual record, including create, read, update, and delete and existence checking. Users could configure whether or not a record is updatable. The *advanced operations* include record lifecycle management and access control. The registry records the lifecycle of a supply chain item and could be managed via a business process executed on the blockchain. Regerator currently generates registries for Ethereum but can potentially support additional blockchain platforms.

5 Digital-Physical Linkage

By creating digital twins of physical items on a blockchain, we can track the movement of goods and make various claims about their constitution, traceability, chain of custody, and provenance. What is common to all of these claims is the need to capture an item's physical attributes, such as its identifier, quantity, quality, time, location, and parties that had access to it. However, a blockchain can only guarantee immutability, consistency, transparency, and availability of the stored data, but it cannot ensure that we capture and act on the correct data. Hence, it is crucial to ascertain that the data about items and parties on a blockchain correctly reflects the respective claim and represents the actual physical state. Therefore, mechanisms that establish and guarantee the integrity of the linkage between real-world items

or parties and their digital twins on a blockchain are needed. In the following, we extend the discussion from Sect. 4 to link physical items to their digital counterpart in an efficient and trustable manner.

5.1 Techniques to Capture Physical Properties of an Item

Commonly used tagging technologies such as serial numbers, barcodes, and QR codes can connect a physical item to its digital representation. However, such techniques are vulnerable to attacks, e.g., a counterfeit item with authentic packaging and tag might mistakenly be recognized as a genuine item [24]. It is necessary to enable supply chain stakeholders to verify the connection between an item and its digital twin to counter fraud. This can be achieved at different levels with varying costs and benefits.

Certification is one of the most established ways of proving the authenticity and quality of an item and processes used in manufacturing, logistics, and retail. A certificate typically contains information about (i) the party that issued the certificate, (ii) the party that it was issued to, (iii) when and where it was issued, (iv) a set of attributes that are certified (e.g., laboratory test results, production capacities, and carbon footprint), and (v) a signature(s), seal, or watermark to prove its authenticity. As certificates are issued by third parties that may not be members of the blockchain, we need specialized modules to track their identities and provide controlled access for writing the specified data.

Another approach is to track items based on supply chain events and physical attributes. For example, double scanning of items, reuse of packaging, or introducing white-labeled items can be detected by tracking the behavior of items from their origin to their destination and by verifying new events against the recorded information. Also, instead of recording the average ambient temperature of a container at the end of the shipment, a contemporary cold chain can use IoT-style sensors to track and report an item's temperature at much higher spatial and temporal resolution. In this regard, the ability to capture and store this data immutably on a blockchain goes a long way in enhancing the trust in the supply chain, when this data is made available to stakeholders or, for example, used to respond to outlier events in real time.

However, as fraudsters attempt sophisticated ways to adulterate an item, an item's provenance can be ultimately proven only by laboratory testing. For example, Stable Isotope Ratio Analysis (SIRA) is a technique in this direction where chemical isotope levels on food products (e.g., on muscles of cows or fish tissues) can be analyzed to verify claims of origin. However, techniques like SIRA and genomic analysis require manual intervention and hence are too slow and uneconomical at large scale. Today, IoT and machine learning techniques can supplement such laboratory testing by performing low-fidelity testing on large volumes of items while saving time and cost [24]. Therefore, recording IoT data on the blockchain and assessing it based on results from machine learning models can contribute to

achieving provenance. However, a blockchain can only store and process a limited volume of data and instructions per transaction, respectively. Moreover, certain aspects of third-party services such as certification, field and laboratory testing, and weather data sources remain outside the blockchain. Therefore, this information also needs to be incorporated in a trustable manner.

5.2 High-Level Architecture for Digital-Physical Linkage

Figure 7 illustrates how external data sources that support various techniques for ensuring the digital-physical linkage can be integrated into a blockchain-based system. While the design includes certificate management, IoT sensors, and multiple claimed attribute assessment techniques, specific implementation may include only a subset of those. Also, note that the proposed design is, in principle, applicable to all architectures for multi-party integration from Sect. 3.

A *certification management system* that sits outside the blockchain (aka., off-chain) can record the issuance, verification, and revocation of a certificate using the *certificate registry* smart contract on the blockchain (see Sect. 4). Real-world certificates with a large set of certified attributes are likely to be too big to fit into a blockchain transaction. For example, electronic phytosanitary certificates (ePhyto) that are used to certify that plant-based items meet the receiving country's biosecurity requirements are represented as XML files [14]. In such cases, the certificate file can be stored in *off-chain storage* while adding the hash of the certificate file to the blockchain (aka., on-chain) as a proof of existence.

Each instance of the *key management application* safely stores the private key of a third-party, cryptographically signing certificates or laboratory/field visit reports, and records those on the blockchain as transactions. A key management application

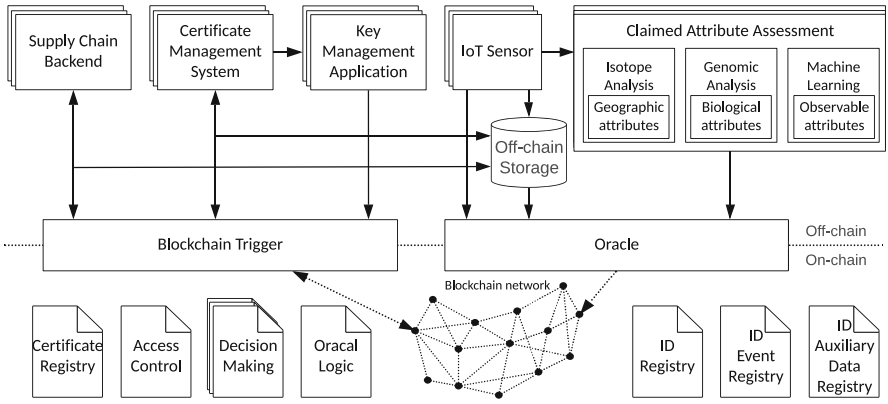


Fig. 7 Design of a supply chain management system to enhance the digital and physical linkage of items

and *access control* smart contract together can enforce who can write what data on to the blockchain. This way, we can ensure that only authorized third parties can provide attestations about the items and processes on the supply chain.

IoT sensors provide event data such as time, location, and ambient measurements to the *ID event* and *ID auxiliary data* registries. Such data are useful in demonstrating the traceability and chain of custody of an item. *Claimed attribute assessment* modules could include a spectrum of techniques, such as machine learning techniques to validate observable attributes captured by IoT sensors, SIRA to validate geographic attributes, and genomic testing to validate biological attributes. Data from both IoT sensors and claimed attribute assessment modules are also used by the *decision-making* smart contracts on the blockchain to calculate item quality and detect outliers and to then make various business decisions regarding the flow of items along the supply chain. Also, validation results reported by the claimed attribute assessment modules are recorded on the *ID auxiliary data registry* as provenance information.

As a blockchain cannot communicate with the external world, *oracles* are required to enable the interaction between blockchain and the off-chain components such as IoT sensors and claimed attribute assessment modules. Blockchain oracles are agents that fetch data from the external world into the blockchain [25]. While most parts of the oracle service lie off-chain, *oracle logic* smart contracts are used to inject oracle data and provide on-chain data access to other smart contracts [25, 26]. Here, oracle logic can be used to either push or pull data from the external world into the blockchain [26]. Also, an oracle is essential to ensure that all the blockchain nodes see the same value to ensure that consensus can be reached even though IoT sensor data are transient. Moreover, they can filter out IoT data, run complex computations off-chain, and then only make critical tracking events (e.g., outliers) available to the blockchain. This drastically reduces the data to be written into the blockchain, as raw data are stored in an *off-chain storage*, and only their hashes, attestations, or metadata are recorded in the on-chain oracle logic. Furthermore, oracles can provide data from other external services such as weather and exchange rates to smart contracts executing various business logic on the blockchain. For example, a penalty may not be enforced on the carrier due to late delivery when there is evidence of bad weather that may have affected the timely delivery.

5.3 Property Analysis

Off-chain Data Integrity At the blockchain level, data integrity is achievable as each party holds a local copy of the blockchain. If a consortium blockchain or a set of pairwise blockchains is used, anchoring to a public blockchain can enhance integrity. Further, the proposed design enhances the integrity of data fed into the blockchain by ensuring that the digital representation reflects the actual physical item and its current state. For example, in addition to the accuracy of data attested by a certificate, a key management application can establish a certificate's authenticity

by digitally signing it before storing it in the blockchain. In fact, all off-chain modules can use the key management application to digitally sign their data to establish authenticity. Moreover, access control smart contracts ensure that only authorized parties can update the data. Claimed attribute assessment modules can confirm whether the supply chain items are the ones they claim to be in terms of their geographic, biological, and observable attributes. IoT sensors' ability to provide data at a high spatial and temporal resolution enhances the quality of data fed into the blockchain, as occasional misses or errors in data can be supplemented by the stream of data. Data placed by oracles further confirms the accuracy and authenticity of external data.

Trust The proposed architecture can enhance trust in various ways. Trust in the data that is stored on the blockchain depends on how trustable the source of data and the processes for collecting, transforming, and storing the data on the blockchain. Here, evidence that data have been provided by a certification body, a claimed attribute assessment module, an IoT sensor, or an oracle can increase the trust. To confirm that a particular party provided data, it can digitally sign the data. Moreover, key management applications can ensure that digitally issued certificates belong to trusted third parties by limiting write access to those parties. Alternatively, a third party may present its self-sovereign identity that could be verified on a public blockchain by other supply chain parties. Trust in certification bodies is typically based on their reputation. As all actions by the *certificate management systems* are transparently executed on the blockchain, supply chain parties can observe a certification body's behavior and assign an appropriate reputation score. Trust in claimed attribute assessment techniques depends on their accuracy. For example, typically, SIRA and genomic testing are more trustable than machine learning techniques. Trust can also vary depending on the chosen machine learning model and the party responsible for its execution. Therefore, their assessments need to be weighted by the reputation and trust score of the model and its executor. For example, in [24], we presented a neural network-based classifier to discriminate between grain-fed and grass-fed beef based on regular and thermal images of the meat texture. We further introduced a blockchain-based reward system to incentivize honest behavior and the ingestion of accurate data. Redundant or different types of IoT sensors and oracles can be hosted by independent parties to collect and attest the same external data, and the majority of their attestations can be considered as the ground truth [25]. A reputation score can then be assigned to their data accuracy and behavior, too.

Scalability While frequently tracking an item's location, time, quality, and other attributes enhances traceability, a blockchain cannot store large volumes or high-velocity data. Instead, such high volume and velocity data can be stored on the off-chain storage. Then, their hashes, attestations, and metadata can be provided to the blockchain via an oracle, such as proposed in Sect. 3. Further, oracles can process such data off-chain and inject only the critical tracking events into the blockchain. Therefore, this design enables many status updates to be tracked off-chain, while data stored on-chain remains within the transaction size and throughput limits.

Availability Because each off-chain component in Fig. 7 runs a few instances, their availability is lower than the blockchain with a large number of nodes. This negatively impacts the availability of the overall supply chain management system. Moreover, some components are more mission-critical than others. For example, more components typically rely on an oracle than those that rely on a certificate management system. Hence, their availability needs to be critically assessed and enhanced to increase the overall availability. In [25], we presented a framework to analyze an oracle's reliability based on several attributes, such as the number of oracle nodes, parties involved, reliability and isolation of data sources, and the voting mechanism. In general, typical practices such as active-active and active-passive high-availability designs of off-chain components can enhance the system's overall availability [25].

6 Business Process Management

The coordination of organizations within a supply chain is typically assigned to a dedicated supply chain member. Problems that arise in this setting include a lack of trust in the coordinator, inefficiencies due to a lack of transparency, and increased supply chain complexity due to communication overhead. The ability to execute arbitrary functionality defined in a set of smart contracts has been recognized as a means to overcome those problems [2]. That is, in addition to using the blockchain as a transparent data store, smart contracts provide a means to encode business rules and reliably enforce them during supply chain execution. Hence, smart contracts unlock the potential to improve supply chain coordination and governance. In particular, relying on smart contracts to coordinate the individual parties can reduce complexity, as a blockchain network reliably performs communication and rule enforcement, instead of being transferred to a coordinator. This ultimately reduces the number of errors and also can eliminate security risks like man-in-the-middle attacks. Considering these potentials, we provide an overview of how collaborative supply chain processes can be managed on the blockchain.

In this regard, *Business Process Management* (BPM) offers a rich toolset for designing, implementing, and analyzing business processes [27]. In recent years, those tools have been adopted to enable blockchain-based processes. In this regard, the following aspects are of particular relevance for collaborative supply chain processes:

- Enforce business rules: Define smart contracts that coordinate activities performed by different organizations while ensuring that predefined business rules are adhered to during the process execution. At the process level, this pertains to the order in which activities are carried out, while at the activity level, it includes operational procedures and service levels that organizations must comply with.
- Integrate the data flow: Facilitate the exchange of data along the process to support the communication of relevant information and the selection of

appropriate execution paths. This includes on-chain data that are created or consumed during the process execution, as well as data received from and sent to off-chain components that interact with the smart contracts.

- Analyze process behavior: Obtain insights into the execution of the collaborative processes to support decision-making. At run-time, individual process instances must be monitored to react to unforeseen circumstances on time. In the analysis phase, historic process behavior is evaluated to identify bottlenecks and potential for improvement.

In the following, we introduce approaches that implement those requirements. We conclude by discussing advanced topics that are not covered in this section.

6.1 Techniques for On-Chain Process Management

In BPM, process models have long served as a foundation for many activities in the life cycle of a business process, including process automation and analysis [27]. That is, otherwise, intangible business processes are captured in process models that contain information from various perspectives: (i) the functional perspective defines the activities that are carried out during the process execution, (ii) the control flow perspective defines the execution order for those activities, (iii) the organizational perspective specifies the roles that are responsible for individual activities, and (iv) the informational perspective defines the data that is created and exchanged.

To illustrate how process models record information about business processes, Fig. 8 introduces a running example for this section. The figure shows a simplified and idealized model for how picking orders are processed in a distribution center. The notation that the process model relies on is the *business process model and notation*[10] (BPMN), which is a widely adopted notation in industry and academia and that also serves as a foundation for the approaches from this chapter. The process starts when the branch places a picking order with the distribution center. The branch also needs to arrange the pickup and nominate a driver to collect the goods on behalf. Note that the branch might change the nomination at a later point in time. Based on the branch's input, the distribution center schedules the pickup and authorizes the nominated driver for pickup. When the scheduled date is reached, the goods are picked and handed over to the nominated truck driver. In this regard, the driver needs to provide proof of identity, which is then verified, before goods are handed over. However, before the driver can transport the goods, both parties have to sign off the handover and, this way, acknowledge that the correct quantity of goods has been transferred in the specified quality.

[10] https://www.omg.org/bpmn/.

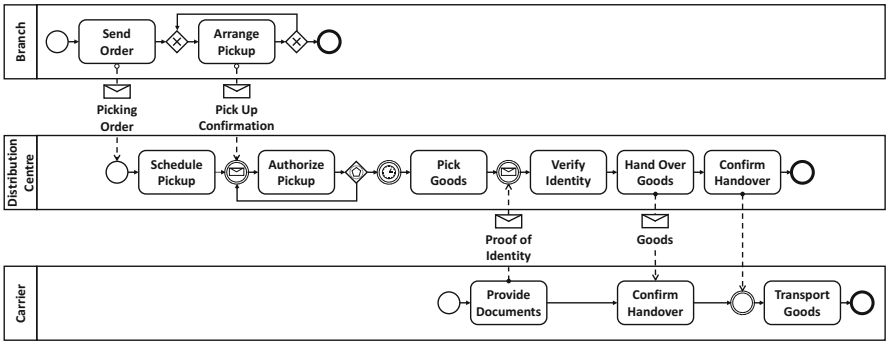


Fig. 8 A collaborative process model for the processing of picking orders at a distribution center

6.1.1 Enforce Business Rules

The approaches for executing, coordinating, or recording process instances and enforcing business rules on blockchain networks build on top of process models and adopt the workflow engine paradigm. A workflow engine can interpret process models and automatically execute the respective processes based on the rules from the model. Accordingly, blockchain-based process execution approaches work by translating process models into a representation that can be deployed to and executed on the blockchain. Similar to the MDE approach for registry design (see Sect. 4.3), these approaches have the advantage that organizations can focus on the business perspective when determining the design of the collaborative processes and the business rules that the processes need to adhere to, while the executable version of the process is automatically generated, minimizing sources for development errors.

The first type of approach is *compiler-based*. That is, these approaches translate a process model into a set of executable smart contracts. We here discuss the approach from [12] for which a refined version that generates more efficient smart contracts was later introduced in [28]. This approach takes a BPMN-based collaborative process model as input and compiles it into two smart contracts that can be executed on the blockchain. The *process contract* constitutes the executable version of the process model. However, it is not directly deployed to the blockchain. Instead, it is wrapped into a *factory contract* that can be used to instantiate the process contract for each process instance. It also serves as the interface for interacting with the process instances. Regarding to the example, this means that the factory contract instantiates a process contract for each pickup order, and this smart contract then manages the processing of the respective pickup order. In more detail, a process contract records the state of the process instance, making sure that the recorded state is valid. That is, whenever it receives a message regarding a process event, e.g., when an activity is completed, it evaluates the event against the process logic and the current state. It updates the state of the instance only if the event was expected and follows the control flow specified in the process model. Otherwise, it rejects

the update. For example, in the picking order scenario, the process contract would only record the beginning of the transportation, after receiving the confirmation for the handover from both the distribution center and the carrier. Finally, the approach distinguishes between two different types of *process contracts* that it can generate from process models. A *choreography monitor* solely records and makes available information about the current state of a process instance, whereas a *mediator* also plays an active role in process execution, coordinating the execution of activities. That is, a mediator sends messages to off-chain applications by writing log entries to the blockchain's log regarding the state of an instance, which the off-chain components can listen to.

A limitation of the approach from [12, 28] is that it does not support the upgrading of business processes. That is, all changes that need to be made to a process, no matter how small, require a full development life cycle of updating the model, re-generating the entire set of smart contracts, re-deploying them to the blockchain, and updating the integration with the off-chain systems. This potentially leads to situations where multiple smart contract versions need to be run in parallel to support the completion of ongoing process instances. Thus, updating an existing solution can become an expensive task. To overcome this problem and to progress on the path of delivering a blockchain-based workflow engine, recent work [29] has proposed to follow a different paradigm: the *interpreter-based* process execution. In contrast to the compiler-based approach, a process model is not translated into a set of smart contracts. Instead, a generic interpreter contract that implements the BPMN semantics is used to execute representations of BPMN models. Here, the interpreter receives and operates on an optimized data structure for serializing BPMN models. Moreover, the interpreter can store multiple process models such that different variants of the same process can be operated in parallel. If only one process variant can exist, its older variants can be deactivated. Finally, the interpreter also manages the state of the process instances.

So far, the approaches primarily focused on enforcing business rules at the process level and, in particular, rules that are related to the control flow logic. To support the enforcement of business rules at the activity level, such as product-quality specifications and production limits, merging and splitting of items into batches, and mixing of ingredients, a smart contract-based approach is presented in [30]. For example, let us assume that the branch in Fig. 8 is a hardware store that is ordering a set of doors from the distribution center to be delivered to a construction site. The picking order sent by the store is specified in the form of a Bill of Materials (BoM) and consists of a matching set of finished doors, door frames, locks, and screw sets to be delivered as a single shipment. All four items must arrive in a single shipment with the specified quality and quantity, as any missing doors, wrong combinations, or a different quality would lead to a delay, which potentially results in the violation of service-level agreements. Further, there may be multiple orders for the same construction site or branch. While aggregation of orders is beneficial to the stakeholders, certain items should not be transported together. For example, in the case of finished doors, it is unsafe to carry liquids, such as paint, paint thinner, turpentine, liquid bleach, and cleaning fluid in the same shipment.

All these rules constrain the activity of order picking. The approach from [30] represents such rules as a *validation specification* (VS) that can be enforced via a smart contract. Figure 9 depicts the respective structure for the door picking example. Here, *item types* are represented as circle-shaped vertices and comprise the four parts for the door and the shipment. Moreover, the *activity* of picking doors is shown as a diamond-shaped vertex which transforms the four parts of a door into a shipment, as shown by the edges. The distribution center, shown as a stickman, is defined as the *handler* of the picking activity and is thus responsible for carrying it out. By contrast, the hardware store *regulates* the activity by providing the VS used to *validate* the activity. In the example, the VS comprises of two things. First, it contains the BoM, which defines the specific instances that need to be considered. Second, it subsumes the set of rules that the activity needs to adhere to and that are used to validate the BoM. While in the example, the hardware store sets the rules for order picking, this could, in principle, be regulated by other parties like a regulator that determines prohibited combinations of items. This graph is then stored on the blockchain to track supply chain relationships and to enforce data access rights using access control smart contracts [30] (see also Sect. 5). The graph can also be updated as the items move along the supply chain.

In [30], we provide a generic template for specifying VSs. A VS is defined as a JSON document and consists of a set of fields, such as compare, ratio, quota, and exclusions. For example, a *compare* field can be used to define item attributes and mathematical operators to compare them, e.g., dimensions of a door. A *ratio* field validates the ratios between attributes, e.g., the ratio between a door and the number of screws. Further, a *quota* field could be used to validate whether production quotas are not violated along the supply chain. In contrast, an *exclusion* field can specify that a finished door and thinner cannot be carried in the same shipment. At the time of handover of goods, the VS is validated dynamically using a special smart contract that can interpret and enforce a given VS. This smart contract ensures that only a

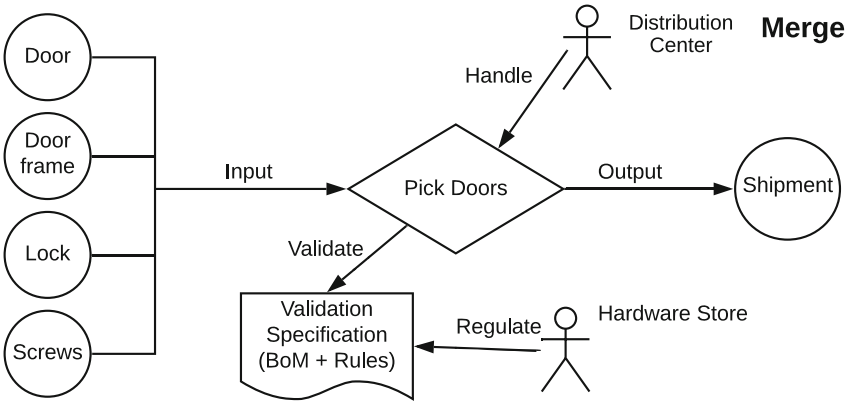


Fig. 9 Merging of items in a supply chain

shipment that can satisfy the VS could be transferred to the carrier. Otherwise, the process is blocked until what leads to an invalid VS is resolved.

When the VS is valid, and the confirmation for the handover from both carrier and distribution center is received, the handler to the shipment is automatically transferred from the distribution center to the carrier. Because an authoritative body entered the VS (in this case, the hardware store) into the blockchain, and the smart contract enforces the VS before adding item data to the blockchain, the relationship acts as the authoritative body's endorsement on the validity of the data. The same idea can be extended to enforce merging and splitting of items as batches, mixing and transforming of items (e.g., mixing milk and salt to produce cheese), and product-quality specifications and production limits within the blockchain [30].

6.1.2 Integrate the Data Flow

So far, we only looked at ways to encode business rules at the process and activity level in smart contracts, so that they can be enforced at run time. However, the execution of these processes and process instances requires integration with other off- or on-chain components. On the one hand, processes and their instances are not executed automatically. Instead, to create or update an instance, other components must send messages and data to the respective smart contracts. In Fig. 8, the branch, for example, needs to be able to send the picking order. On the other hand, the smart contracts only record information about the state of the instances and make sure that this state is coherent with the underlying process model. The smart contracts do not include any specific functionality that needs to be executed when an activity is started nor do they ensure that the intended recipient actually receives the message. Consequently, the communication between the smart contracts that represent the process (instances) needs to be configured to implement the transmission of data to and from the respective stakeholders or to automate certain computations. For example, in Fig. 8, the generation of the process smart contract does not include an invocation of the smart contract for order picking from Fig. 9. Such an invocation would thus need to be set up manually. While in this regard, the considerations from Sects. 4 and 5 generally apply, there are a few interesting proposals for easing the integration of external components into smart contracts for process execution.

In the context of compiler-based execution, the generation of application-specific interfaces for integrating on- and off-chain components has been proposed. With respect to the off-chain components, the compiler-based approach from [12] can, in addition to factory and process contracts, generate a set of blockchain triggers. These off-chain components provide role-specific interfaces and are intended to be run locally by each organization involved in the process. As these organizations have different responsibilities in the process, i.e., they carry out different activities, the blockchain triggers only contain querying and interaction functionality relevant to the respective organization. In the example from Fig. 8, there are three roles represented by the three swim lanes (branch, distribution center, and carrier). For each of these roles, the compiler would generate a specific blockchain trigger. For

example, the branch's blockchain trigger would only include the "Send Order" and "Arrange Pickup" functionality, as well as generic functionality for tracking the state of process instances. This way, the blockchain triggers ease the integration of off-chain components and support role-specific interaction functionality.

Regarding the on-chain components, the integration of the compiler-based approach from [12] with the registry generator from Sect. 4.3 was outlined in [31]. Here, experts can bind the data schema specified in the Regerator [21] and, in particular, the respective update operations to activities in the process models at modeling time. To provide experts with a means for binding data schemas from the Regerator to a process model, the approach extended the BPMN standard with custom attributes, so that the required invocation of update operations can be linked to the process model. At compile time, the Regerator and the process model specification are independently translated into smart contracts, and during the deployment, they are then woven together by registering the Regerator contract address with the process factory contract.

Contrary to generating specialized interfaces, standardized interfaces that off- and on-chain components can use to interact with interpreter contracts have been proposed. That is, the approach from [29] defines a generic and process-agnostic off-chain facade that can be utilized to deploy process models to the interpreter smart contract and then instantiate and update process instances. Moreover, there are two types of on-chain interfaces that standardize the interaction between process instances and other components [32]. Hence, these interfaces facilitate reuse and extensibility. That is, they allow for binding or plugging in any smart contract that implements one of these interfaces into activities that are part of the processes managed by the interpreter contract. Here, the *worklist handler* interface must be implemented by smart contracts that handle user interactions, and the *service bridge* interface is reserved for smart contracts that interact with on- or off-chain components.

Based on the approaches for business rule enforcement and data flow integration, two tool suites for model-based process execution on the blockchain have been developed [33]. *Lorikeet* is a compiler and web-based development environment for Ethereum. It can be used to design, customize, deploy, and interact with process and registry-based smart contracts. The application of the system was demonstrated in [34], and the respective screencast is publicly accessible.[11] *Caterpillar* [35] is a web-based workflow engine for Ethereum that provides both compiler- and interpreter-based process execution mechanisms. It also offers user interfaces for designing and executing processes. The tool is publicly available,[12] and a screencast demonstrates its application.[13]

[11] https://drive.google.com/file/d/1rpy-oHbDVkXa6u4Fn73wSX8rINn1sv3U/view.

[12] https://github.com/orlenyslp/Caterpillar.

[13] https://youtu.be/zOmx5kD1Og0.

6.1.3 Analyze Process Behavior

While the smart contracts can record information about the state of a process instance and enforce business rules to coordinate the execution of activities in a collaborative process, there is still a need for real-time monitoring of process execution to react to deviations from the intended behavior. That is because rules at the process level might allow for some variability and smart contracts cannot influence the real-world context in which the processes and activities are carried out. For example, while the smart contract for the picking scenario can record the state of the transportation, it cannot ensure that the goods arrive in time due to having no control over the traffic conditions on the road. Moreover, it is important to analyze the overall process to identify bottlenecks and opportunities for process improvements in the supply chain context. For instance, the analysis might reveal that in many instances, the nominated driver differs from the driver that later wants to pick up the goods, indicating that the nomination needs to be improved, e.g., by also requiring the carrier to confirm the nomination of a truck driver.

In this context, an approach to tracing the execution of collaborative processes was presented in [36, 37]. This approach builds on top of compiler-based process execution and is intended to visualize the state of process instances. In particular, the approach monitors the blocks that are added to the blockchain for transactions sent to a factory contract. Data regarding process events are extracted from these transactions, including the executed activity, the process instance, and the block number at which the event occurred. These data are then used to visualize the state of individual instances and detect deviations from the expected behavior at the level of process instances, for example, to identify an unusually high number of driver re-nominations, possibly indicating an attack.

However, this approach is limited in the type of data used for analysis and supported analysis techniques, primarily focusing on basic instance monitoring. These points are addressed in a more general framework for process analysis [38, 39]. The proposed framework is independent of the approach used for process execution and can be applied independent of whether a compiler-based, an interpreter-based, or a conventional development approach was taken. It enables supply chain process analysts to declaratively describe the event data that must be extracted from the blockchain, not limiting analysts to transaction data but also supporting them in extracting data from other blockchain data structures, including blocks, log entries, and smart contracts. Moreover, analysts can establish a process notion different from the initially defined process model, allowing for analysis of different perspectives. For example, in addition to analyzing the picking order process as defined in Fig. 8, the framework can focus on the distribution center's scheduling activities and provide an overview of the scheduling process that spans all picking process instances. Additionally, the framework supports data formatting in standard file formats including text-based logs, CSV, and the extensible event stream format. This way, arbitrary tools and techniques can be employed for analysis.

In the context of blockchain-based process execution, techniques from process mining are especially relevant [38]. These techniques can provide insights at the

process level, e.g., by discovering the process model that explains the historic process behavior [40] or at the instance level by predicting the process behavior [41]. An instantiation of the framework for Ethereum-based applications and exemplary analysis scenarios is publicly available.[14]

6.2 Advanced Topics

This section provided an overview of basic approaches for enabling blockchain-based process execution in the context of supply chain management. Here, we complement this overview with pointers to work that introduce extensions to the presented approaches, primarily with regard to the enforcement of business rules.

With regard to the organizational perspective, the presented approaches broadly assumed that the role management is implemented independently of the smart contracts. To this end, an approach for dynamically binding actors to the executed process instances when needed is outlined in [42]. This is, for example, relevant when different carriers are chosen for picking up goods from the distribution center. Moreover, the approaches from this section also focused on models that capture processes with strict activity flows. Madsen et al. [43] propose an approach that relies on implementing smart contracts based on declarative process models. Such models do not specify the flow of a process from start to end, but capture the primary constraints and goals, allowing for more flexible process execution. Based on such an approach, the model from Fig. 8 might be reduced to the most relevant constraints, only specifying that a truck driver must be nominated before pickup or that the handover needs to be confirmed by both parties while providing parties with the opportunity to freely organize the remaining activities. The list of alternative modeling approaches includes approaches that put more emphasis on aspects related to communication and sharing of data. In this regard, Hull et al. [44] proposed a modeling language for defining data-aware and shared ledger-based processes, primarily focusing on aspects related to conceptual modeling and formal verification of such processes. Similarly, the modeling of BPMN choreography diagrams and the generation of smart contracts that implement those models are discussed in [45]. In contrast to collaboration diagrams, as, for example, shown in Fig. 8, choreography diagrams capture the sequence of interactions and communications between process participants. Finally, a discussion of open issues is presented in [46].

7 Summary

Blockchain technology has been proposed as a highly promising technology to address challenges arising from the complexities of modern supply chains. Con-

[14] https://github.com/ChrisKlinkmueller/Ethereum-Logging-Framework.

sequently, the industry has started to adopt blockchains, and numerous real-world implementations have been conducted or initiated in the last years. With that in mind, we outlined how blockchain solutions that support the management of supply chains can be developed in this chapter. In particular, we focused on five essential tasks. First, we provided guidelines for suitability analysis and discussed situations in which the application of blockchain technology can be beneficial to supply chain management. Second, we presented three basic architectures for blockchain-based supply chain management systems that enable balancing secondary properties like data confidentiality, costs, and performance isolation and that can be adapted to the specific supply chain context. Third, we summarized an approach to designing on-chain data stores that can be used to, for example, track and trace items or publish and manage certifications on the blockchain. Fourth, we demonstrated how off-chain components that enable the validation of on-chain data concerning the degree to which it represents the real-world state could be integrated into a blockchain-based system. Finally, we provided an overview of BPM tools adopted to enforce business processes and rules on the blockchain, coordinate supply chain participants, integrate the data flow into a process, and monitor and analyze processes to quickly react to unforeseen circumstances or to identify bottlenecks, among others.

References

1. M. Staples, S. Chen, S. Falamaki, A. Ponomarev, P. Rimba, A. B. Tran, I. Weber, X. Xu, J. Zhu, *Risks and Opportunities for Systems Using Blockchain and Smart Contracts*. Data61 (CSIRO) (2017)
2. K. Sundtoft Hald, A. Kinra, How the blockchain enables and constrains supply chain performance. Int. Journal of Physical Distribution and Logistics Management **49**(4), 376–397 (2019)
3. Hyperledger. How Walmart brought unprecedented transparency to the food supply chain with Hyperledger Fabric (2018). https://www.hyperledger.org/wp-content/uploads/2019/02/Hyperledger_CaseStudy_Walmart_Printable_V4.pdf
4. N. Popper, S. Lohr, Blockchain: A better way to track pork chops, bonds, bad peanut butter? (2017). https://www.nytimes.com/2017/03/04/business/dealbook/blockchain-ibm-bitcoin.html. Accessed: 29/09/2020
5. K. Hannam, This emerging tech company has put Asia's tuna on the blockchain. https://www.forbes.com/sites/keshiahannam/2016/09/30/this-emerging-tech-company-has-put-asias-tuna-on-the-blockchain/#562761602649 (2016), accessed: 29/09/2020
6. S. Higgins, Defense giant Lockheed Martin integrates blockchain. https://www.coindesk.com/defense-giant-lockheed-martin-integrates-blockchain (2017), accessed: 29/09/2020
7. R. Campbell, Modum.io's tempurature-tracking blockchain solution wins accolades at Kickstarter accelerator 2016. https://bitcoinmagazine.com/articles/modum-io-s-tempurature-tracking-blockchain-solution-wins-accolades-at-kickstarter-accelerator-1479162773 (2016), accessed: 29/09/2020
8. P. Gonczol, P. Katsikouli, L. Herskind, N. Dragoni, Blockchain implementations and use cases for supply chains—A survey. IEEE Access 8, 11856–11871 (2020)
9. H. Juma, K. Shaalan, I. Kamel, A survey on using blockchain in trade supply chain solutions. IEEE Access **7**, 184115–184132 (2019)

10. N. Kshetri, 1 blockchain's roles in meeting key supply chain management objectives. Int. Journal of Information Management **39**, 80–89 (2018)
11. X. Xu, I. Weber, M. Staples, *Architecture for Blockchain Applications* (Springer, Berlin, 2019)
12. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in *Business Process Management*, ed. by M. La Rosa, P. Loos, O. Pastor (Springer International Publishing, Cham, 2016), pp. 329–347
13. Q. Lu, X. Xu, Adaptable blockchain-based systems: A case study for product traceability. IEEE Soft. **34**, 21–27 (2017)
14. Q. Lu, M. Staples, H. O'Connor, S. Chen, A. Guabtni, Software architecture for blockchain-based trade certificate systems, in *2020 IEEE International of Conference on Blockchain and Cryptocurrency (ICBC)* (2020), pp. 1–3
15. Feng Tian, A supply chain traceability system for food safety based on haccp, blockchain Internet of things, in *2017 International Conference on Service Systems and Service Management* (2017), pp. 1–6
16. J. Benet, Protocol Labs. Interplanetary file system. https://ipfs.io/
17. I. Weber, Q. Lu, A.B. Tran, A. Deshmukh, M. Gorski, M. Strazds, A platform architecture for multi-tenant blockchain-based systems, in *2019 IEEE International Conference on Software Architecture (ICSA)* (2019), pp. 101–110
18. S. Malik, S.S. Kanhere, R. Jurdak, Productchain: Scalable blockchain framework to support provenance in supply chains, in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)* (2018), pp. 1–10
19. K.A. Schaefer, D. Scheitrum, K. Nes, International sourcing decisions in the wake of a food scandal. Food Policy **81**, 48–57 (2018). http://www.sciencedirect.com/science/article/pii/S0306919218305426
20. P. Downey, The characteristics of a register (2016), accessed: 31 July 2020. https://gds.blog.gov.uk/2015/10/13/the-characteristics-of-a-register
21. A.B. Tran, S. Xu, I. Weber, M. Staples, P. Rimba, Regerator: A registry generator for blockchain, in *29th International Conference on Advanced Information Systems Engineering, Forum Track* (2017), pp. 81–88
22. M. Fowler, *Domain Specific Languages*, 1st edn. (Addison-Wesley, New York, 2010)
23. Q. Lu, I. Weber, S. Mark, Why model-driven engineering fits the needs for blockchain application development, in *IEEE Blockchain Technical Briefs* (2018). https://blockchain.ieee.org/technicalbriefs/september-2018/why-model-driven-engineering-fits-the-needs-for-blockchain-application-development
24. S.K. Lo, X. Xu, C. Wang, I. Weber, P. Rimba, Q. Lu, M. Staples, Digital-physical parity for food fraud detection, in *Blockchain—ICBC 2019*, ed. by J. Joshi, S. Nepal, Q. Zhang, L.-J. Zhang (Springer International Publishing, New York, 2019), pp. 65–79
25. S.K. Lo, X. Xu, M. Staples, L. Yao, Reliability analysis for blockchain oracles. Comput. Electr. Eng. **83**, 106582 (2020)
26. R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: connecting blockchain to the off-chain world, in *Business Process Management: Blockchain and Robotic Process Automation Forum*, ed. by A. Asatiani, J.M. García, N. Helander, A. Jiménez-Ramírez, A. Koschmider, J. Mendling, G. Meroni, H.A. Reijers (Springer International Publishing, Cham, 2020), pp. 35–51
27. M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, *Fundamentals of Business Process Management* (Springer, Berlin, 2018)
28. L. García-Bañuelos, A. Ponomarev, M. Dumas, I. Weber, Optimized execution of business processes on blockchain, in *Business Process Management*, ed. by J. Carmona, G. Engels, A. Kumar (Springer International Publishing, Cham, 2017), pp. 130–146
29. O. López-Pintado, M. Dumas, L. García-Bañuelos, I. Weber, Interpreted execution of business process models on blockchain, in *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)* (2019), pp. 206–215
30. T. Perera, S.P.T. Hettige, S.A.P. Jayatilake, W. Abeygunawardhana, H.M.N.D. Bandara, Blockchain-based product traceability beyond proof of existence, in *2020 Moratuwa Engineering Research Conference (MERCon)* (2020), pp. 1–6

31. Q. Lu, A.B. Tran, I. Weber, H. O'Connor, P. Rimba, X. Xu, M. Staples, L. Zhu, R. Jeffery, *Integrated Model-driven Engineering of Blockchain Applications for Business Processes and Asset Management* (2020)
32. O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the ethereum blockchain. Softw.: Pract. Exper. **49**(7), 1162–1193 (2019)
33. C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. Tran, I. Weber, Blockchain support for collaborative business process. Informatik Spektrum **42**, 182–190 (2019)
34. A.B. Tran, Q. Lu, I. Weber, Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management, in *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9–14, 2018*, ser. CEUR Workshop Proceedings, ed. by W.M.P. van der Aalst, F. Casati, R. Conforti, M. de Leoni, M. Dumas, A. Kumar, J. Mendling, S. Nepal, B.T. Pentland, B. Weber, vol. 2196 (2018), pp. 56–60
35. O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, Caterpillar: A blockchain-based business process management system, in *BPM'17* (Springer, Barcelona, 2017), pp. 1–5
36. C. Di Ciccio, A. Cecconi, J. Mendling, D. Felix, D. Haas, D. Lilek, F. Riel, A. Rumpl, P. Uhlig, Blockchain-based traceability of inter-organisational business processes, in *Business Modeling and Software Design*, ed. by B. Shishkov (2018), pp. 56–68
37. R. Mühlberger, S. Bachhofner, C. Di Ciccio, L. García-Bañuelos, O. López-Pintado, Extracting event logs for process mining from data stored on the blockchain, in *Business Process Management Workshops*, ed. by C. Di Francescomarino, R. Dijkman, U. Zdun (Springer International Publishing, Cham, 2019), pp. 690–703
38. C. Klinkmüller, A. Ponomarev, A.B. Tran, I. Weber, W. van der Aalst, Mining blockchain processes: Extracting process mining data from blockchain applications, in *Business Process Management: Blockchain and Central and Eastern Europe Forum*, ed. by C. Di Ciccio, R. Gabryelczyk, L. García-Bañuelos, T. Hernaus, R. Hull, M. Indihar Štemberger, A. Kő, M. Staples (Springer International Publishing, Cham, 2019), pp. 71–86
39. C. Klinkmüller, I. Weber, A. Ponomarev, A.B. Tran, W. van der Aalst, Efficient logging for blockchain applications. ArXiv, vol. abs/2001.10281 (2020)
40. W. van der Aalst, *Process Mining: Data Science in Action* (Springer, Berlin, 2016)
41. C. Klinkmüller, N.R.T.P. van Beest, I. Weber, Towards reliable predictive process monitoring, in *Information Systems in the Big Data Era*, ed. by J. Mendling, H. Mouratidis (2018), pp. 163–181
42. O. López-Pintado, M. Dumas, L. García-Bañuelos, I. Weber, Dynamic role binding in blockchain-based collaborative business processes, in *Advanced Information Systems Engineering* (2019), pp. 399–414
43. M.F. Madsen, M. Gaub, T. Høgnason, M.E. Kirkbro, T. Slaats, S. Debois, Collaboration among adversaries: Distributed workflow execution on a blockchain, in *Symposium on Foundations and Applications of Blockchain* (2018)
44. R. Hull, V.S. Batra, Y.-M. Chen, A. Deutsch, F.F.T. Heath III, V. Vianu, Towards a shared ledger business collaboration language based on data-aware processes, in *Service-Oriented Computing*, ed. by Q.Z. Sheng, E. Stroulia, S. Tata, S. Bhiri (2016), pp. 18–36
45. J. Ladleif, M. Weske, I. Weber, Modeling and enforcing blockchain-based choreographies, in *Business Process Management*, ed. by T. Hildebrandt, B.F. van Dongen, M. Röglinger, J. Mendling (Springer International Publishing, Cham, 2019), pp. 69–85
46. J. Mendling, I. Weber, W.V.D. Aalst, J.V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C.D. Ciccio, M. Dumas, S. Dustdar, A. Gal, L. García-Bañuelos, G. Governatori, R. Hull, M.L. Rosa, H. Leopold, F. Leymann, J. Recker, M. Reichert, H.A. Reijers, S. Rinderle-Ma, A. Solti, M. Rosemann, S. Schulte, M.P. Singh, T. Slaats, M. Staples, B. Weber, M. Weidlich, M. Weske, X. Xu, L. Zhu, Blockchains for business process management—Challenges and opportunities. ACM Trans. Manage. Inf. Syst. **9**(1), 1–16 (2018)

Blockchain Technology for E-Governance Applications

Ras Dwivedi, Nilesh Vasita, Mukul Verma, Tanmay Yadav, and Sandeep Shukla

1 Introduction to Blockchain Technology in E-Governance

One of the cornerstones of better governance is the interaction between the government and the citizens. Simple, convenient, and effective interaction between government and citizens has been a common expectation in society. While, on the one hand, there is a demand for an increase in transparency and fairness, on the other hand, there are concerns for privacy and accountability [1]. Blockchains meet both the world. Blockchain, by itself, brings integrity and transparency to the system [2]. Armed with smart contracts, it brings accountability and fairness to the system [3]. Using state-of-the-art cryptography like zero-knowledge proof or private transactions, one can also ensure privacy [4]. For example, one can enhance location privacy in location-based services using zero-knowledge proofs [5].

The smart contract enables automated decision-making at the national level, thereby increasing the efficiency of governance and bringing fairness [6]. Blockchain brings decentralization in decision-making. Hence, multiple organizations can share their data, automate their decision-making process, and bring different departments in sync.

Government systems are prone to insider attacks and manipulation of data. Hence, there is a need for an immutable data structure, such as one provided by the blockchain [7]. Because the data itself or some integrity measure on the data are distributed and decentralized, no single entity has the authority to manipulate the

R. Dwivedi · T. Yadav · S. Shukla (✉)
Indian Institute of Technology Kanpur, Kanpur, India
e-mail: dwivedi@cse.iitk.ac.in; tanmayy@iitk.ac.in; sandeeps@cse.iitk.ac.in

N. Vasita · M. Verma
CRUBN Foundation, Kanpur, India
e-mail: nilesh@crubn.com; mukul@crubn.com

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_13

system illegally. Various governments across the world have piloted various projects using blockchain in E-Governance.

- Estonia is using blockchain to keep government records on blockchain and make them immutable [8, 9]. Most of its services are online, and it has one of the fastest court proceedings in Europe. Estonia chose to move on the blockchain after the 2017 cyber-attack [10].
- Canada uses blockchain to issue digital CVs, providing a permanent, self-owned, and secure record for the skills [11].
- Mexico is planning to conduct public procurement using blockchain [12]. It also intends to conduct government tenders using blockchain [13].
- US Air Force has implemented a blockchain-based supply chain network [14]. US Navy is planning to use blockchain to track aviation parts throughout its life cycle [15]. US postal service is using blockchain to authenticate its users [16]. FDA is also exploring the utility of blockchain in tracking drugs [17].
- Argentina has announced blockchain-based identity management to improve citizens' access to government [18].
- The Republic of Georgia was among the first countries to adopt blockchain to record land registries [19]. The average land registration time in Georgia is about 3 minutes.
- Germany has set up the blockchain lab to unleash blockchain's potential in achieving the United Nations' Sustainable Development Goals [20].
- In Spain, Catalonia's government launched a self-sovereign identity project to allow the citizen to manage their identity [21].
- Sweden is testing real estate and other multi-party transactions on the blockchain [22, 23].
- In the United Kingdom, her majesty's Land Registry set up a digital asset management platform to explore blockchain's potential in the UK's real estate industry [24].
- In China, the Management Committee of Xiong'an New District announced that blockchain would be used to establish a clean, transparent, and efficient government through efficient supervision [25].

One of the convincing argument to put the citizen data on a blockchain or at least record the transactions on citizen data on a blockchain is that it enhances citizen's trust on the safe keeping and integrity of their data and ability to audit who might have accessed or changed that data. In a democracy, trust of citizens on government's handling of citizen's data (such as identity, land records, tax records, medical data, etc.) is extremely important. All these pilots by various governments bear witness to the fact that governments are slowing understanding the concern of citizens in terms of privacy and integrity of their data and auditability seriously. Therefore, E-Governance is an apt target for blockchain technology to demonstrate its importance in modern democracies.

2 Work at IIT Kanpur's National Blockchain Project

The National Blockchain Project (NBP) at IITK is sponsored by the National Security Council Secretariat, Government of India. It has the mandate to research and develop blockchain-based solutions fit for Indian scenarios. NBP has been analyzing different blockchains and their potential to be used in India.

Some of the work done at NBP is described in the subsequent subsections. Starting from creating a self-sovereign identity management system, where an individual is in complete control of his data to developing a blockchain-based land records maintenance system that is fit to be applied in most Indian states, the work done at NBP has been primarily been focused around E-Governance scenarios in India. It has also created a new-age digital health ecosystem that allows patients to securely share their data and maintain their privacy by using state-of-the-art cryptographic techniques to ensure that the patient's data is never leaked and is only shared with the entities he wants. In addition, a blockchain-based consent manager that records every consent user gave to share their data is also being developed at NBP.

2.1 Self-Sovereign Identity of Citizens

2.1.1 Introduction

Today, to enter a bar or apply for a bank loan, an individual has to prove specific attributes about his identity to the service provider to get access to these services. For this, people use state-issued identity documents and credentials. They need to carry these documents physically at every point where verification of their identity is needed. They are often asked to submit a scanned copy or a photocopy of these documents, which ends up revealing more information about the individual than the verifier needs to know. It might result in identity theft and misuse, giving rise to fraudulent and criminal activities.

With the rise of Internet service providers and smartphone advancements, individuals' online activities have also increased. It allows the large tech giants to collect humongous data generated by these individuals' activities. Although there are rules on how the user data would be shared, user is hardly in control on how this data is to be used.

In the sections to follow, we will describe what are the current challenges related to Identity and Credential Management, what is self-sovereign identity, how does it solve these challenges, and what are the prospective areas and use cases where it can create a significant impact.

2.1.2 Problems with Current Identity Solutions

Identity and the related credentials can be divided into two categories: first is the physical credentials, i.e., the paper-based credentials like Aadhaar, driving license, degree, etc., whereas the other one is the identity of an individual on the Internet, i.e., his digital identity. Some of the notable challenges with physical, paper-based credentials are as follows:

- Inefficient credential obtaining processes because of the bureaucratic red-tapes
- Lack of secure digital counterparts of physical credentials and documents
- Identity theft and impersonation
- Unauthorized use of credentials

The online identity of an individual is mostly derived from the username password pairs used by an individual over the Internet. Some of the notable challenges with digital identity are the following:

- Need of different login credentials for each website in the form of multiple username passwords results in selection of weak and easy to remember passwords which creates a security threat to users.
- Use of same username and passwords across multiple website increases the attack surface for hackers.
- No privacy is preserved while using third-party logins. Third-party providers get to know additional information about users like which online services they are using, for how much time they are using it, and how often.
- Use of common identifiers across many websites opens up the possibility of correlation of data without user consent.
- Fake online IDs on social media platforms can be created with ease.
- Large repositories of identity data become the favorite targets of hackers.

2.1.3 Self-Sovereign Identity

As described in a report [26] from the World Economic Forum, "A self-sovereign identity is 'owned' by the individual. As owner, the individual has access to, can refer to and share components of this identity at their discretion. While certain components of the identity are established by issuing authorities (i.e. passport number, bank details), the individual must consent to the sharing of their identities and any related data. This is achieved by individuals securely storing their own identity data on their own personal devices and providing it efficiently to those who need to validate it, without relying on a central repository of identity data."

The term self-sovereign identity was first used by Christopher Allen in his paper *The Path to Self-Sovereign Identity* [27] where he describes about the evolution of identity and how and why the concept of self-sovereign identity emerged out. He talks about the four stages of identity that the Internet has advanced through since its inception.

Fig. 1 An example DID [29]

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Listing 1: A sample DID Document [28]

2.1.4 Decentralized Identifier (DID)

Decentralized identifiers (DIDs) [28] are a new type of globally unique identifier designed to enable individuals and organizations to generate their own identifiers using systems they trust and to prove control of these identifiers (authenticate) using cryptographic proofs (e.g., digital signatures, privacy-preserving biometric protocols, and so on) (Fig. 1).

DIDs resolve to DID Documents, like the one shown in Listing 1, which are the resources that give information about the public keys, service endpoints, and authentication protocols needed to interact with the owner of the DID. DIDs act as a crucial medium for managing connections with other entities in a self-sovereign identity system. In a DID infrastructure, the DID Documents are values of the keys defined by the DIDs in a global decentralized key value store.

In the SSI ecosystem described in Sect. 2.1.6, the issuers publish **public DIDs** on the ledger for anyone to access them. It enables any entity in the network to resolve the DID and know all the public details related to the issuer using the resolved DID Document. The public key in the DID Document and the digital signature in an issued credential are used to ensure that the credential is valid and has not been tampered with. One can also cryptographically verify that a credential was issued to a holder and that it has not been revoked.

Private DIDs, on the other hand, need not be written on the ledger, hence increasing the scalability of the overall system. They help in establishing a secure and private channel between two parties to exchange messages. The channel is established by both the parties sharing their DID and the corresponding DID

document with each other. The use of private DIDs also solves the problem of two-way verification on the Internet. Since both parties have exchanged verifiable information with each other, both parties can be assured that they are interacting with the entity they intended to interact with. The pair-wise private DIDs are resolved through the interaction between the communicating entities rather than through the blockchain.

2.1.5 Verifiable Credentials

Verifiable credentials [30] can be considered as the digital counterpart of the credentials like driver's licenses, transcripts, medical certificates, etc. with added cryptographic features like digital signatures which make them tamper-proof, secure, and verifiable. The crucial properties of a verifiable credential are as follows:

1. It can be used to prove the issuer of the credential.
2. It can be used to prove that the credential was issued to a specific holder.
3. It can be used to prove that the credential has not been tampered with.
4. It can be used to prove that the issuer has not revoked the credential.

The current ways of transmitting credentials via the Internet make it difficult to make it verifiable yet protecting individual privacy. The Verifiable Credentials ecosystem is composed of the following major roles:

- **Issuer**: The entity that issues verifiable credentials from claims asserted about one or more subjects and transmits them to a holder.
- **Holder**: The entity that securely stores verifiable credentials and presents verifiable presentations on demand of a verifier.
- **Verifier**: The entity that receives the verifiable presentations from the holder and verifies that the credentials inside these presentations are fit for purpose.
- **Identifier Registry**: The system that mediates the creation and verification of identifiers, keys, and other relevant data that might be required to use verifiable credentials. Trusted databases, decentralized databases, and distributed ledgers can be considered as examples of verifiable data registries.

A pictorial representation of the credential relationship is depicted in Fig. 2.

2.1.6 The SSI Ecosystem

As can be seen in Fig. 3, the SSI Ecosystem primarily consists of a public permissioned blockchain [31] and entities with the roles of Trustees, Stewards, Issuers, Verifiers, and Holders. Trustees and Stewards are the highly privileged roles in the network responsible primarily for the governance, functioning, and maintenance of the network. To issue credentials, an Issuer must first get itself

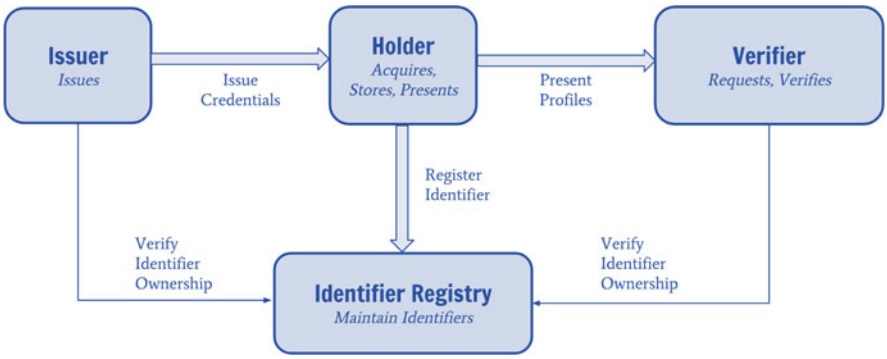


Fig. 2 Credential relationships [30]

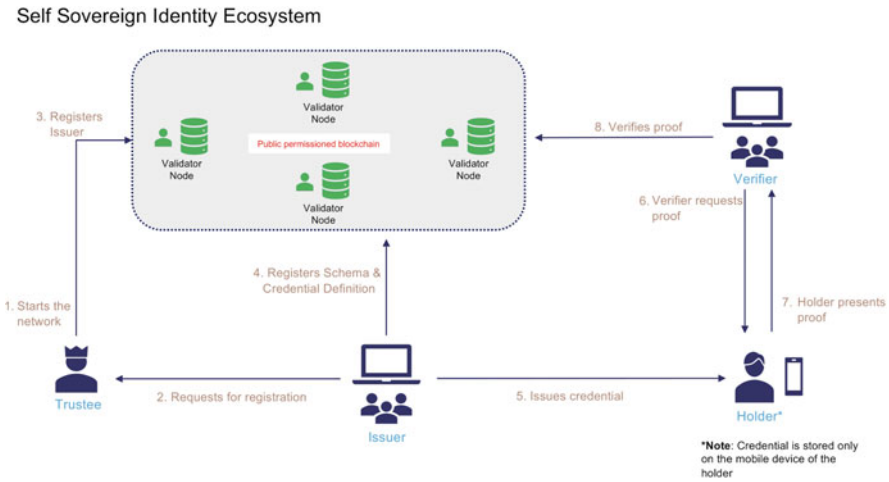


Fig. 3 SSI ecosystem

registered on the blockchain. Stewards and Trustees can register Issuers on the blockchain after doing the necessary due-diligence. After registration, the DID and the related DID Document of the Issuer can be checked on the blockchain by anyone in the ecosystem. The Issuers in the ecosystem could be entities from any of the sectors starting from Healthcare, Education, and Governance to Banking and Finance. All they need to do before issuing the credentials is to publish the Schema (containing the attributes in a credential) and Credential Definition (cryptographic details specific to an Issuer and a Schema) on the ledger along with a Revocation Registry, which can be later on used by Verifiers to verify the authenticity and non-revocation of credentials used by Holders to send them proofs. Issuers can then issue credentials to Holders who can store these credentials on their mobile device or any other personal device for later use.

2.1.7 Use Cases

In all the use cases, we assume that the Holder would be using his digital wallet on his mobile device for interactions with Issuers and Verifiers. Below are some use cases and scenarios in which presence of SSI ecosystem can bring a lot of value:

Employment Document Verification

Problem Description Whenever an individual joins a new company, the employer is interested in verifying all the details about the newly hired employee. Currently, the new hires are requested with a list of documents that they need to supply to employer for the verification of these credentials. All these credentials are mostly provided to the employer in the form of copies of the physical document along with the original copies at times. Most of the times, the only option for the employer to verify such credentials is to manually check the inked signatures of the Issuing Authorities on these credentials.

Solution Using the SSI ecosystem, the employer has to simply generate a proof request containing all the credentials and the attributes associated with these credentials that the employer is interested in to verify about the new hire. The new hire will receive the proof request on his mobile application. He can create a proof presentation for this request using the credentials that were issued to him by the Issuing Authorities in a verifiable and privacy preserving manner.

Availing Public Transport Services

Problem Description Year 2020 has witnessed the global pandemic COVID-19 which has created huge disruptions in the travel industry. Once a vaccination is rolled out, individuals will need to prove at certain points in their journey that they are vaccinated. This could be achieved by issuing cards or certificates to the individuals who are vaccinated. The use of cards or certificates comes with their own inefficiencies; hence a solution is needed to efficiently verify if an individual is vaccinated.

Solution The organization that does the vaccination can issue a digital credential to a holder in his mobile wallet which can be later on used by the holder at all the points in the journey where a proof of vaccination is required. Since it would be a verifiable credential, there would be no chances of any tampering with the credential, and the verification would be efficient as well.

2.2 Land Registration

2.2.1 Introduction

Land as an asset is unique mainly because it is immovable and fixed in supply. With increasing population and urbanization, demand keeps on increasing and thereby increasing its value. A conclusive land record system, well integrated with other agents like banks, can be a boost to the economy. Nevertheless, land records, in countries like India, are poorly maintained, making it one of the most litigations and controversial subjects. The lack of a proper system to maintain land records and the system's inability to provide conclusive land titling has resulted in long-drawn legal conflicts. Around 25% of the cases decided by the Supreme Court are related to the land dispute, with 30% of them related to acquisition. Surveys show that 66% of all civil cases in India are related to land or property dispute [32]. The average dependency of land related cases in India is about 20 years [32].

The land titling system in India is presumptive. More than 20 documents could be shown to claim land ownership [33], but none of them act as a government guarantee for the land title. Two most commonly used documents are registered sales deed and record of rights, which are maintained by two different departments that work in silos. Although registration has to be followed by mutation, not everybody does that, mainly because of the terribly slow process and because the situation on the ground might not mirror the actual records.

A mutation has to be preceded by registration, but this is not mandatory in all cases, for example, no registration is necessary in the case of undisputed succession. As a result, one has a parallel system of land records, with different claims, and none of them guarantying the claim to the title.

2.2.2 Problems with the Current System

Presumptive land titling, coupled with the multiplicity of the records, created a system which has the following problems:

- Land ownership is presumptive. The onus of proving ownership lies on citizens and not the government. This usually leads to a large number of court cases.
- Different departments work in silos, and land ownership from one may not match with the other. Worse, the situation on the ground might be completely different.
- Due to manual records, the process of registration and subsequent mutation is slow. For each transaction, buyers and sellers have to run from one department to another. Since the land records were presumptive, land title transfer could not be instantaneous. Usually, registration and mutation are separated by a month to allow a third party to put forward a claim.
- Manual records are error-prone. Apart from human errors, corrupt officers could deliberately introduce such errors. Removing such errors again is a painstaking task.

- Verification of the records is time-consuming.
- Mismatch between the status on the ground and the land record with the citizens.
- Since land records were always presumptive, access to credit with land as collateral is difficult.

The land record modernization program was started by the government of India in 2008, which led to the digitization of the land record. It only solved one aspect of the problem. Various departments still work in silos, and hence the following problems are still there:

- Land ownership still presumptive.
- Every mutation is not preceded by registration, creating two separate legally valid systems of records.
- Information of land ownership at various departments is not updated concurrently.
- Banks are still not linked with the land records system, hence no ease of access to agricultural credit.
- No reduction in the number of court cases based on the land.

2.2.3 How Blockchain Can Help?

Blockchain is a cryptographically secure decentralized distributed ledger. The ledger is append-only and is jointly maintained by all the stakeholders. Also, every stakeholder verifies each entry in the ledger. Use of blockchain for land records brings the following advantages:

1. Blockchain creates immutable records, which means that errors could not creep into the system once the owner verifies the records.
2. Blockchain allows different departments to own their records and still maintain synchronicity using a smart-contract. Smart-contract can fill the gap that leads to different versions of the land records.
3. Blockchain makes registration and mutation faster. Since all the departments share an identical view of the records, the third-party claims vanish, and the land transfer process becomes instantaneous.
4. Proving ownership using blockchain is fast and final. It provides a guarantee to the land title.
5. Blockchain allows the database to be publicly verifiable. The verification process in India still involves a manual component. Verifying officers validates the records and then digitally signs it [34] Blockchain completely automatizes the process. Land records are viewable by the public without any latency. Everything on the blockchain is verifiable and public. This brings transparency to the system, which helps a lot in improving efficiency.
6. A publicly verifiable, traceable, fast, and final system of records would also reduce the number of court cases.
7. Blockchain allows tokenization of a property and makes land tradable like equity in the share market.

2.2.4 IIT Kanpur Blockchain for Land Records

IITK, under the aegis of the national blockchain project, designed a blockchain-based system for the registration and maintenance of the land record in India. In India, land is a state subject, and the land records maintenance differs slightly from state to state. We have used the land records management system as followed in the state of Uttar Pradesh. In Uttar Pradesh, land-related documents are primarily owned by three organizations.

1. **Board of Revenue:** It is responsible for maintaining khatauni, which is the primary document to establish land ownership. Board of revenue also maintains the agricultural records of the land.
2. **Department of Stamp and Registration:** This department is concerned with the collection of revenue and stamp duty of a land. As per Stamp and Registration Act of 1908, every transaction regarding land transfer must be registered at this department.
3. **Department of Survey and Settlement:** This department is concerned with the survey of land and change of land use pattern. It maintains the certified copies of village maps, periodically updates land records as per situation on the ground, and undertakes the land consolidation process.

We used a permissioned blockchain, Hyperledger Fabric [35]. The different departments were modeled as the organization in Hyperledger fabric. Each department was given the read and write access as per the existing law. Because the ledger is shared, every department views the same version of the land records. We used a three-organization network with single channel as shown in Fig. 4.

We implemented our model in Hyperledger fabric version 1.4. All the existing records were uploaded into the system as initial records. A land transfer request

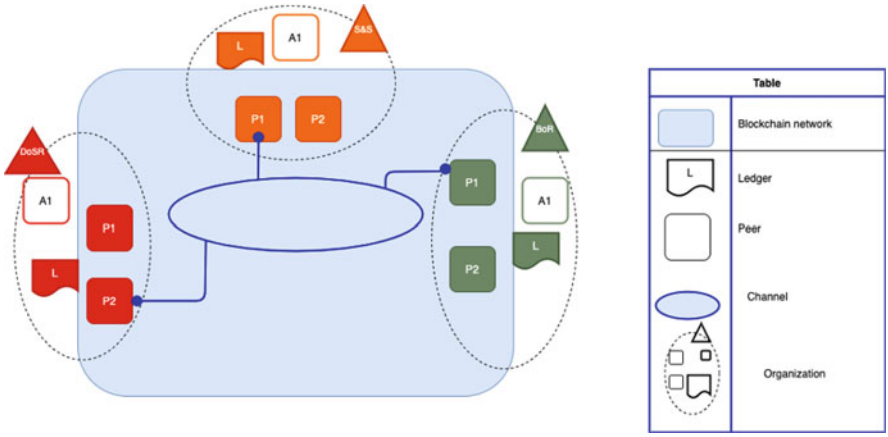


Fig. 4 IIT Kanpur blockchain network has three organizations operating on shared ledger. All organizations are part of a single channel

begins with the registration. As soon as the registration requests are accepted and pushed into the blockchain, the land records are marked as under transfer, and no new registration or mutation request for that record could be initiated. Since the blockchain synchronizes all the departments, the buyer does not need to create a mutation request separately. The mutation request is automatically sent to the board of revenue. Since the request is based on the blockchain records, officers need not to verify them separately. This reduces the verification time, and land records mutation becomes a faster process.

The concerned officer then accepts and endorses the mutation request, and then the records are updated in real time. Records could then be easily verified. In our implementation, land records verification does not need manual intervention.

The Department of Survey and Settlement, which is inactive in most of the land transfer cases, also have their records updated in real time, significantly reducing future survey works.

2.2.5 Examples Across the World

Even though blockchain is barely a decade-old technology, it has been widely accepted. We present here a few examples across the word where the blockchain technology has been adopted.

Sweden

- **Blockchain type:** Permissioned
- **Partnerships:** Cromaway, Kairos Future, and Telia Company [36]
- **Key benefits:**
  - Reduce errors: Manual registration is prone to errors, and unintentional or malicious error always creeps in the records. Blockchain reduces such errors.
  - Optimize efforts: The land registration department is involved at the very end of the real transaction. So if something is wrong, much effort is wasted. Blockchain would ensure that the land registration department is in the loop from the beginning of the process.
  - Reduce time: Time between signing a legally binding purchasing contract and when Lantmäteriet (Sweden's land registration department) receives the bill of sale and makes the approval of the title is often 3 to 6 months. Blockchain could significantly reduce this time interval.
  - Improving ease of doing business: Because of the time-consuming processes and the high value of the property, every agent (seller, buyer, banks, and real estate agents) has created their own complex red tape process for agreements between them. Blockchain would help in reducing red tapism.

England

- **Blockchain type:** Permissioned
- **Partnerships:** Methods, R3, Persistent Systems, and Blockchain Digital [37]
- **Key benefits:**
  - Digitally signed property transaction and digital mortgage.
  - Blockchain being immutable, allows one to find correct information about the property.
  - Blockchain reduces the time and cost of doing a transaction.
  - Blockchain creates an environment that is less error-prone and hence increases trust and transparency.

Ghana

- **Blockchain type:** Permissioned
- **Partnerships:** BenBen [38]
- **Key benefits:**
  - Ghana had a very poor land record system, to the extent that it was very difficult for the court to accept a valid land registration certificate. BigChain Db was then deployed. It is a permissioned blockchain that uses byzantine fault-tolerant consensus.
  - It is hoped that blockchain-based land registries would eliminate corruption and make land records enforceable in a court of law.
  - Blockchain would bring traceability and speed up the land transfer process.

Dubai

- **Blockchain type:** Public
- **Partnerships:** Consensys
- **Key benefits:**
  - Dubai aims at placing all the government records on the blockchain. This would make the records immutable and publicly verifiable.
  - Dubai aims to tokenize land and enable trade in land tokens.
  - Dubai aims to allow fractional ownership of the land, and this allows buyers to be a part owner of the property [39].

2.3 Healthcare Applications

2.3.1 Introduction

The present state of healthcare infrastructure has some severe shortcomings, which include but are not limited to inefficient management of medical records, lack of immediate availability of medical history, fraudulent prescriptions, medical certificates and medicines, and no system to collect, analyze, secure, and exchange data seamlessly [40].

Through various schemes and initiatives, the Government of India has taken steps to lay the foundation of a twenty-first-century health system. It is expected that the provision of services through public and private sectors under these schemes and initiatives will generate enormous amounts of health data, mostly in the digital space. To ensure that cutting-edge digital technologies are leveraged, it is crucial to create an appropriate architecture and data structures at a national level. With the current fragmented data capture system by multiple stakeholders without any standardization, there is a serious risk of compartmentalizing digital health assets.

To avoid such compartmentalization of health data, a digital health information ecosystem under the National Digital Health Mission [41] is being developed in India that incorporates the following features:

- Patient Unique Identifier that identifies a patient.
- Ensure the confidentiality, integrity, and availability of all the e-Personal Health Information.
- The system maintains a complete audit trail of change and access.
- The patient owns all protected health information contained in the EHR and should have sufficient privileges to:
  - Inspect and view their medical records without any time limit
  - Restrict access to and disclosure of individually identifiable health information
  - Need to provide explicit consent, which will be audited, to allow access and/or disclosures
- Ensure portability of these e-Personal Health Records between different stakeholders

Blockchain, being a technology that brings disintermediation of trust, provides a shared and immutable ledger that guarantees data integrity and improves process efficiency and offers the potential to transform the National Digital Health Ecosystem, making it more patient-centric and trustworthy. The National Digital Health Mission can also be an excellent stage to bring rapid mainstream adoption of blockchain technology.

In the next few sections, we will explore some key areas where blockchain can benefit the National Digital Health Ecosystem.

2.3.2 Decentralized Consent Management

The goal of the Consent Management Service is to ensure that the data principal (user/patient) is in complete control of what data is collected, and how/with whom it is shared and for what purpose, and how it is processed. The service applies to data relating to the entire Electronic Health Record, both longitudinal (over a period of time) and vertical (relating to an episode).

The various entities that are involved in the process are the following:

1. Information provider: an entity that collects or generates digital information and stores it in a software system.
2. Information user: an entity that seeks digital information from information providers to provide services to the user.
3. Consent manager: an entity that acts as a consent collector for the user and mediates information flow from information providers to information users.
4. User: any individual who is availing the services of an information provider or information user. User data is stored with information providers.

Consent management is a critical component of the National Digital Health Ecosystem. Thus, trust in the consent life cycle management process is of utmost importance as it handles the permissions related to the user's health data. The proposed architecture, although workable, takes away the control from the user (patient) and puts too much trust in the consent manager, which handles all the consent management processes. Another drawback is that information users, information providers, and the consent managers use an intermediary to communicate with each other, making the process inefficient.

To overcome the above challenges, a decentralized consent management service can be developed through permissioned blockchain technology, which will make the process more trustworthy, user-centric, and efficient, as shown in Fig. 5. The consent life cycle can be managed through smart contracts deployed on the consent management blockchain network. The user (patient) can deploy self-configured private contracts that can be used for managing consent privately among the involved entities, and public contracts can be used for timestamping and integrity checks.

The benefits of a decentralized consent management service are as follows:

- User-centric process.
- The consent life cycle management process is and not under the control of any entity.
- Implementation of these processes is agreed upon by all involved entities.
- Increased efficiency over the existing process because of disintermediation.
- Increased trust in the ecosystem.
- Immutable log of consent history maintained.

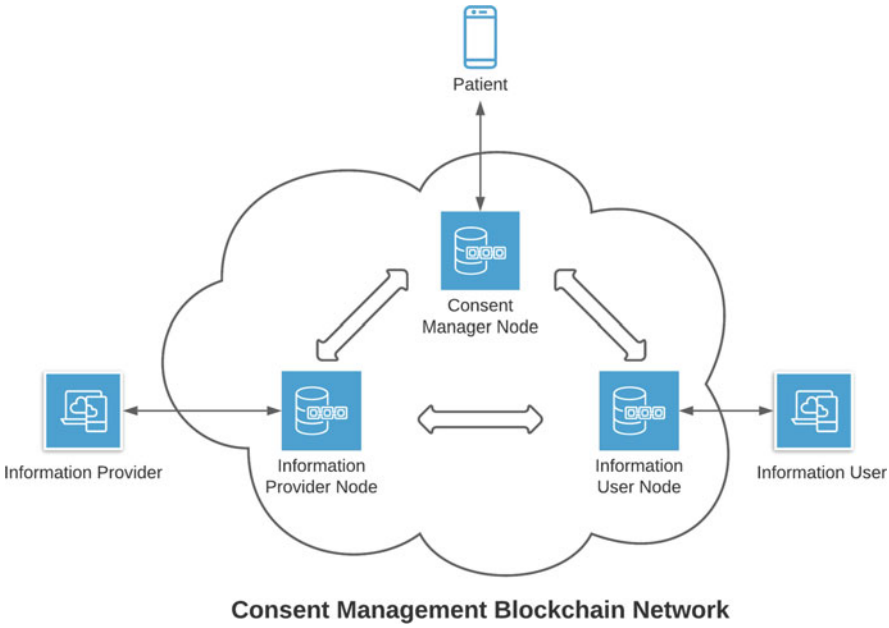


Fig. 5 Decentralized consent management

2.3.3 Blockchain as a Trust (Integrity) Layer

One of the core principles of the National Digital Health Blueprint is that the information created by the entities in the ecosystem should be trustworthy. Still, there is no system in the proposed architecture that actively makes the entities comply with this requirement except for the rules and regulations. So, an information provider with malicious intent can, at any point in time, change the health data of the patient, which can severely affect the future diagnosis and health of the patient.

Blockchain technology can help rectify this vulnerability. A permissioned blockchain (public or private) can be set up as a trust layer for this ecosystem. Although uploading all health-related data to any blockchain would be wasteful and inefficient, the hash (digital fingerprint) of every data point, like diagnosis, prescriptions, consent, etc., along with the additional metadata, can be pushed to this trust layer. This ensures that if any data is maliciously tampered by any entity in this ecosystem, it will be flagged as the on-chain hash will not match with the hash calculated when this data is fetched, hence making the ecosystem more trustworthy. This trust layer can also be used to record and timestamp each instance of data access, making the ecosystem auditable. Figure 6 shows how blockchain act as a trust layer between the information provider, information user, and the consent manager.

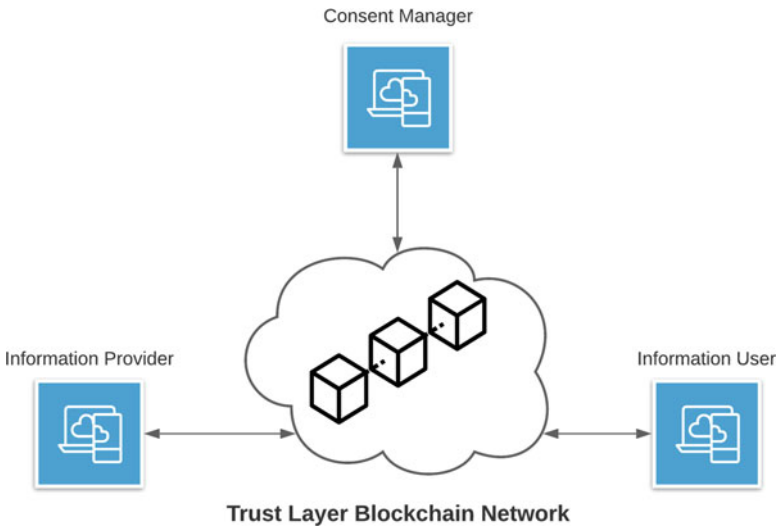


Fig. 6 Blockchain as a trust layer

2.3.4 Credentials for Health Staff

The health ecosystem of a country provides a number of facilities to the medical staff, especially the doctors. These services require the doctors to prove at various points that they hold a legit degree from affiliated colleges. As mentioned in Sect. 2.1.6, and elaborated with example in Sect. 2.1.7, issuing authorities like various medical councils, boards, and hospitals, can issue verifiable credentials to the doctors, which can be later used by the doctors to prove at various places the authenticity of their associations and education.

2.4 Blockchains in Supply Chain

2.4.1 Introduction

Supply chains are a country's lifelines, establishing a network between companies and suppliers of raw material and distributing the final product to the customers. According to the World Economic Forum report [42], reducing supply chain barriers to trade could increase GDP by nearly 5 percent and international trade by 15 percent. In India, supply chain costs represent as much as 13 percent of the gross domestic product (GDP) [43], almost double the percentage in developed countries. A typical supply chain usually includes producers, vendors, central and state warehouses, logistics companies, and retailers, and trusting the integrity of each other is a significant challenge.

In the last 10 years, India saw significant growth in the flow of goods and services to consumers spread all over the country. The country's supply chain has become more complex over time, hampering the procurement time of consignments. In India, companies are currently using point-to-point communication systems that consist of electronic data interchange (EDI) and XML messaging protocol to maintain the flow of information. This point-to-point communication mechanism leads to poor forecasting accuracy, which is the biggest challenge faced by logistics organizations. Also, because of poor infrastructure, the weak distribution system, and lack of technology adoption, most organizations asserted that they lack end-to-end supply chain visibility.

2.4.2 Problems with Current System

Complex interconnected, interlinked channels and nodes, with a lack of technology adaptation, lead to the following problems in supply chain management:

- Inefficient customer collaboration as the information is present in silos, and usually there is only one-up and one-down visibility.
- Poor forecast accuracy in existing supply chains leads to a lack of trust and customer dissatisfaction.
- Poor consensus among entities as there is no end-to-end visibility leading to mismatch in demand and supply plans which in turn causes clogging of channels and wastage of resources.
- Lack of trusted, easily auditable records brings in compliance overheads, which increases reconciliation time which makes the supply chains inefficient.
- Since databases are maintained centrally, there are chances of data manipulation, and privacy is compromised. Businesses prefer silos as a lot of critical data related to a particular trade can be exposed.

2.4.3 How Blockchain Can Add Value?

Blockchain can help alleviate many frictions in global trade logistics by leveraging existing standards and enabling all stakeholders to communicate in a trusted manner. Key areas where blockchain can add value to supply chain logistics are as follows:

1. Blockchain being an immutable ledger will instill trust among retailers, suppliers, and consumers and drastically reduce the procurement time.
2. Current blockchain technology has a feature of self-executable smart contracts, which will automate and reduce manual certificate management.
3. Verifiable smart contracts enhance ability to meet compliance standards.
4. Blockchain provides efficient data provenance and maintains the data integrity and, hence, caters to the issue of lack of end-to-end visibility, thereby assuring customers' goods supplied are authentic.

Fig. 7 Source: IBM Food Trust blockchain

5. Blockchain eliminates lots of manual paperwork, therefore reducing fraudulent certificates and increasing renewal speed.
6. In critical supply chains, such as food supply chains, integration of smart contracts with IoTs enables real-time data monitoring and improves forecasting, which helps you find identify the point of failure quickly and significantly reduces the compliance time, due to self- executable nature based on meeting the requirements.
7. Better risk assessments and fewer unnecessary interventions.
8. Far lower administrative expenses and elimination of costs to move physical paper across borders.

2.4.4 Examples

IBM Food Trust

The effectiveness of the IBM Food Trust solution [44] was demonstrated with a Walmart mango pilot. The results as shown in Fig. 7 clearly show how blockchain can drastically impact the supply chains of critical goods.

TradeLens

IBM and Maersk

- TradeLens is built for the industry and offers benefits to trade participants from across the supply chain ecosystem.
- Establishes an open and neutral supply chain platform.

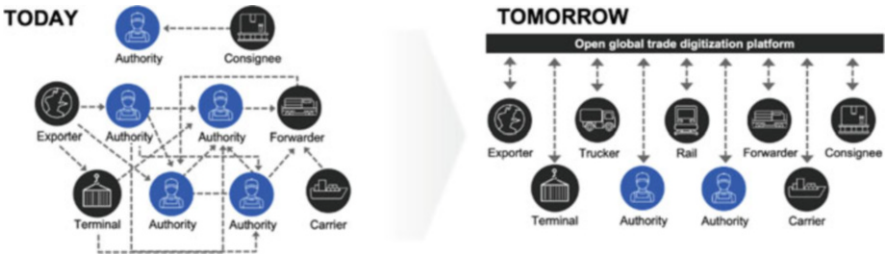


Fig. 8 Source: IBM Blockchain—Maersk

- Figure 8 shows how blockchain could simplify the relationship between various stakeholders.

3 Other Blockchain-Based E-Governance Efforts in India

The various state governments in India are piloting a blockchain-based E-Governance system to bring transparency and fairness and improve ease of doing business. Some of the initiatives are listed below.

- The government of Karnataka is piloting with the National Blockchain project and the CRUBN foundation to integrate the land registration system with the blockchain.
- Maharashtra state board of skilled development launched the world's largest blockchain implementation for education certificates [45].
- The government of Tamil Nadu launched a blockchain policy that aims at creating a set of standards and guidelines to be followed by all Tamil Nadu Government Blockchain implementations and strives to ensure enhanced inter-operability, security, privacy, and uniformity in the technology stack across government departments [46].
- The government of Rajasthan took one of India's first initiatives of implementing blockchain in the health sector. They aim to link health ID (Bhamashah) with the blockchain-based EHR (Electronic Health Records) [47].
- Uttar Pradesh has launched India's first peer-to-peer rooftop solar power trading platform. The Uttar Pradesh Power Corporation (UPPCL), Madhyanchal Vidyut Vitran Nigam Limited (MVVNL), and Uttar Pradesh New and Renewable Energy Development Agency (UPNEDA) are hosting the first-of-its-kind pilot project, involving energy generated from solar rooftop systems installed on buildings in Lucknow [48].

4 Conclusion and Future Work

Blockchain technology has evolved from Blockchain 1.0, a simple public ledger capable of recording transactions and executing some rudimentary smart contract. Various flavors of blockchain, along with the Turing complete smart contract executions, are available now. Blockchain is now used in areas that were marred with piracy. Blockchain-based non-fungible tokens (NFT) are now used to provide fair prices to the artist for their creation.

In the initial stages, blockchain struggled to find a use case in governance, but now it is readily adopted by the government to bring transparency in the system and improve ease of doing business. However, blockchain is not a panacea, and there are limits to the problems it can solve. For example, blockchain can only help the land records if the records placed on the blockchain are correct in the first instance. In a country like India, occasionally, official records may not depict the situation on the ground. In such cases, blockchain is useful only when the cleaning of the record is done first. In the case of the supply chain, if blockchain picks the data from IoT devices, the reliability of the devices themselves becomes a bottleneck. In the case of e-tender, while blockchain ensures fairness, confidentiality can be a bottleneck [49]. Sometimes the blockchain-based system uses state-of-the-art cryptography that has been theoretically validated but still has not accepted the country's standards.

References

1. M. Oede, R. J. Neuwirth, Good governance and confidentiality: a matter of the preservation of the public sphere. Corporate Governance **14**(4), 543–554 (2014). https://doi.org/10.1108/CG-08-2013-0099
2. S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf
3. G. Wood, *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. https://ethereum.github.io/yellowpaper/paper.pdf
4. L. George, J.J. Kizhakkethottam, A comparative study of zero knowledge proof and homomorphic encryption in guaranteeing data privacy in blockchain applications. Int. J. Adv. Res. **9**(Feb), 359–361(2021). ISSN 2320-5407
5. P. Jagwani, S. Kaushik, Defending location privacy using zero knowledge proof concept in location based services, in *2012 IEEE 13th International Conference on Mobile Data Management* (2012), pp. 368–371. https://doi.org/10.1109/MDM.2012.23
6. S. Terzi, K. Votis, D. Tzovaras, I. Stamelos, K. Cooper, Blockchain 3.0 Smart Contracts in E-Government 3.0 Applications (2019)
7. E. Politou, F. Casino, E. Alepis, C. Patsakis, Blockchain mutability: challenges and proposed solutions. IEEE Trans. Emerg. Top. Comput. **9**(4), 1972–1986 (2019). https://doi.org/10.1109/TETC.2019.2949510
8. https://e-estonia.com/solutions/security-and-safety/ksi-blockchain/
9. https://guardtime.com/timestamping
10. https://en.wikipedia.org/wiki/2007_cyberattacks_on_Estonia
11. *Canada Pilots Blockchain Staff Records*. https://www.globalgovernmentforum.com/canada-pilots-blockchain-staff-records/

12. Economic Development in Mexico and the Role of Blockchain. Advances in Economics and Business **7**(1), 55–64 (2019). https://doi.org/10.13189/aeb.2019.070106. https://www.hrpub.org/download/20190130/AEB6-11812580.pdf
13. *Mexico Tests Blockchain to Track Public Contract Bids*. https://www.coindesk.com/markets/2018/04/05/mexico-tests-blockchain-to-track-public-contract-bids/
14. US Air Force Gives Blockchain Firm $1.5M to Build Supply Chain Network. https://www.coindesk.com/markets/2020/06/15/us-air-force-gives-blockchain-firm-15m-to-build-supply-chain-network/
15. US Navy pilots blockchain project to track aviation parts through their life cycles. https://siliconangle.com/2018/09/21/us-navy-looks-pilot-blockchain-project-track-aviation-parts-life-cycle/
16. Blockchain Technology: Possibilities for the U.S. Postal Service. https://www.uspsoig.gov/sites/default/files/document-library-files/2016/RARC-WP-16-001.pdf
17. How the FDA is piloting blockchain for the pharmaceutical supply chain. https://www.ibm.com/blogs/blockchain/2020/05/how-the-fda-is-piloting-blockchain-for-the-pharmaceutical-supply-chain/
18. Two Blockchain-Based Digital Identity Projects Are Being Developed in Argentina. https://news.bitcoin.com/two-blockchain-based-digital-identity-projects-are-being-developed-in-argentina/
19. Digital Transformation: Blockchain and Land Titles. https://www.oecd.org/corruption/integrity-forum/academic-papers/Georg%20Eder-%20Blockchain%20-%20Ghana_verified.pdf
20. Blockchain technology: New solutions for sustainable development. https://www.giz.de/en/mediacenter/78035.html
21. Catalonia launches a project for Self-Sovereign Identity to empower citizens in the digital world. https://politiquesdigitals.gencat.cat/en/detalls/Noticia/El-Govern-presenta-el-projecte-IdentiCAT-un-nou-model-didentitat-digital-autosobirana-que-converteix-el-ciutada-en-propietari-gestor-i-custodi-exclusiu-de-la-seva-identitat-i-dades
22. This Country is Testing the Tech Behind Bitcoin for Real Estate Transactions. https://fortune.com/2016/06/16/sweden-blockchain/
23. http://ica-it.org/pdf/Blockchain_Landregistry_Report.pdf
24. UK Land Registry successfully completes blockchain trial. https://www.ledgerinsights.com/uk-land-registry-successfully-completes-blockchain-trial/
25. China's Xiong'an New Area Begins Using Digital Yuan for Salary Payments. https://www.coindesk.com/business/2021/06/14/chinas-xiongan-new-area-begins-using-digital-yuan-for-salary-payments/
26. World Economic Forum: The Known Traveller. http://www3.weforum.org/docs/WEF_The_Known_Traveller_Digital_Identity_Concept.pdf
27. Christopher Allen: The Path to Self-Sovereign Identity. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html
28. W3C: Decentralized Identifiers (DIDs) v1.0. https://www.w3.org/TR/did-core/
29. W3C: A Primer for Decentralized Identifiers. https://w3c-ccg.github.io/did-primer/
30. W3C: Verifiable Credentials Data Model 1.0. https://www.w3.org/TR/2019/REC-vc-data-model-20191119
31. Hyperledger Indy: Indy Introduction. https://indy.readthedocs.io/en/latest/#hyperledger-indy
32. Indian courts clogged with land disputes because laws keep conflicting each other. https://theprint.in/opinion/indian-courts-clogged-with-land-disputes-because-laws-keep-conflicting-each-other/254033/
33. Land Records and Titles in India. https://www.prsindia.org/policy/discussion-papers/land-records-and-titles-india
34. Karnataka Bhoomi Project. https://landrecords.karnataka.gov.in/
35. Hyperledger Fabric. https://www.hyperledger.org/use/fabric
36. The land registry in the blockchain. https://www.kairosfuture.com/publications/reports/the-land-registry-in-the-blockchain/

37. HM Land Registry to explore the benefits of blockchain. https://www.gov.uk/government/news/hm-land-registry-to-explore-the-benefits-of-blockchain
38. Blockchain Powered Land Registry in Ghana with BenBen. https://www.bigchaindb.com/usecases/government/benben/
39. Dubai Land Department (DLD) Announces the Fractional Title Deed initiative (Accessed on 13/10/2020). https://bsabh.com/fractional-title-deed/#:~:text=With%20the%20aim%20of%20attracting,a%20quarter%20of%20the%20unit
40. D. Maesa, P. Mori, Blockchain 3.0 applications survey. J. Parallel Distrib. Comput. **138**, 99–114 (2020). https://doi.org/10.1016/j.jpdc.2019.12.019
41. National Digital Health Blueprint—Government of India. https://ndhm.gov.in/home/ndhb
42. Enabling Trade Valuing Growth Opportunities. http://www3.weforum.org/docs/WEF_SCT_EnablingTrade_Report_2013.pdf
43. Supply Chains in India a reality check. https://assets.kpmg/content/dam/kpmg/in/pdf/2018/11/Indian-Supply-Chains-A-Reality-Check.pdf
44. IBM Blockchain Transparent Supply. https://www.ibm.com/in-en/blockchain/solutions/transparent-supply
45. Maharashtra State Board of Skill Development. https://msbsd.edu.in/msbsd/mr/legitdoc/
46. Connect 2020: TN Govt advocates big role for blockchain in e-governance. https://www.thehindubusinessline.com/news/national/connect-2020-tn-govt-advocates-big-role-for-blockchain-in-e-governance/article32650251.ece
47. Rajasthan leads India's first government Blockchain implementation. https://www.expresscomputer.in/news/rajasthan-leads-indias-first-government-blockchain-implementation/24028/
48. Uttar Pradesh Starts India's First Blockchain-Enabled Rooftop Solar Trading Platform. https://mercomindia.com/uttar-pradesh-blockchain-rooftop-solar-trading/
49. Blockchain technology and it's applications in e-governance services. International Journal of Recent Technology and Engineering (IJRTE) **8**(4), 5795–5802. (2019). ISSN: 2277-3878. https://www.meity.gov.in/writereaddata/files/research_paper-6.pdf

When Blockchain Meets Smart Cities: Opportunities, Security and Future Research

Roben C. Lunardi, Regio A. Michelin, Maher Alharby, Volkan Dedeoglu, Henry C. Nunes, Eduardo Arruda, Avelino F. Zorzo, and Aad van Moorsel

1 Introduction

Currently, more than 4.1 billion (≈55%) of the world's population live in urban areas, and this number is predicted to grow by a total of 2.5 billion of urban dwellers until 2050 [1]. As the number of people living in urban centres grows, problems such as traffic congestion, public health system overload, energy consumption and environmental changes will become more common. Hence, to address these growing problems, there is a need to make our cities and urban environments smarter by integrating innovative technologies based on real-time data collection and analytics

R. C. Lunardi
Federal Institute of Education, Science and Technology of Rio Grande do Sul (IFRS), Porto Alegre, Brazil
e-mail: roben.lunardi@restinga.ifrs.edu.br

R. A. Michelin (✉)
The University of New South Wales (UNSW), Sydney, NSW, Australia
e-mail: regio.michelin@unsw.edu.au

M. Alharby
Taibah University, Medina, Saudi Arabia
e-mail: mharby@taibahu.edu.sa

V. Dedeoglu
CSIRO Data61, Brisbane, QLD, Australia
e-mail: volkan.dedeoglu@data61.csiro.au

H. C. Nunes · E. Arruda · A. F. Zorzo
Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
e-mail: henry.nunes@edu.pucrs.br; eduardo.arruda@acad.pucrs.br; avelino.zorzo@pucrs.br

A. van Moorsel
School of Computer Science, University of Birmingham, Birmingham, UK
e-mail: a.vanmoorsel@bham.ac.uk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_14

together with improved connectivity and seamless interoperability into decision-making processes. Figure 1 shows some examples of areas that can benefit from this integration.

The Internet of Things (IoT) is an important concept that enables embedding microcontrollers and introducing network connectivity in many different pieces of technology, such as sensors and actuators. As these devices have become more popular and widely adopted, people are surrounded by this pervasive technology. In many cities, these devices are deployed in settings as diverse as hospitals, power grids, road (side) infrastructure, schools, shops or government buildings, where they perform many different roles and actions.

IoT technology and its wide adoption are key factors to enable the smart city concept. Currently, there exist many different specific definitions of smart city, but most rely on the emergence of IoT as a pervasive technology across city infrastructures. A smart city processes data from numerous sensors and interacts through actuators to implement smart decisions with more effective usage of resources [2]. The main goal of a smart city is to provide a better environment for the dwellers, i.e. an environment capable of adapting to observed circumstances and taking actions proactively or quickly when problems arise.

Despite the benefits that smart cities can bring to inhabitants as well as environmental resource utilisation, the amount of data produced by different sensors can be substantial and a potential technological challenge. Additionally, these sensors can be expected to rely on different protocols and patterns to communicate, while the produced data are likely not always standardised. As a consequence, there is a gap between the huge amount of data produced by heterogeneous devices and the ability to exchange and use this information. For instance, smart city administration needs to manage information gathered from sensors deployed in vehicles, buildings, infrastructure and other applications. These sensors can vary in terms of data format, deployment environment, communication protocols and application domains. For

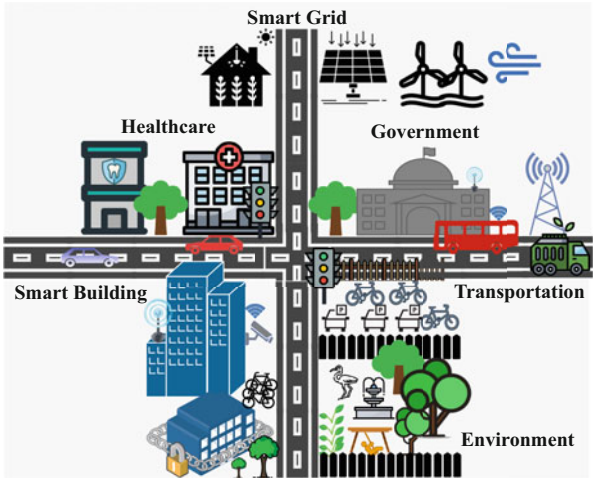


Fig. 1 Applications within smart cities enabled by blockchains

example, traffic cameras can use different protocols and data format than the sensors used by private surveillance systems.

In a smart city, effective management of large amounts of data is crucial to facilitate smart decision-making [3]. Data-driven automation of decision-making based on information received from different systems has the potential to improve the operation of smart cities. Artificial intelligence and data analytics can play a significant role as enablers of automation and smart decision-making. As these decisions can have significant impact, the information management should ensure that data can be trusted, while, more important than ever, privacy of the city inhabitants is maintained.

To tackle some of these challenges, blockchain technology is a candidate technology for smart city applications. Blockchain technology provides a transparent and trusted ledger, which maintains immutable data, integrating information from multiple systems while still providing integrity and auditing capabilities. The decentralised, peer-to-peer nature of blockchains enables aggregation of data from various resources. For example, blockchain can be used to facilitate data management for public transportation or to help in the energy trading. Figure 1 presents an overview of some smart city applications for which the blockchain technology is suitable for.

In this chapter, we discuss the role blockchain may be able to play in facilitating a variety of smart city applications. We identify existing application-specific challenges and survey existing research that aims to tackle these challenges. Figure 2 presents a taxonomy of main blockchain application domains for smart city challenges, namely, smart buildings, smart grids, Transportation, Government, Healthcare and Environment. The rest of this chapter consists of three sections. Section 2 presents how blockchains can be used in different smart city applications. Section 3 discusses cyber security requirements for smart cities that blockchains can help fulfil. Finally, Sect. 4 summarises and reflects on the chapter.

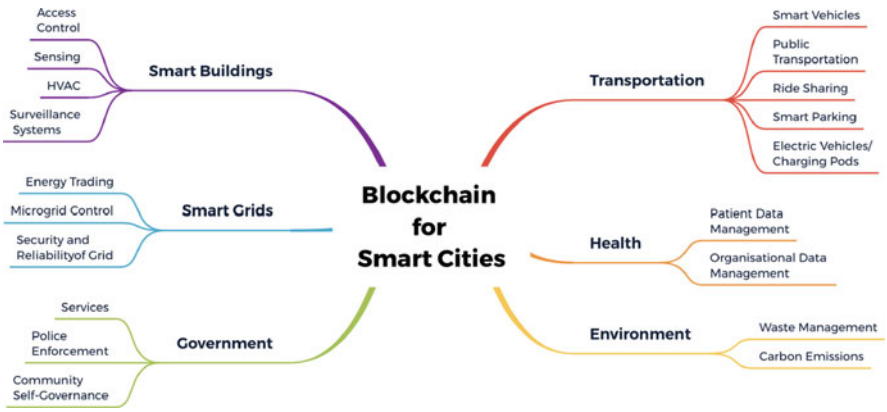


Fig. 2 Taxonomy for blockchains applied to smart cities

2 Applications

The smart city ecosystem integrates information from many heterogeneous areas as shown in Fig. 2. In this section, we highlight a number of areas of importance in smart cities and identify the challenges and the potential role of blockchain to resolve these challenges. Each subsection identifies "future directions", that is, remaining future research challenges pertaining to these areas.

The specific areas we consider within smart cities are the following:

- Smart Buildings, Sect. 2.1
- Smart Grids, Sect. 2.2
- Government, Sect. 2.3
- Smart Transportation, Sect. 2.4
- Healthcare, Sect. 2.5
- Environment, Sect. 2.6

2.1 Smart Buildings

One important and crucial aspect of a smart city is how autonomous and independent entities collaborate to provide a smart solution. Therefore, a smart city has to provide an infrastructure that enables information sharing among various entities. For example, the surveillance system of a residential building, a speed camera and the access control of a hotel can be used to collaboratively to track a suspect and, as a consequence, to solve a crime. At the same time, gas and smoke detectors of a building can help to prevent a fire from spreading by activating sprinklers or communicating with the fire department. Another example is the integration of energy storage in smart buildings, which then connect into the smart grid and electric vehicles [4]. For all these applications of smart buildings, the smart infrastructure has to connect different entities and sensors to be able to make smart decisions in the city. Smart buildings can be understood as physical buildings that integrate data from sensors in order to collaborate with the city infrastructure. Figure 3 shows an example of a smart building storey with several sensors in different rooms.

In recent years, blockchain has been proposed to handle the IoT data produced by smart buildings [5] in a secure and resilient way. Many works proposed the adoption of a hierarchical architecture for smart cities and, particularly, for smart buildings [6]. In this kind of architecture, nodes have different roles, some of them acting as overlays (also called gateways) to protect devices from external attacks, i.e. all interactions with private sensors from a private building are performed through these overlays [7].

A smart home is a particular case of a smart building. Even in this small environment, many devices, e.g. smoke sensors, surveillance cameras, etc., can produce data to collaborate with the city. As an example, sensors in pipe valves can help to detect that water has low pressure. This might indicate that water pipes

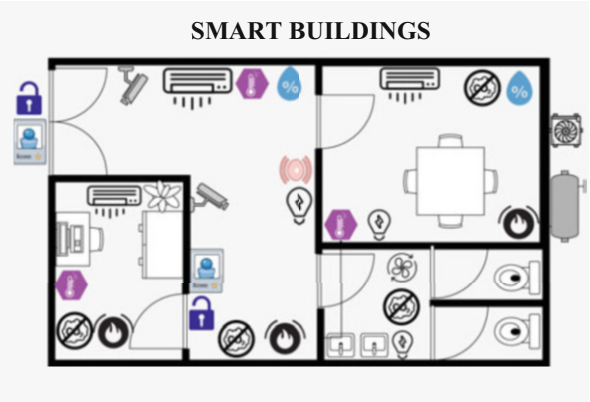


Fig. 3 Smart building storey example

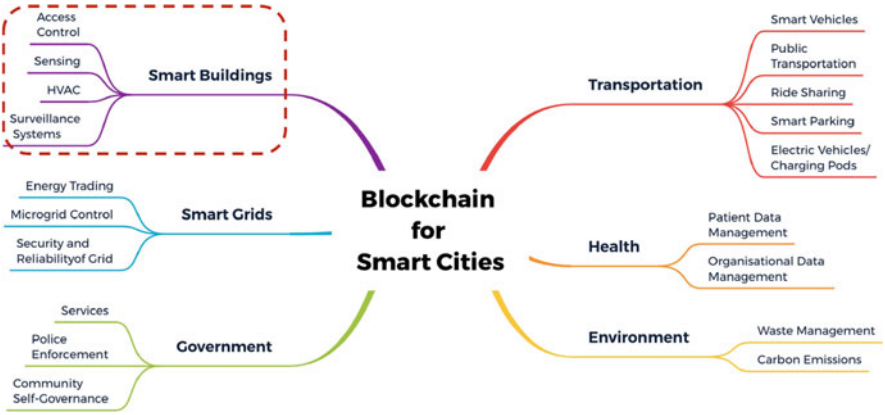


Fig. 4 Blockchain application in a smart building

in the neighbour can have a water leak problem. Consequently, integrating these data through the smart infrastructure is very important to prevent or remediate problems. Furthermore, smart home devices are vulnerable to security attacks. In order to mitigate security threats related to data sharing, a smart home node (also called as an overlay node) can manage the connection of the smart home devices and share their information with other homes or other nodes in a private blockchain [8]. This overlay node can maintain all the public access data in a blockchain shared with other homes or other buildings from a city.

There are many applications that can benefit from using blockchain to manage smart building data. Particularly, significant improvements can be achieved in the following smart building subsystems: access control, sensors, HVAC (heating, ventilation and air conditioning) and surveillance systems (Fig. 4). In the next subsections, we present some initiatives to adopt blockchain in different smart building subsystems.

2.1.1 Access Control

Access control is a recurrent topic in researches on blockchain usage in smart buildings. There are some proposals for smart homes [8], smart offices [9] and interconnecting different smart buildings [10]. The general idea is to control the access in the system, as well the level/permissions of each user and/or IoT device in the system.

In general, the majority of proposals suggest the adoption of a private (or consortium) and permissioned blockchain solution for smart buildings [11]. Consequently, each node has to authenticate in the blockchain through specific nodes (overlays, gateways or any other special node) [8]. There are different approaches that can be used to manage the permissions of each node. One solution for permission management is to use a system to define or set manually the privileges of each device (usually identified by its public key). An alternative solution is the use of smart contracts to perform the permissions management in a more systematic and transparent manner [12]. In any case, the access control is performed using the node's public key as unique identifier in the blockchain.

The two main types of access control are physical and logical, where smart building applications can benefit from a combination of both mechanisms. Logical access refers to the restriction for virtual connection to data, services and digital resources, e.g. using authorisation mechanisms to grant devices permission to access data. On the other hand, physical access control refers to the restrictions to access physical locations, such as rooms or floors in a building. Also, a building may have both logical and physical access control. For instance, a person using a smart band can unlock his office smart door-lock based on the devices' permissions. Thus, in a smart building, it is expected that physical access control is performed based on the permissions from logical access control.

Both physical and logical access controls require the correct policy management. In a blockchain, policies and its application can be performed through smart contracts. One advantage of this approach is the decentralisation of the application of the policy (i.e. execution of the smart contract). Consequently, blockchain can help to increase the resilience of the policy management used as part of the access control.

Future Directions The most important open issue in access control is the management of initial permissions for new or unknown nodes in the blockchain. An interesting approach can be the usage of reputation/trust systems to define initial permissions. Also, a further discussion about the usage of smart contracts adapted for lightweight blockchain [13], particularly in smart buildings, should be performed.

2.1.2 Sensing

Sensors and actuators have an important role in a smart building. Sensors are responsible for collecting and producing relevant data that will be used, after being

processed, to make decisions. Based on these decisions, actuators can be used to interact with the building systems and other agents. As an example of smart building, the Urban Science Building from Newcastle University (Newcastle upon Tyne/UK) has over 9,000 sensors and actuators [14] interacting with physical and virtual environments to provide smart decisions. In such buildings, it is possible to develop applications to provide smart decisions, such as using presence sensors to control lightning of rooms or using CO_2 sensors to warn occupants when CO_2 level is above a threshold. Thinking in a larger scale, a city lighting infrastructure can be used to collect environmental data (CO levels, temperature, humidity, etc.) to monitor the quality of air in different neighbourhoods [15].

Due to the sensitive information, many pieces of research proposed the adoption of blockchain to keep all data produced by these sensors and actuators safe. Blockchain can help to provide resilience and tamper-resistance to these data. However, sensors can produce a large amount of data in a very short time interval. This kind of scenario can present latency and throughput issues when using traditional blockchain solutions such as Bitcoin and Ethereum, which should be analysed carefully using model-based approaches [16].

Therefore, different blockchain solutions have been proposed [6, 17, 18] to deal with the large amount of information produced and low latency requirements. Lightweight Scalable Blockchain [19], for example, is a solution that proposes a hierarchical architecture and a lightweight consensus mechanism to deal with the throughput required by this kind of environment. Using a different approach, appendable-block blockchains [17] deal with low latency issues by allowing transaction insertion (with a high level of parallelism) in blocks that were previously inserted in the blockchain. Alternatively, blockchains can be used as a third-party service that will process IoT information on demand [20].

Future Directions There are some important open issues in the management of sensors and actuators data on blockchain. One of them is the huge amount of data that can be produced by these devices in short period of time, i.e. low-latency and high-throughput solution is required. Another problem is how the blockchain ageing affects the performance of the system, i.e. how the growing of the number of blocks/transactions impacts its performance in a long-term usage [21]. Finally, but not less important, the system has to be able to trust sensors' data, i.e. the system should be able to identify sensors/actuators' physical tampering.

2.1.3 HVAC

An important subsystem in smart buildings is the management of heating, ventilation and air conditioning (HVAC) data. Usually, HVAC is responsible for most of the energy consumption in commercial buildings [22]. A correct HVAC planning and management can result in benefits to the environment, to the city power grid and to the building users.

HVAC systems in smart buildings are composed of a large set of sensors and smart data analytics in order to provide a good relationship between adequate temperature with optimised energy consumption. Some of the sensors used in the building for monitoring the condition of a room can be also be used in HVAC systems. For example, CO_2, humidity and temperature sensors are essential to the correct HVAC operation. This kind of system can be very complex, particularly in buildings that require temperature control for different rooms/applications. Also, different applications/contexts can have different requirements on both data collection and fast response to avoid catastrophic situations [23]. For example, the temperature of a data centre room usually is lower than the temperature in a regular office room. Even usual office rooms can be dynamically configured to present higher comfort for each room. In some situations, HVAC can be a mission-critical subsystem in the building, such as to avoid thermal expansion on delicate equipment or to improve computing performance on data centres.

Additionally, the HVAC actuators can be managed and controlled based on blockchain data. For example, HVAC systems are composed of air handling units (AHUs) and variable air volume (VAV) systems. AHUs can supply cold or hot air through the ducts that will be handled by VAVs to control the temperature for each thermal zone. All this information should be maintained in a tamper-resistant way [24]. As presented previously, many researchers proposed different solutions to adopt blockchain to manage sensors. Due to HVAC characteristics and the dynamism in its operation, blockchain can help to protect against malicious attacks that try to tamper with these data. Additionally, blockchain can be used to help in a distributed processing of the data generated by temperature, presence and any other sensors used in the HVAC systems [24].

Future Directions The adoption of blockchain for HVAC systems still has some open issues to integrate different subsystems, for example, how smart contracts can be used to integrate data from solar panels, water cooling/heating, air conditioning, venting systems, etc. There are few solutions proposed in the integration of multiple subsystems.

2.1.4 Surveillance Systems

Nowadays, surveillance systems are being broadly adopted in big cities. These systems can be used both in private and public environments [25]. However, surveillance systems can present some challenges, such as data should be available; data should not be tampered with; a large amount of data should be stored and backed up; images should be processed in a very short amount of time. Recently, some works proposed the adoption of blockchain, in special to ensure resilience, tampering-proof, distributed verification of stored data and distributed processing of data.

In general, blockchains can help with the resilience and tamper-resistance of data produced. However, traditional blockchains present issues to store large amounts of

data. To tackle these issues, there are different proposals. One of them is using the blockchain to safely store the hash of images and video streams and storing them into an off-chain solution (e.g. using a cloud storage system) [26]. An interesting alternative for off-chain storage is the usage of the InterPlanetary File System (IPFS) [27]. IPFS is a peer-to-peer file system that uses structure based on hash keys [28]. Another solution to deal with a large amount of data is the usage of a more flexible and lightweight blockchain, such as appendable-block blockchains [29].

One important aspect of surveillance systems is the processing required to stream and store the generated images and videos. Blockchain can be used to control the access to the images and backup it in a distributed way through the usage of smart contracts [26]. In a simplified definition, smart contracts can be understood as pieces of code that are executed by the nodes in a blockchain. This distributed processing can help to reduce the bottleneck of centralised systems. Also, the use of smart contracts can help to define access policies to data, helping in private issues that can occur with unauthorised access to the surveillance systems [30].

Future Directions Surveillance systems have to deal with a huge amount of images and videos. This kind of data demands a large amount of storage, processing and network bandwidth. All these data, even using off-chain solutions, can be hard to be processed and transferred by smart city infrastructure. This remains a challenge for the application of blockchains in surveillance systems.

2.2 Smart Grids

Sustainability, security and reliability are important aspects of the development of the energy infrastructure for smart cities. Several conventional energy grids rely on fossil fuels for energy generation, which accounts for higher carbon emission and pollution. Furthermore, several conventional grids, not only fossil fuel, suffer from inefficiencies due to long-distance transmission and distribution losses and lack of information visibility to energy producers and consumers. The centralised energy generation and control of the conventional grid are susceptible to attacks and malfunctioning, which has caused wide-area blackouts and system failures before.

Recently, there has been a wider adoption of sustainable and environmental-friendly distributed energy generation from renewable energy sources, for example, solar and wind [31]. With the advances in distributed energy generation and energy storage technologies, energy prosumers have appeared, which are agents that can produce energy for their own consumption and trade their surplus energy with other consumers locally. However, integration of the distributed energy sources to the conventional energy grid has introduced new challenges. The intermittent nature of the renewable energy sources causes fluctuations in the amount of energy produced and transmitted affecting the reliability of the grid [32]. The integration of centralised and distributed energy sources requires the grid to facilitate bi-directional energy and information flow. Smart grid concept has been introduced

to improve the efficiency of electrical grid and enable utilisation of distributed energy sources using smart sensing, communication and control technologies [33]. However, centralised control of smart grids is challenging due to scalability, security and privacy issues. There is a need for a distributed system for managing the energy supply and demand enabling trade between agents locally without requiring intermediaries. Recently, blockchain-based decentralised management solutions have been proposed by researchers to address scalability, security and privacy challenges for smart grids [34–41]. Figure 5 shows an example of a distributed energy source integrated into the main energy grid using blockchain technology. This section presents the application of blockchain technology in smart grids (Fig. 6).

2.2.1 Energy Trading

One of the main application domains for blockchain technology in smart grids is energy trading. In a smart city, the smart grid enables agents to trade their surplus energy with other agents locally by facilitating bi-directional energy and information flow. Centralised trading platforms rely on trusted intermediaries and are susceptible to cyber attacks, data manipulation and vulnerabilities due to single point of failure. Furthermore, storing large amounts of sensitive data on a centralised platform raises privacy concerns. Historical data regarding the energy generation patterns can be used to predict user activities and behaviour. Centralised trading platforms incur high operational costs due to the costs associated with the centralised management of a large number of distributed prosumers, consumers and producers and the involved intermediaries.

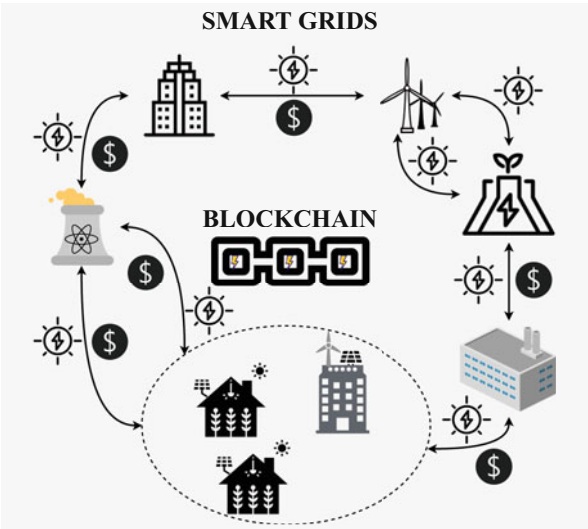


Fig. 5 Energy grid integrated using blockchain

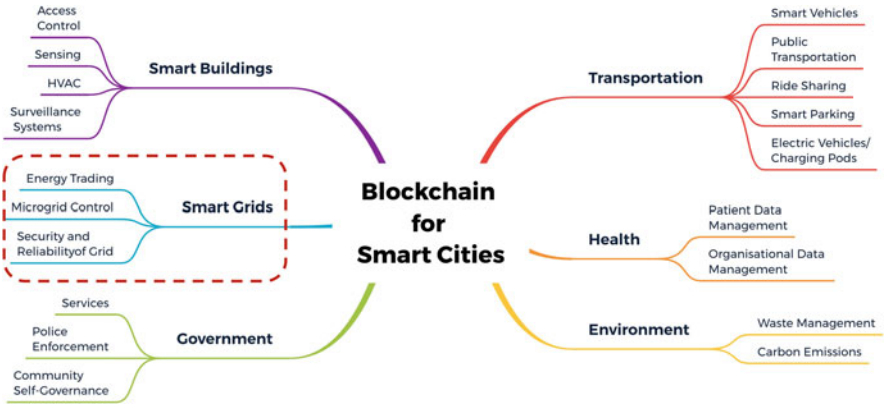


Fig. 6 Blockchain application in a smart grid

Recently, blockchain-based platforms have been proposed to enable energy trading in a decentralised, transparent and traceable manner without relying on trusted intermediaries. Blockchain-based energy trading platforms are more resilient against system failures and attacks due to the decentralization, inherent redundancy and the consensus mechanism of the blockchain network. The privacy of the users can be preserved using multiple pseudonym addresses, ring signatures, zero-knowledge proofs and data and route hiding techniques [34]. Furthermore, functionalities and proactive features such as auctions, pricing, payments, incentives, demand-response and alerts can be automated using smart contracts on the underlying blockchain-based trading platform.

In [35], Li et al. propose a secure peer-to-peer energy trading framework for microgrids, energy harvesting networks and vehicle-to-grids based on a consortium blockchain. The proposed trading framework addresses the transaction confirmation latency challenge by a credit-based payment system and maximises the financial benefits for credit banks by using an optimal loan pricing strategy.

Preserving the privacy of energy trading users is challenging when unencrypted user data is recorded on blockchain, which is accessible by the network users. Furthermore, using other datasets and data mining algorithms, it is possible to launch linking attacks. In [36], the authors propose a noise-based privacy-preserving mechanism for anonymizing the trading data recorded on a consortium blockchain while still guaranteeing traceable and accurate energy trading with low latency.

In [37], Dorri et al. propose a Secure Private Blockchain (SPB) framework for distributed energy trading between smart grid participants. The proposed framework utilises atomic transactions to eliminate the reliance on a trusted third party. To reduce the communication and computation overheads, price negotiations between participants are conducted off-chain, and the negotiation transactions are routed by backbone network nodes with sufficient resources. Once the trading participants agree on the price, the agreed price is recorded in the trade transaction. Furthermore,

SPB anonymously authenticates smart meters without revealing the true identities of the devices.

Future Directions Although blockchain is a promising technology for enabling distributed energy trading in smart grids, there are challenges that need to be addressed for its adoption, such as high transaction latency and transaction costs, low throughput, data privacy and security concerns, lack of regulatory frameworks, interoperability and scalability.

2.2.2 Microgrid Control

A microgrid is a local cluster of loads, distributed energy generation and energy storage units that can operate stand-alone to supply electricity and can be connected to the main energy grid at a point of common coupling to facilitate a smart grid [42]. Bi-directional energy flow, the output voltage and frequency deviations of distributed energy generation units, stability issues related to the seamless transition between stand-alone and grid-connected modes of operation, the load uncertainties and the coordination of distributed energy generation units make the control of microgrids a challenging task. Centralised control of microgrids is infeasible due to the extensive communication and computation required [42].

In [38], Pop et al. propose a blockchain-based demand response management solution and implemented their solution on an Ethereum platform. The data from distributed energy prosumers is monitored and stored on a blockchain in a secure and tamper-proof manner. The smart contracts automate the validation of demand response agreements and the expected energy flexibility levels, rewards and penalties mechanism for compliance of prosumers to desired demand energy profiles and the rules for balancing the energy supply-demand at the grid level.

In a microgrid, distributed energy producers may try to inject all of their available power into the grid to maximise their revenues. However, when the consumption is low, the surplus power may cause dramatic voltage increases and grid instability. To address this problem, Danzi et al. [39] propose a distributed proportional-fairness control strategy that dynamically chooses a subset of distributed energy producers to act as voltage regulators and reduce their power outputs based on the record of their control history. The proposed proportional-fairness control strategy is implemented by a smart contract on a blockchain platform to avoid a central authority. For fairness, the distributed energy producers are incentivised to participate in the voltage regulation using a credit exchange mechanism, where credit statuses of all distributed energy producers are tracked on the blockchain platform.

Grid power outages introduce supply-constrained environments and supply-demand gaps. In [43], supply-constrained environments are modelled using game theory by taking into account the utility of consumers and the performance of the system through demand side management strategies. Demand side management reduces the stress on the grid when the supply is constrained due to power outages. Incorporating energy storage units together with distributed energy generation

can further help to balance the supply and demand. Demand side management enables consumers to schedule their loads and reduce the utility bills. The proposed blockchain-based energy trading platform enables users to apply the demand side management strategies in an automated manner using smart contracts.

Future Directions The control of microgrids requires low latency and large amounts of data flow. Thus, future research should focus on reducing the latency of blockchain transactions, efficient data storage, reducing communications and computation costs.

2.2.3 Security and Reliability of Grids

Ensuring the security and reliability of smart grids is challenging due to the requirement of bi-directional energy and information flow, the large scale of the grid, heterogeneous and failure-prone grid components with constrained capabilities, complexity of component interactions and algorithms and the physical exposure of the grid components. In [44], the most important security properties of smart grids have been identified as (1) confidentiality of meter data; (2) integrity of data, commands and software; and (3) availability of information and power.

Recently, blockchain-based approaches have been proposed to improve the security and reliability of cyber-physical systems. As described in [40], to improve the security and reliability of cyber-physical applications, blockchain-based mechanisms can be applied at various layers of the application architecture, such as at the device level, data acquisition level, gateway level, analytics and storage level or end-to-end.

In [41], Liang et al. propose a blockchain-based data protection framework against data manipulation by cyber attacks for improving the security and robustness of smart grids. The proposed framework provides a distributed information gathering and storage mechanism on a private blockchain network, where only authorised smart meters can become network nodes that are allowed to transfer and store grid data based on a majority rule.

Smart grids include devices such as smart meters, power protection devices or other electrical terminals, which are prone to failures and malfunctioning or may require version updates. In [45], a secure equipment diagnosis mechanism has been proposed for smart grid devices based on an Ethereum consortium blockchain for efficient and convenient device maintenance. In case of a device failure, malfunctioning or version update, the device sends a diagnosis request to the blockchain. Using a double auction mechanism, the proposed diagnosis smart contract chooses the vendor for diagnosis service based on the auction bids and the vendor credit.

Future Directions Using pseudonyms on blockchain may not guarantee a complete anonymity of identities, raising confidentiality concerns. Another important aspect is data trust. Although blockchain guarantees immutability of device data once recorded on the ledger, it does not guarantee that the device data is accurate.

Malicious devices may manipulate data before submitting it to the blockchain. Further research may focus on mechanism design to prevent malicious activities from improving the accuracy of data.

2.3 Government

In a smart city, the government is one of the main consumers of the data captured by the networked IoT devices to improve the lives of residents and the management of services and resources. Blockchain technology can enable governments of smart cities to improve the quality and variety of services offered to residents by enhancing transparency, security [46], automation, data-driven decision-making and democratisation of governance by empowering the residents to participate in decision-making processes. Figure 7 shows an example of blockchain technology being used to integrate different government services, such as real estate, the justice system, voting and election, healthcare, etc.

In fact, even before the emergence of the concepts of smart city and blockchain, some preceding ideas already existed. For example, the idea proposed by Nick Szabo [47] involved a notarial service that would operate with self-executing contracts and automatic locks and would fit in the domains of government, blockchain (smart contract) and smart city. The proposed service would work such that when a person buys a house, the legal proceedings would execute automatically, and the house lockers would automatically grant access to the new owner.

In this chapter, some sections have some relation to government, i.e. Sect. 2.5 on healthcare, Sect. 2.6 on environment and Sect. 2.4 on transportation. Those sections have a considerable number of research outputs dedicated to them. Because of that, we decided to have a separate section for each of them instead of including

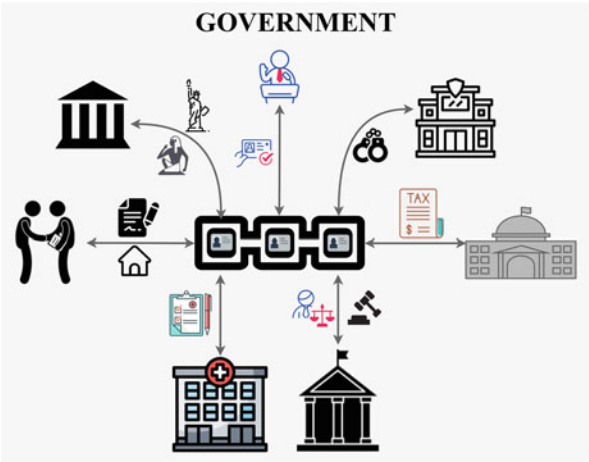


Fig. 7 Blockchain applied to government services

them in this section. Also, in many of the works and discussions in those sections, the government domain is implicit, and these are areas related to the government services, but this relation is not discussed.

In this section we explore three areas: services, policy enforcement and community self-governance (Fig. 8). Services are about improvements in traditional services provided by the government. Policy enforcement regards on how the use of blockchain in smart cities can help government to enforce public laws and regulations. Lastly, community self-governance discusses decentralisation of government functions to give the citizens more decision power and how the concept of smart city works in this context (see the Venn diagram in Fig. 9).

It is important to note that in many works the concept of smart city is extended beyond the use of IoT devices to generate data about a city infrastructure to facilitate their city dwellers' everyday lives. Some works about smart cities [48–50] do not even consider IoT devices in their definition of smart city, using similar definitions presented in the work of Yin et al. [51] for the domain standpoint. Yin's research presents a technical infrastructure standpoint focusing on the Information and Communication Technology (ICT) and city infrastructure.

2.3.1 Services

Successful e-government initiatives often relate, as one of their first steps, to the definition of a secure mechanism for providing reliable government services. The case of Estonia [52] is one of the most illustrative. The country began in 1990, with the creation of a Personal Identification Code (PIC), a unique number associated with each citizen residing in the country. In 2002, the solution evolved into a fully functional population registration system, including the issuance of digital identities and support for digital signature of documents, and came into operation [53].

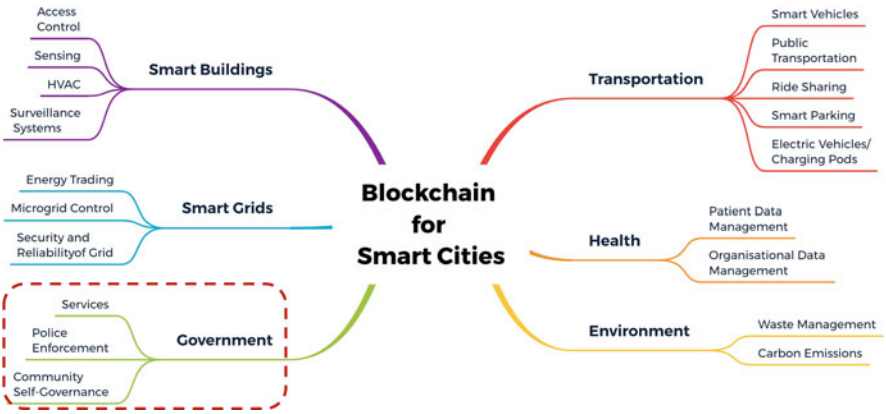


Fig. 8 Blockchain application in the government

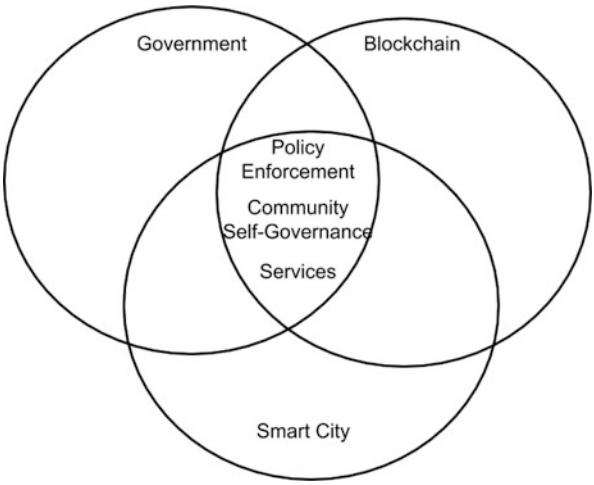


Fig. 9 Government Venn diagram

Currently, the Population Register operates as a central database where the main personal data of citizens are kept: name, sex, date and place of birth, nationality, residence permit, place of residence and marital status, all associated with the PIC. Estonia currently offers all its citizens a digital identity, in the form of an ID-Card, a Mobile-ID or a Smart-ID [54], already issued to approximately 98% of Estonians, with 67% using it regularly. Digital signatures are legally equivalent to a handwritten signature and face-to-face identification in Estonia. More than 900 million digital signatures have already been made in the country, which has 1.329 million inhabitants.

Estonia is an important use case of a government providing numerous services to society, ranging from healthcare to notarial services, and for different entities like people, businesses or other government agencies. It is a challenge on its own to coordinate and track this variety of services, sometimes needing to also coordinate with third parties and solve very specific domain problems to the area of a specific service. Because of that solutions that can help to organise these multiple parts that compose a government service are needed.

On this topic, there are a limited number of papers that treat explicitly about improving government services in the domain of smart city, government and blockchain. The existing works [55, 56] are closely related, but not limited, to the use of smart buildings in public spaces to help in the management of its operations.

In [55], Siounetry et al. propose the use of Building Information Modeling (BIM), Information and Communications Technology (ICT) and IoT to help in museum operations. Museum operations are particularly challenging because multiple actors present different access rights (visitor and personal), and there are security concerns with the artefacts stored (especially important to avoid situations like the painting robbery in the National Museum in Stockholm [57] and the fire in the National Museum in Brazil [58]). The use of IoT sensors can help in daily activities,

including capturing visitors' activities and even emotions. This information can be used to help in the museum curation. Blockchain in these scenarios can help in securing the IoT devices and controlling who has access to their information.

It is important to highlight that in a smart city environment, some systems and solutions do not use devices and still play an important role in providing information to the city governance. Some examples are public registries, tax management and digital identities. These are essential areas in a city but, however, are not directly related to device utilisation and still can have benefited by using blockchain technology.

Digital Identity An example of such a system is the Bitnation project [59] that aims to implement a Decentralised Borderless Voluntary Nation, which "citizens" can voluntarily join. The project is based on a software platform called Pangea, currently built on the Ethereum platform, although it positions itself as an agnostic and promises future integration with other blockchains, such as Bitcoin, EOS and Tezos. Through Pangea, it is possible to define a context of decentralised jurisdiction in which citizens who voluntarily adhere to it are subject to peer-to-peer arbitration rules guaranteed by smart contracts. Each individual citizen of a Bitnation nation is associated with a unique and searchable identification in the form of a pair of cryptographic keys. Every action taken by a citizen on Pangea, such as sending public or private messages or creating a smart contract, is digitally signed with its unique and searchable identification.

Notary and Public Registries The blockchain can also support the notary and public registries. Spielmanin [60] explores public registration of properties using blockchain. Spielmann's research uses blockchain to guarantee time stamping on transactions (virtual notary), resilience and immutable registry. As an example, this application commences when a person wishes to register a property. Then, the person needs to present the documentation related to the property to the registry authority. The authority verifies the documentation and generates a digital transaction. The transaction is signed with the authority keys, transferring the property to the new owner address in the blockchain. Although the technology enables such a solution, some challenges remain, such as governance issues and legacy physical registry management.

Taxes Tax management is an important activity in any city around the world. Traditional tax management involves many processes, some of them manually handled. Technology has been applied in the tax area to improve and automate some of these processes. However, in a smart city, these processes should be automatised and "smart". Chen et al. [61] proposed a system called Blockchain-based Payment Collection Supervision System (BPCSS). The BPCSS system is composed of three main actors: customer (spending digital currency), merchant (earning digital currency) and the government (auditing digital currency transactions and collecting taxes). Such a system makes all the transactions transparent, and computing taxes become clear. This leads to more credibility in the transactions and could reduce the number of disputes in tax collection.

Future Directions The application discussed in this section can be expanded similarly to a myriad of other government-administered domains, such as public agencies, libraries, parks, streets and other public spaces. Also, it can do more than helping in the management and orchestration of different actors that interact with a building. It can act proactively through the use of IoT sensors, for example, detecting a defect in the road infrastructure and orchestrating a repair with a contractor.

2.3.2 Policy Enforcement

Policy enforcement is the means by which a government ensures compliance with its laws and regulations. However, as observed in the work of Maria-Lluisa et al. [50], despite the existence of multiple environmental regulations, some cities are recording the highest pollution levels of their history. Another example is the record forest fires in the Amazon forest and Pantanal in Brazil, generally intentional fires to clear land for agriculture use. Although this kind of practice is illegal and there is legislation in place to combat it, the enforcement of such legislation is a challenging task. Generally, it depends on government officials to verify if legislation is obliged, some degree of trust from other parties that they will actually obey the legislation and that data presented by third parties are honest and correct. To address these challenges, it is already a common practice to use smart devices.For example, almost all cars have Onboard Diagnostics II (OBD2) interfaces that help to keep track of cars' emissions of pollutants; some cars even adapt their configuration during use to reduce the pollutants emission. Another example is the traffic lights that automatically detect red light violations.

Combined with IoT technology, blockchain can help solve the need for trust in regulatory compliance. In [62], the authors present a theoretical model for using blockchain and IoT to back regulatory compliance in sanitary regulation for restaurants. Blockchain would give transparency and integrity to data collected, while still maintaining data privacy, and IoT sensors would help to automate the capture of data. In a similar fashion, Tian [63] proposes a system to help with traceability and regulatory compliance in China agri-food market, in which blockchain would also be used to provide data transparency and integrity, while IoT devices would be used in the supply chain network to keep track of the state of the food products. This information then can be used by consumers and regulators to verify food quality. Other cases include control of illegal products, for example, tracking illegal tuna fishing [64]. In Sect. 2.6 other examples of carbon footprint and waste management will be presented.

Future Directions The existing research focuses on policy enforcement, but there are other benefits that can be explored. One that can be discussed is the reduction of bureaucracy, as the near real-time data generation and ease of data sharing can improve the interaction between public agencies and other stakeholders. This reduction in the bureaucracy can benefit both the government and the businesses by reducing the processing time and the associated costs.

2.3.3 Community Self-Governance

In a democratic society, the decision power is focused on its citizens, where each citizen can propose, vote and debate the directions which its society will take. Community self-governance takes this principle at a local level, where decisions about a community can be openly debated and decided by its members. There are great challenges to implementing such concept; the most notable is the difficulty to create a transparent, secure and efficient voting mechanism. Also, it is hard to organise proposals and implement them once approved. However, blockchain in smart city concept can help tackle these challenges.

Blockchain in many cases presents itself as a disruptive technology. One area that it can potentially change an established paradigm is in the interaction between government and citizens, changing the way decisions are made. First, it enables a more open society [49, 50], which means wider dissemination of information between government and citizens. Second, it can facilitate a voting process while also establishing trust in the process. This can empower a city population allowing a more direct democracy where decisions can be voted by individuals instead of representatives. Finally, smart contracts can automate the execution of decisions.

The smart city empowers this potential, by allowing fast access to information in almost real time and the automatic operation of changes decided by the citizens. One work that gives emphasis to this kind of situation is the work of [56]. The work proposes a scheme to allow different actors that interact with a park to co-build and co-operate the park. At the same time, smart contracts support operations and decision-making. IoT devices support operations too and collect data about the park operation.

The further of this kind of application depends in great part on the acceptance of this paradigm change, where governments decentralise their decision power in favour of other parties. Also, the use of blockchain as a voting mechanism is still disputed, which is essential for this kind of solution.

Future Directions More use cases are necessary in this area. Also, the development of a methodology which can help the transition of a centralised government operation to a decentralised model would also be an important step to solidify this kind of application.

2.4 Smart Transportation

Intelligent or smart transportation is a fundamental area in the smart city ecosystem. The smart transportation domain is composed of different transportation solutions and multiple entities working together. Examples of entities deployed in the smart transportation vertical are vehicles, bicycles, ride-sharing, public transportation (cabs, buses, trains, etc.), charging pods for electric vehicles, smart parking systems, roadside units and incident management (linked to police, firefighter, ambulances,

etc.). Figure 10 shows an example of blockchain technology being used to integrate different transportation solutions. On the one hand, these entities play a key role by producing information; on the other hand, the entities consume information produced by their peers.

In this section, we highlight the most common smart transportation applications (see Fig. 11) present in a smart city and identify how blockchain technology can be applied to address existing problems in each part of this area.

2.4.1 Smart Vehicles

Nowadays, vehicles are following the technology advance and are equipped with many electronic components, sensors, cameras, engine control unit (ECU), light detection and ranging (LIDAR), etc. Combining all these sensors grants the vehicles the capability of assisting the driver or even follow a route without requiring assistance from the driver. Based on these capabilities, we can classify the vehicles as traditional vehicles (with limited driver assistance), semi-autonomous (can accelerate, brake, maintain safe distance) and autonomous (can drive from point A to B without driver assistance). All these vehicles are present simultaneously in the smart city, producing and consuming information. Therefore, information

Fig. 10 Different transportation solutions integrated using blockchain

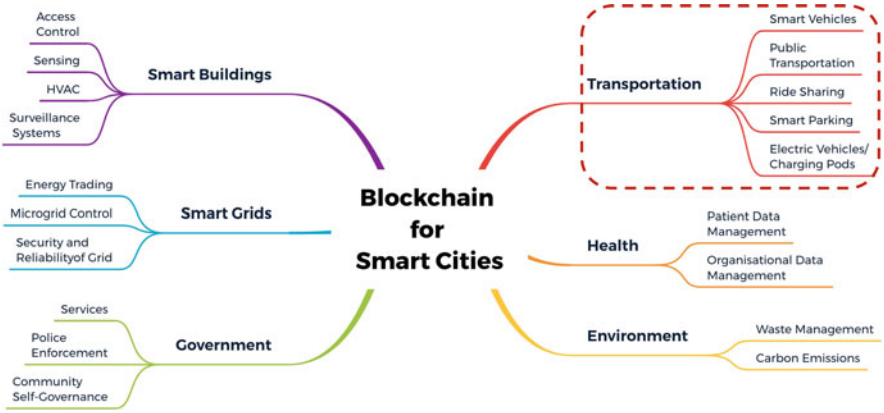


Fig. 11 Blockchain application in the smart transportation

sharing is a key factor in the whole smart transportation solution. Blockchain can be used in different ways to support this scenario: (i) enable the data sharing among vehicles; (ii) introduce an incentive mechanism; and (iii) identify possible liability in accidents.

Many researchers are identifying the blockchain application to manage the data produced by the smart vehicles [65–68]. For example, in [65], Michelin et al. introduce a blockchain-based solution, in which the smart vehicles are responsible for producing information by reading the sensors and the data is sent to the roadside units (RSU). The data is stored as transactions in the blockchain. The RSUs are responsible for storing and managing the blockchain and additionally sharing the data among the peers. The solution introduces a new data model for blockchains, in which each vehicle is represented by one active block at a time, and the data produced by each vehicle is appended into the corresponding block. This data model enables the blockchain to decouple the transactions from the block headers and still introduces the capability of appending multiple transactions in different blocks, thus reducing the time to append new transactions into the blockchain.

One issue identified by Li et al. in [69] is that some users are not keen to share the information produced by their vehicles. Tackling this problem, CreditCoin was proposed as an incentive mechanism provided to the vehicles to share the produced data. In the proposed solution, the blockchain is responsible for controlling the incentives according to the amount and quality of information shared. The architecture grants incentives for vehicles that produce information and for those which are validating the information produced by peers.

In the context of transportation, accidents are a constant variable that should be monitored by the services (police, ambulance, firefighters), and actions should be taken quickly to avoid further problems in the city. The actions are not limited to the accident event but also extend to identify the liability with the parties post-accident [70]. In this context, Oham et al. proposed B-FERL [71], a blockchain-based framework to monitor the liability in case of accidents. The RSUs are responsible for maintaining the blockchain and regularly send challenges to the connected vehicles; the response is proof that the internal vehicle status (ECUs) was not compromised. The data is stored in the blockchain and can be used to determine vehicle credibility when requesting the data.

Future Directions There are many different blockchain applications in the domain of smart vehicles, for example, blockchain technology can be used for data sharing, accident liability, etc. Besides these applications, data privacy and data trust aspects are key research points to promote the wide adoption of this technology.

2.4.2 Public Transportation

Public transportation is one of the most popular and sustainable transportation in urban centres. City administration incentives dwellers to use public transportation over private vehicles to reduce pollution and traffic jams. Blockchain technology

can be applied to improve mobility by creating incentives to promote people to use such sustainable transportation (buses, trains and trams).

Nguyen et al. in [72] proposed a blockchain-based Mobility-as-a-service (MaaS). The goal of the blockchain is to provide a common layer to manage the payment by connecting passengers and transportation providers directly. Using smart contracts, the user performs a request to the blockchain when commencing the journey. When finishing the journey, the system computes the total travelled and bills the user according to the journey. In this architecture, the transportation providers act as blockchain miners, validating transactions and creating the block, while the passengers are responsible for creating transactions requesting services. The public ledger enables the solution to present trust and transparency in the information managed.

Future Directions The research in this area is not limited to the use of blockchain as a payment mechanism, but it can also be further explored for some additional collaborative applications such as identifying public transportation occupancy and routes delay/detours.

2.4.3 Ride Sharing

Rider sharing has become popular with UberPool, and it changed the way how people commute in big cities. This service is essentially provided by some companies (such as Uber) to connect drivers and riders. This service can have several benefits such as reducing traffic congestion, reducing pollution, costs, etc. However, the current model presented relies on a central trusted party that connects these parties. The traditional model of ride sharing, with a single company managing the data from all drivers and passengers, due to the traditional centralised architecture, is susceptible to common problems such as data leakage [73] and Distributed Denial of Service(DDoS) [74] among others. Blockchain technology appears as a possible solution to address these issues.

Baza et al. [75] proposed B-Ride, a decentralised ride sharing service based on a public blockchain. In this research, the blockchain works as a layer to connect the rider with the drivers. The solution starts with the rider requesting a ride by performing an initial deposit in a time-locked smart contract. Thus the driver that chooses to pick the rider also performs a deposit in the same smart contract. The driver will receive the payment once the arrival is proven at the pick-up location and completing the journey. The total fare is computed based on the elapsed distance from the pick-up and the drop-off. In addition to that, the blockchain is also responsible for maintaining the driver's reputation, and the score is also computed upon travel completion. The authors demonstrated the framework by implementing the proposed solution using Ethereum public test network blockchain.

BlockV [76] is a blockchain-based solution presented by Pat et al. This research focused on managing the fairness of the rides for ride sharing solutions. The blockchain is responsible for maintaining the public records regarding the travel

creation, completion, dissatisfaction or even abortion of the travels. The solution was evaluated in a private instance of the Ethereum network.

Such ride sharing solution is not limited to vehicles but can also be extended to other sharing transport like bicycles. As people are getting aware of environmental problems, especially related to pollution, many people are adopting alternative transportation. One sustainable transportation that is increasing in cities is bicycle utilisation. Aiming to provide some incentives to those that are using bicycles as transportation, Jaffe et al. [77] introduced a blockchain-based solution to grant incentives to people that opt for bicycles. The proposed solution uses deployed smart contracts in the Ethereum blockchain to validate the information and compute the incentives to the users. Bicycles are required to be equipped with sensors that run an Ethereum light client and share the information collected from GPS.

Future Directions The ride share main application is focused on controlling the fare payment and reputation management either for the drivers and riders. These applications can also raise user awareness to reduce the environmental impact by increasing sharing. However, future research is required to apply blockchain technology to enforce the rider and drivers' privacy, as some sensitive data can be inferred based on their routes.

2.4.4 Smart Parking

The task to find parking spaces in the central business districts of major urban centres is a cumbersome task. It can take the driver to spend a long time before finding a parking space. This causes some side effects as traffic congestion, time wasted and pollution. Some companies proposed proprietary solutions [78–80] to address the parking problem; however, this introduces a different problem, related to that each parking building/public parking adopt a different solution, and the driver is again overwhelmed by managing to find a parking space.

Amiri et al. [81] proposed to use a consortium blockchain to tackle the problem where multiple parking lot owners can cooperate and provide space availability. The parking lot owners are responsible for maintaining the nodes running the blockchain. These nodes are responsible for processing the transactions and reach a consensus before persisting it. The system defines five distinct phases, initialisation, submitting parking offers, parking offer retrieval, parking reservation and parking and payment. The proposed solution also aims to ensure the driver's privacy by using private information retrieval (PIR) techniques when the driver is fetching parking information from the blockchain.

The blockchain can also become a connection layer between the driver and the parking lot owners. Ferreira et al. [82] used the blockchain as a collaborative platform where drivers can provide information regarding parking spaces and thus used gamification techniques to increases users' participation.

Future Directions The current research was focused on identifying possible parking spots, either collaboratively or by parking owners. Additionally, the blockchain

can also provide middleware for data integration in the sensors deployed in a parking building, providing accurate and near real-time utilisation information.

2.4.5 Electric Vehicles Charging Pods

The number of electric vehicles (EV) is increasing in most cities around the world. Some governments are encouraging people to adopt these vehicles as it collaborates to reduce gas emissions when compared with traditional fossil fuel engines. Instead of traditional fuel, the EV requires to connect to a power grid to recharge the batteries, and this process can take from minutes to hours. The energy trading process is similar to the discussion about smart grid (see Sect. 2.2) and also related to smart parking (see Sect. 2.4.4). In this context, blockchain has been applied as a middleware to mediate the energy trading between the car and the payment system.

Su et al. [83] proposed a blockchain-based solution that uses a permissioned energy blockchain to secure charging services for EVs with the execution of smart contracts. The EV selection follows a reputation-based DBFT consensus algorithm to reduce the processing cost. Through the proposed framework, the driver can choose the energy type (traditional, clean, mixed). In this blockchain, preselected EVs are responsible for auditing and sharing transactions without the trusted intermediary, thus identifying possible suspicious energy trading transactions.

Aiming to manage the power fluctuation levels in the grid network, Liu et al. [84] proposed a blockchain-based solution (AdBEV). The proposed solution takes into account the EV battery capacity, charging rates and EV user behaviour, and based on this data, the system proposes an optimised schedule for charge and discharge. This proposal uses the Iceberg algorithm, which correlates the energy market with the financial sector, to present the best schedule for the user. The proposed solution was evaluated using Ethereum blockchain and smart contracts.

Another blockchain application for EVs is to manage the billing system between charging pod [85] and EV. Blockchain can also act as a mediator to negotiate and select the most convenient charging pod [86].

Future Directions Blockchain technology enables the distributed solution to extend the energy market for vehicles in different areas, such as enabling the user to choose the energy source, auditing existing transactions and receiving incentives for that. Another research topic is to use the blockchain for interoperability among multiple different energy markets.

2.5 Healthcare

Healthcare is a mission-critical application in smart cities. Particularly in pandemic situations (e.g. COVID-19), smart applications can help cities and governments to provide better healthcare services. For example, many contact tracing apps

were recently used to help to mitigate the propagation of COVID-19 [87]. In a catastrophic situation (e.g. hurricanes or other natural disasters), the aggregation of data from different hospitals can be used to provide an optimised help to dwellers. Data about the bed occupancy, drug inventory, human resources and other mission-critical information can be shared between hospitals, resulting in an optimised allocation and usage of the (usually) scarce resources. Also, sharing data about transferred patients (e.g. MRI, blood tests, etc.) can help to save lives. Figure 12 shows an example of blockchain technology being used to integrate different healthcare entities and solutions.

We can categorise healthcare in two different views (see Fig. 13): patient data management (treatments, diagnosis, exams, etc.) and organisational data management (law, regulations, drugs and inventory control, human resource, equipment, etc.). Smart healthcare should integrate both views. Blockchain can help in this integration, ensuring an updated and not tampered view of the whole system. Both patient data management and organisational data management have received a lot of attention from Academia recently. In the following subsections, we discuss some important advances in both patient data management and organisational data management.

2.5.1 Patient Data Management

In the case of patient data management, blockchain was proposed as a solution to solve many issues. For example, Electronic Health Records (EHR)—data about patient diseases, lab tests, exams, diagnosis and treatments—can be persisted in a blockchain to guarantee the availability of the patient data [88].

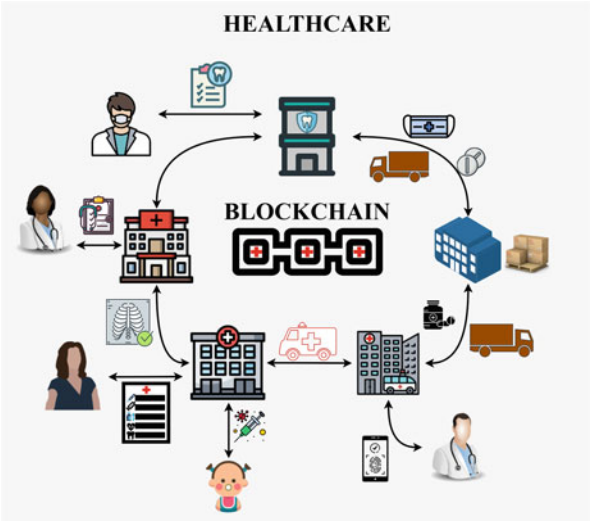


Fig. 12 Fully integrated healthcare system with blockchain

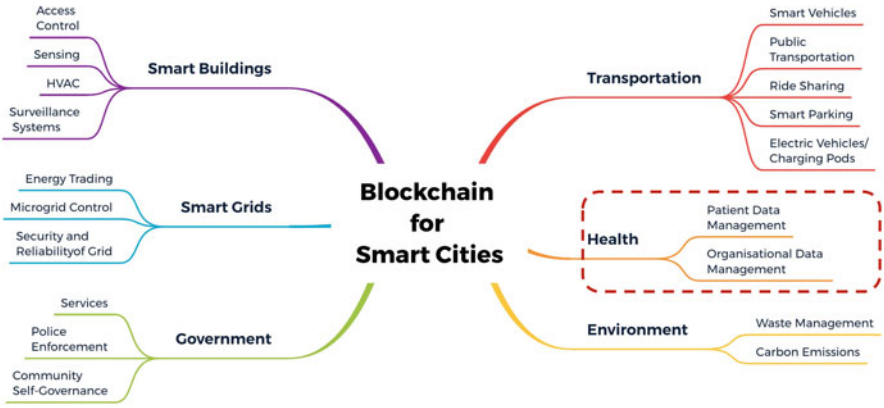


Fig. 13 Blockchain application in smart healthcare

Using blockchain technology to maintain EHR can help in addressing the issues of misdiagnosis, which may result in patients receiving incorrect treatments and medicines. This is particularly true in emergencies that require fast treatment. Updated and correct history of the patient can help doctors and nurses to avoid an incorrect treatment (e.g. for patients with diabetes or allergic reactions to specific drugs).

Blockchain can provide many benefits to data stored, in special due to its decentralised characteristics and cryptography adopted (e.g. digital signatures and hash). Smart contract is an important feature that blockchain can support and can help to perform smart decisions and to produce reliable information.

For example, smart contracts can be used to provide a data sharing system in accordance with regulations and law [89], maintaining and updating EHR [90] or even performing the treatment payments [91]. Additionally, smart contracts can help to define the sharing of the data or direct payments.

The remote patient monitoring should ensure that the data produced is stored and privacy is preserved. For example, Protected Health Information (PHI) produced by smart devices can be handled by smart contracts [92]. Also, smart contracts can help to manage information about telemedicine, in particular virtual health consultation [93] and telesurgery [94]. Telemedicine is being largely used (particularly after COVID-19), and the secure storage and sharing of the information produced are critical.

Another usage of blockchain is to provide incentives to help people to improve health and life quality, for example, designing a blockchain application that leverages smart contracts to release tokens for patients when performing healthy activities (gyms and other paid physical activities, running/cycling apps, receiving vaccination jabs, etc.). Patients can then use the received tokens to buy medicines or in other healthcare expenses [95].

Future Directions Privacy is one of the issues that have to be addressed while managing patient information in healthcare systems. Patient data should be shared

only with the consent of the patient, but many systems delegate the responsibility to blockchain management. Alternative mechanisms can be used to help in this procedure, such as zero-knowledge proof algorithms (see Sect. 3.3).

2.5.2 Organisational Data Management

Considering organisational data management, blockchain can be used to avoid counterfeit drugs in the healthcare systems and to control any other problem in the supply chain of equipment for healthcare [96, 97]. This is a big issue in many countries. For example, during the COVID-19 pandemic, many health institutions worldwide received counterfeit drugs and medical supplies [98].

Additionally, blockchain can be used in drug traceability and regulations procedures. Many countries adopt strict regulations and protocols to approve new drugs. Blockchain and smart contracts have been presented as a solution to help drugs traceability and regulation. The usage of blockchain can offer tamper-resistance and non-repudiation to the data about the workflow/process for new drugs approval [99].

Using blockchain can help hospitals and health institutions to cooperate and share experiences about treatments and procedures [100]. In addition, it can help to improve the quality of the public health system, using smart contracts to improve logistics, human resources management and disease surveillance systems [101].

Blockchain can help in important validation/signature issues present in hospitals from different countries. For example, Brazil had many issues with staff using fake silicone fingers to register doctors' presence [102]. Also, counterfeit vaccination cards and counterfeit COVID-19 test results are being used in different countries to avoid border control and other health restrictions [103]. Thus, using blockchain can help to avoid unsigned or tampered data in the health sector.

Future Directions The amount of data produced and maintained in health institutions is a real issue to use blockchain. Hospitals have to store logs about the usage of drugs, human resources, dealers, equipment, etc. Regulations use to enforce institutions to store a large amount of data for a long period of time. Even using lightweight blockchains, the amount of data is a problem that still exists in the health sector. Similar to patient data management, privacy is a serious concern in the health sector, particularly in health institutions. Controlled drug usage, critical and private exams/tests information and human resources data must not be exposed to unauthorised people due to legal and ethical reasons.

2.6 Environment

The environment is a theme that has been highlighted as critical for many governments and organisations. The current lifestyle is causing a considerable environmental impact. Thus, legislation and awareness campaigns are being promoted to make everyone aware of pollution, climate change, etc. A smart city has to provide

solutions the rational use (and reuse) of resources. At the same time, a smart city should promote the reduction of CO_2 emissions and the usage of renewable energy (as presented previously in Sect. 2.2. Figure 14 shows examples of smart solutions, such as usage of cars and factories with reduced CO_2 emissions and the usage of renewable energy.

In this context, we explore the waste management and carbon footprint management for smart cities (see Fig. 15) and how the blockchain can address some challenges in this area.

2.6.1 Waste Management

Among the existing problems in big urban centres, the waste is one that draws attention to the city management due to the high environmental impact. Most of these centres adopt selective waste collection to enable and incentivise recycling, thus avoiding the increase in pollution. Improper waste management leads large urban centres to increase the pollution, contaminating water, soil and air, and could even generate an impact on the dwellers' health. Most of the cities adopt legislation regarding waste management, aiming to address or at least mitigate this problem [104]. Waste management is composed of different phases, such as collection, transport, classification and disposal. In each of these phases, some technology can be applied, aiming to improve the efficiency. In this context, blockchain technology has been evaluated as a possible technology to tackle some

Fig. 14 Example of smart environment solutions

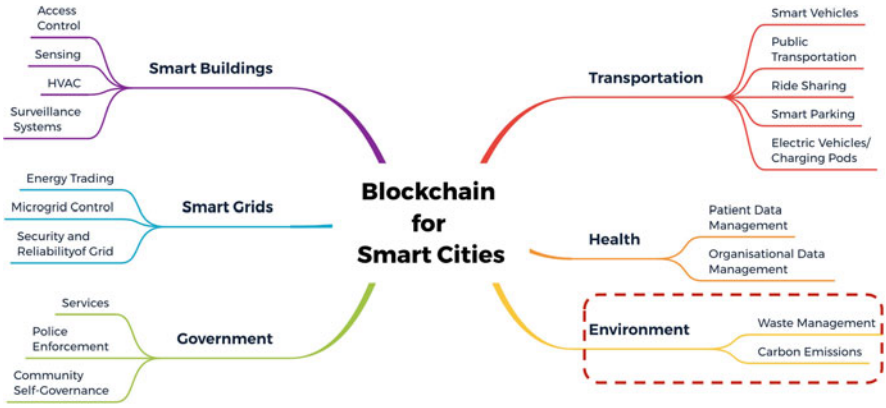


Fig. 15 Blockchain application in a smart environment

of the problems present in these phases, especially due to some similarities with the supply chain, in which blockchain has many applications.

The authors in [105] introduced a blockchain-based solution to handle the waste in smart cities. The research proposes a smart waste management system in which the users pay per amount of waste produced. The solution uses Ethereum smart contracts to manage the payment by introducing a custom cryptocurrency. The goal of using the blockchain technology is to reduce the cost of maintaining the system, once it relies on the public Ethereum network.

Focused only on e-waste, the authors in [106] use the blockchain technology to monitor e-waste collection and recycling. The solution is based on Ethereum. The government is responsible for deploying the smart contracts, authorises produces and retailers that can join the blockchain and is accredited to manage the e-waste. The proposed solution aims to introduce transparency and trust between interacting parties.

Future Directions Despite some overlaps of using blockchain for supply chain can also be applied for waste management, there are still many research opportunities in the waste management area. The most notable is integrating IoT sensors such as trash bins and monitoring/defining collecting truck routes. In these scenarios, the blockchain is a distributed solution that enables information sharing and can even provide incentives to the solution users.

2.6.2 Carbon Emissions

Climate change and global warming are great challenges facing humanity today, and cities are responsible for a large share of the carbon emissions [107]. Smart

cities can play a crucial role in reducing greenhouse gas emissions through utilising renewable energy resources, sustainable and environmentally friendly policies and infrastructure and the influence on residents' lifestyle choices. As described in the previous sections, smart grids can enable the shift to renewable energy resources (Sect. 2.2) and improve the efficiency of power grids; smart transportation (Sect. 2.4) and smart buildings (Sect. 2.1) can reduce carbon emissions.

Recently, blockchain-based initiatives have been launched to develop platforms to reduce or offset carbon emissions. Carbon emission trading is a market mechanism that aims to reduce carbon emissions by allowing the entities to buy or sell carbon emission allowances depending on their emission levels and the allowed emission caps. Entities can also buy carbon offsets to reduce their carbon footprints by funding green projects that reduce global emissions. Blockchain technology can improve carbon trading transparency, trust and complexity by facilitating an immutable, traceable and verifiable digital registry for carbon credits. Furthermore, smart contracts can be utilised to prevent the entities from hoarding carbon credits by enforcing expiry timers for the credits traded [108].

Blockchain-based solutions have also been proposed to incentivise renewable energy production to reduce carbon emissions. For instance, SolarCoin [109] is a blockchain-based solar energy platform that incentivises solar energy producers with free tokens that can be traded or converted to other currencies.

Future Directions Although blockchain can enable mechanisms to reduce carbon emissions, blockchains may be inefficient due to the redundancies. Furthermore, computationally expensive consensus mechanisms consume large amounts of energy and cause high carbon emissions. Future research should focus on lightweight consensus mechanisms and efficient data storage.

3 Security Aspects

In this section, we focus on the security of smart cities. First, we highlight some of the security and privacy issues that could threaten smart cities. Then, we define important security properties to consider in smart cities as well as discuss how blockchain technology could play a role in achieving these properties and, thus, securing smart cities.

3.1 Security and Privacy Issues in Smart Cities

Smart cities play a significant role in enhancing the lives of citizens within a society. In smart city-based applications, different kinds of data are collected from various sources to be stored and analysed [110]. Some data could come from sensors that monitor the electrical grid, traffic, air and water quality as well as other

public utilities. Other data might come from public transportation, libraries, parking meters, camera or waste management.

Data is an essential asset in smart cities as it may include, but is of course not limited to, location coordinates, medical records, travelling history and credit/debit card numbers. Such data can be exploited and forged by unauthorised people to inject false data or to gain access to sensitive information [111].

Data security and privacy are thus among the major concerns and challenges that smart city-based applications face [112]. It is essential to protect resident data from different attacks, as well as to preserve the privacy of such data. This implies challenges from a technological perspective, as we discuss in this chapter, but also implies challenges concerning business continuity processes and risk assessment and management, as discussed in detail in the context of smart buildings in [113].

Smart applications can be vulnerable to several attacks such as collision attacks, eavesdropping attacks, background knowledge attacks, spam attacks, forgery attacks, data vandalism attacks and Sybil attacks [114]. An example of such attacks is the electricity disconnection that affected over 200 thousand Ukrainian citizens in 2015 due to the power grid system being attacked [112]. In addition to potential attacks, citizens' data can be subject to privacy threats since service providers and third parties often collect it. For example, service providers of smart healthcare and home applications might gain access to sensitive data [115]. Similarly, residents' living habits such as working hours can be monitored by the smart metering infrastructure in smart grid applications [116]. Besides, smart mobility applications can collect trajectory information of residents and use it to predict their location coordinates and mobility patterns [117].

3.2 Security Properties for Smart Cities

Several security properties or requirements are essential to consider in the design of applications related to smart cities. In this section, we consider three important security properties to adopt in smart applications, which are data confidentiality, data integrity and authenticity. Such properties are to secure smart applications through protecting citizen data from being accessed or altered by unauthorised parties as well as through ensuring that data always comes from the right and trusted sources.

Data Confidentiality It refers to the ability to keep information secret by hiding it from unauthorised access or read. Confidentiality involves a set of rules and restrictions that control access to certain kinds of information. In smart cities, various data must be kept secret, including sensitive and personal information such as healthcare records as well as living habits and locations of the citizens. Unauthorised people should not be able to access such data. The challenge is how to ensure that only authorised access is permitted throughout the entire lifespan of the data. For instance, data should be protected during transmission in the network

to prevent eavesdropping attacks. Also, it should remain secret while it is stored in the database or while it is being processed and analysed.

Confidentiality property directly impacts the privacy of smart city citizens. Therefore, it is essential to assure the secrecy of the citizens' data by preventing unauthorised access to the data. A well-known approach to achieve confidentiality property is through the utilisation of cryptographic encryption techniques. Data can be encrypted or translated into a different form using a cryptographic key to make it hard to understand the original data. Only authorised people who maintain the key can recover and access the original data.

Data Integrity Another essential property for a secure smart application is data integrity. It refers to the ability to protect the data from being altered, except by authorised people. An unauthorised party should not be able to modify data either during transmission or after storing it in the database. Any such modification (either intentional or accidental) would violate the data integrity property as it might lead to unpleasant consequences. Altering the data to inject false information could lead to incorrect decisions and results. Forging healthcare records, for instance, may result in a patient receiving the wrong prescription, putting the patient's health under severe risk [118].

In smart cities, data integrity violations can take different forms of how the content of data could be changed. An example of such violations can be the alteration of a vehicle's speed either through reducing it to be within the speed limit or increasing it beyond the limit [110]. Another example is the change of the vehicle speed to be a negative value, which can be considered as a type of data vandalism as the data, in this case, does not make any sense. Such violations can impact the decisions and penalties applied within smart cities and, therefore, may require the need for establishing more strict speed limits or the presence of police patrolling.

It is essential to enable data integrity checking techniques in smart cities in order to ensure that the data is always intact and valid. There are different cryptographic approaches for verifying the integrity of the data. The basic approach is to create metadata (extra information about the data) to be linked and attached to the original data. Such metadata makes it easy to detect any alteration to the original data, and thus, data cannot be falsified without being noticed. The hash value of the data can be used as metadata to ensure the integrity of the data as any modification to the data would result in a different hash value. Message Authentication Code (MAC) is another approach that defines a secret key (a short piece of information) to detect alteration of the data. Any authorised person who has the correct private key can verify the integrity of the data.

Authenticity It refers to the ability to verify and guarantee that the data comes from the right source (i.e. the data belongs to the person who claimed to be). In other words, authenticity means that the identity of devices or citizens is not impersonalised. Authenticity is an essential security property in smart cities, as there are a large amount of data circulated. This data comes from different sources,

including but not limited to government and citizens, sensors, vehicles and utilities. A malicious attacker may ruin the system by creating falsified data on behave of a data source, which can be considered as legitimate data by others (systems, users, applications, etc.).

Consequently, it is important to consider technological approaches to efficiently and securely authenticate the authorship of such data to ensure it is always created by the right entity, and thus, it can be trusted. Authenticity property, however, does not prevent a faulty sensor or device from generating invalid data. Cryptographic primitives such as *public-key cryptography* are widely used for achieving the authenticity property [119].

3.3 Securing Smart Cities Through Blockchain Technology

There are some technological approaches to secure smart cities, including the use of cryptography, access control and blockchain technology. Since the focus of this chapter is on the application of blockchain technology to smart cities, we limit our discussion in this section to blockchain technology. In particular, we discuss in more detail how blockchain technology could play a significant role in improving the security aspects of smart cities, especially concerning data confidentiality, data integrity and authenticity.

Blockchain technology has received huge attention recently. A blockchain is a distributed and public verifiable ledger that records all transactions that have occurred in the peer-to-peer network. Using blockchain, non-trusted parties can securely communicate and exchange data and assets without involving trusted third parties. Blockchain is a set of linked blocks, where each block has a unique identifier and it comprises a set of transactions. Every block is linked to the preceding block, resulting in a chain of blocks [120]. Once a block is appended to the blockchain ledger, it would be hard to forge or modify its transactions as one needs to modify or rewrite all subsequent blocks [121]. This is what makes blockchain technology an excellent approach for securing smart applications.

Data Confidentiality This security property is not explicitly considered in most popular blockchain implementations, such as Bitcoin and Ethereum. This is because such implementation mainly focuses on making data publicly available and accessible to anyone inspecting or verifying its correctness, contradicting with the confidentiality property that the main purpose is to restrict access to data. Instead, such blockchain implementation relies on an anonymity model to hide the identity of the network nodes. Every node is assigned to a unique address to use as an alternative to its real identity. However, this anonymity model does not preserve the privacy of transactions since it is feasible to trace all transactions submitted by a particular node through inspecting its address [122].

In the literature, however, there are some blockchain platforms (e.g., Zcash, Monero and Dash) that were mainly designed to preserve the privacy of transactions and data. Such blockchains leverage cryptographic techniques (e.g. zero-knowledge proof algorithms) to make it infeasible to trace the submitter and the recipient of the data or to access the content of the data [123]. In Monero, for instance, nodes are instructed to use a different address for each transaction to make it hard to trace transactions back to an individual node [124].

Data confidentiality and privacy can be highly desirable in smart city-based applications that involve sensitive data that must remain secret as the case in smart healthcare. When leveraging blockchain as a secure solution to such smart applications, it is essential to consider blockchain solutions that strongly support the confidentiality property.

Data Integrity Data integrity is one of the main elegant security properties that blockchain is often advertised for [110]. Blockchain offers a strong data integrity property through an immutable and distributed ledger that is hard to alter, as mentioned earlier in this section. Blockchain uses a data structure technique named *Merkle tree* to securely manage and store data as well as to verify its correctness. The Merkle tree relies on the cryptographic hash function to construct a binary tree that consists of a set of nodes. At the bottom of the tree are the leaf nodes that contain the underlying data. At the top of the tree is the root hash of the entire data, which is recorded in the blockchain for future data integrity verification. Any modification to a piece of data stored in a leaf node would result in a different root hash, making it easy to detect data integrity violations.

Although blockchain provides an efficient way for data integrity verification, several attacks can threaten the integrity of the data and the system as a whole. One of the most well-recognised attacks is the *double-spending* attack that the main purpose is to spend the same coins in two or more transactions. This attack can be alleviated by waiting for a minimum period of time before confirming the data and making a response (i.e. exchanging the goods) [125]. However, this mitigation solution might not be practical or acceptable in applications where decisions and responses must be made as quick as possible [126].

That is, smart cities considering blockchain should be aware of blockchain developments and potential attacks that may threaten the integrity of data. To prevent such attacks, it is essential to consider some effective countermeasures.

Authenticity This property is achieved in most blockchain systems such as Bitcoin through the usage of public-key cryptography (e.g. Elliptic Curve Digital Signature Algorithm) [127]. In such blockchains, every user is assigned to a pair of keys (private and public) to establish trust between the submitter and the recipient of messages. The private key of a user is to create and sign a new transaction or data, and thus, it must be kept secret. If the private key of a user is compromised, then anyone who has the key can generate a transaction on behave of that user. The public key is shared with the network users to use when authenticating the author of the data.

4 Chapter Summary

Smart cities have to deal with heterogeneous devices, different applications, a high volume of information produced and data analytic in a dynamic and untrusted environment. Blockchain technology can help to deal with all these issues. Particularly, some blockchain characteristics such as decentralisation, tamper-resistance and resilience can help to increase the security of data and users.

In this chapter, we identified the most relevant smart cities applications. In each of these applications, we highlighted the issues that blockchain technology has the potential to help in smart cities ecosystems. Additionally, we presented open issues and possible future research in these areas. Finally, we discussed the main security aspects present in the blockchain application for smart cities.

Acknowledgments This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001. Avelino F. Zorzo is supported by CNPq (315192/2018-6) and FAPERGS. This work was supported by the INCT Forensic Sciences through the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq − process # 465450/2014-8). Also, Roben C. Lunardi was supported by IFRS.

References

1. United Nations. World urbanization prospects the 2018 revision. https://www.un.org/development/desa/pd/sites/www.un.org.development.desa.pd/files/files/documents/2020/Jan/un_2018_wup_report.pdf (2020)
2. A. Gharaibeh, M.A. Salahuddin, S.J. Hussini, A. Khreishah, I. Khalil, M. Guizani, A. Al-Fuqaha, Smart cities: A survey on data management, security, and enabling technologies. IEEE Commun. Surv. Tutorials **19**(4), 2456–2501 (2017)
3. R.R. Harmon, E.G. Castro-Leon, S. Bhide, Smart cities and the Internet of Things, in *2015 Portland International Conference on Management of Engineering and Technology (PICMET)* (2015), pp. 485–494
4. F.D. Moya, J.L. Torres-Moreno, J.D. Álvarez, Optimal model for energy management strategy in smart building with energy storage systems and electric vehicles. Energies **13**(14), (2020)
5. M. Conoscenti, A. Vetro, J.C. De Martin, Blockchain for the internet of things: A systematic literature review, in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)* (2016), pp. 1–6
6. V. Dedeoglu, A. Dorri, R. Jurdak, R.A. Michelin, R.C. Lunardi, S.S. Kanhere, A.F. Zorzo, A journey in applying blockchain for cyberphysical systems, in *International Conference on COMmunication Systems NETworkS (COMSNETS)* (2020), pp. 383–390
7. A.F. Zorzo, H.C. Nunes, R.C. Lunardi, R.A. Michelin, S.S. Kanhere, Dependable IoT using blockchain-based technology, in *Latin-American Symposium on Dependable Computing* (2018), pp. 1–9
8. A. Dorri, S.S. Kanhere, R. Jurdak, Towards an Optimized BlockChain for IoT, in *2017 Second International Conference on Internet-of-Things Design and Implementation (IoTDI)* (ACM, New York, 2017), pp. 173–178
9. R.C. Lunardi, R.A. Michelin, C.V. Neu, A.F. Zorzo, Distributed access control on IoT ledger-based architecture, in *IEEE/IFIP Network Operations and Management Symposium (NOMS)* (2018), pp. 1–7

10. M.A. Islam, S. Madria, A permissioned blockchain based access control system for IoT, in *2019 IEEE International Conference on Blockchain (Blockchain)* (2019), pp. 469–476
11. V. Dedeoglu, R. Jurdak, A. Dorri, R.C. Lunardi, R.A. Michelin, A.F. Zorzo, S.S. Kanhere, Blockchain technologies for IoT, in *Advanced Applications of Blockchain Technology* (Springer, Singapore, 2020), pp. 55–89
12. O. Novo, Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE Internet Things J. **5**(2), 1184–1195 (2018)
13. H.C. Nunes, R.C. Lunardi, A.F. Zorzo, R.A. Michelin, S.S. Kanhere, Context-based smart contracts for appendable-block blockchains, in *IEEE International Conference on Blockchain and Cryptocurrency* (2020)
14. I.G. Johnson, A. Puussaar, J. Manuel, P. Wright, Neighbourhood data: Exploring the role of open data in locally devolved policymaking processes. *Proceedings of the ACM on Human-Computer Interaction*, vol. 2(CSCW) (2018)
15. A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014)
16. M. Alharby, A. van Moorsel, BlockSim: an extensible simulation tool for blockchain systems. Front. Blockchain **3**, 28 (2020)
17. R.C. Lunardi, R.A. Michelin, C.V. Neu, H.C. Nunes, A.F. Zorzo, S.S. Kanhere, Impact of consensus on appendable-block blockchain for IoT, in *16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)* (2019), pp. 228–237
18. R.A. Michelin, R.C. Lunardi, H.C. Nunes, V. Dedeoglu, C.V. Neu, A.F. Zorzo, S.S. Kanhere, Appendable-block blockchains: Overview, applications, and challenges, in *Enabling Blockchain Technology for Secure Networking and Communications* (2021), pp. 66–88
19. A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, Lsb: A lightweight scalable blockchain for IoT security and anonymity. J. Parallel Distrib. Comput. **134**, 180–197 (2019)
20. M. Samaniego, R. Deters, Blockchain as a service for IoT, in *IEEE International Conference on Internet of Things (iThings)/IEEE Green Computing and Communications (GreenCom)/IEEE Cyber, Physical and Social Computing (CPSCom)/IEEE Smart Data (SmartData)* (2016), pp. 433–436
21. M. Alharby, R.C. Lunardi, A. Aldweesh, A. van Moorsel, Data-driven model-based analysis of the ethereum verifier's dilemma, in *Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (IEEE, New York, 2020), pp. 209–220.
22. A. Ghofrani, S.D. Nazemi, M.A. Jafari, HVAC load synchronization in smart building communities. Sustainable Cities Soc. **51**, 101741 (2019)
23. R.C. Lunardi, M. Alharby, H.C. Nunes, C. Dong, A.F. Zorzo, A. van Moorsel, Context-based consensus for appendable-block blockchains, in *IEEE International Conference on Blockchain* (2020), pp. 401–408
24. L. Bindra, C. Lin, E. Stroulia, O. Ardakanian, Decentralized access control for smart buildings using metadata and smart contracts, in *2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)* (2019), pp. 32–38
25. S.N. Jyothi, K.V. Vardhan, Design and implementation of real time security surveillance system using IoT, in *2016 International Conference on Communication and Electronics Systems (ICCES)* (2016), pp. 1–5
26. D. Nagothu, R. Xu, S.Y. Nikouei, Y. Chen, A microservice-enabled architecture for smart surveillance using blockchain technology, in *2018 IEEE International Smart Cities Conference (ISC2)* (2018), pp. 1–4
27. R. Wang, W. Tsai, J. He, C. Liu, Q. Li, E. Deng, A video surveillance system based on permissioned blockchains and edge computing, in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)* (2019), pp. 1–6
28. J. Benet, IPFS—content addressed, versioned, P2P file system. CoRR, abs/1407.3561 (2014)

29. R.A. Michelin, N. Ahmed, S.S. Kanhere, A. Seneviratne, S. Jha, Leveraging lightweight blockchain to establish data integrity for surveillance cameras, in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2020), pp. 1–3
30. O.B. Mora, R. Rivera, V.M. Larios, J.R. Beltrán-Ramírez, R. Maciel, A. Ochoa, A use case in cybersecurity based in blockchain to deal with the security and privacy of citizens and smart cities cyberinfrastructures, in *2018 IEEE International Smart Cities Conference (ISC2)* (2018), pp. 1–4
31. J. Xie, H. Tang, T. Huang, F.R. Yu, R. Xie, J. Liu, Y. Liu, A survey of blockchain technology applied to smart cities: research issues and challenges. IEEE Commun. Surv. Tutorials **21**(3), 2794–2830 (2019). Conference Name: IEEE Communications Surveys Tutorials
32. K. Moslehi, R. Kumar, A reliability perspective of the smart grid. IEEE Trans. Smart Grid **1**(1), 57–64 (2010)
33. X. Fang, S. Misra, G. Xue, D. Yang, Smart grid—the new and improved power grid: A survey. IEEE Commun. Surv. Tutorials **14**(4), 944–980 (2012)
34. N. Wang, X. Zhou, X. Lu, Z. Guan, L. Wu, X. Du, M. Guizani, When energy trading meets blockchain in electrical power system: the state of the art. Appl. Sci. **9**(8), 1561 (2019)
35. Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, Y. Zhang, Consortium blockchain for secure energy trading in industrial Internet of Things. IEEE Trans. Ind. Inf. **14**(8), 3690–3700 (2017)
36. K. Gai, Y. Wu, L. Zhu, M. Qiu, M. Shen, Privacy-preserving energy trading using consortium blockchain in smart grid. IEEE Trans. Ind. Inf. **15**(6), 3548–3558 (2019)
37. A. Dorri, F. Luo, S.S. Kanhere, R. Jurdak, Z.Y. Dong, SPB: A secure private blockchain-based solution for distributed energy trading. IEEE Commun. Mag. **57**(7), 120–126 (2019)
38. C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, M. Bertoncini, Blockchain based decentralized management of demand response programs in smart energy grids. Sensors **18**(1), 162 (2018)
39. P. Danzi, M. Angjelichinoski, Č. Stefanović, P. Popovski, Distributed proportional-fairness control in microgrids via blockchain smart contracts, in *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)* (IEEE, New York, 2017), pp. 45–51
40. D. Minoli, B. Occhiogrosso, Blockchain mechanisms for IoT security. Internet of Things **1**, 1–13 (2018)
41. G. Liang, S.R. Weller, F. Luo, J. Zhao, Z.Y. Dong, Distributed blockchain-based data protection framework for modern power systems against cyber attacks. IEEE Trans. Smart Grid **10**(3), 3162–3173 (2019)
42. D.E. Olivares, A. Mehrizi-Sani, A.H. Etemadi, C.A. Cañizares, R. Iravani, M. Kazerani, A.H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, G.A. Jiménez-Estévez, N.D. Hatziargyriou, Trends in microgrid control. IEEE Trans. Smart Grid **5**(4), 1905–1919 (2014)
43. S. Noor, W. Yang, M. Guo, K.H. van Dam, X. Wang, Energy demand side management within micro-grid networks enhanced by blockchain, Appl. Energy **228**, 1385–1398 (2018)
44. Y. Mo, T.H.-J. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, B. Sinopoli, Cyber–physical security of a smart grid infrastructure. Proc. IEEE **100**(1), 195–209 (2011)
45. X. Zhang, M. Fan, Blockchain-based secure equipment diagnosis mechanism of smart grid. IEEE Access **6**, 66165–66177 (2018)
46. M. Devetsikiotis, K. Christidis, Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
47. N. Szabo, Formalizing and securing relationships on public networks. First Monday **2**(9) (1997). https://firstmonday.org/ojs/index.php/fm/article/view/548
48. F.R. Batubara, J. Ubacht, M. Janssen, Challenges of blockchain technology adoption for e-government: a systematic literature review, in *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*, dg.o '18 (Association for Computing Machinery, New York, 2018)
49. M.-L. Marsal-Llacuna, The people's smart city dashboard (PSCD): Delivering on community-led governance with blockchain. Technol. Forecasting Social Change **158**, 120150 (2020)

50. M.-L. Marsal-Llacuna, Future living framework: Is blockchain the next enabling network? Technol. Forecasting Social Change **128**, 226–234 (2018)
51. C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, B. David, A literature survey on smart cities. Sci. China Inf. Sci. **58**(10), 1–15 (2015)
52. T. Martens, Electronic identity management in Estonia between market and state governance. Identity in the Information Society **3**(1), 213–233 (2010)
53. A.M. Aguilar Rivera, K. Vassil, *Estonia: A Successfully Integrated Population-Registration and Identity Management System* (World Bank, New York, 2015)
54. e-Estonia. e-Identity. https://e-estonia.com/solutions/e-identity/. [Accessed on 10-October-2020]
55. K. Siountri, E. Skondras, D.D. Vergados, Towards a smart museum using bim, iot, blockchain and advanced digital technologies, in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ICVISP 2019 (Association for Computing Machinery, New York, 2019)
56. J. Zeng, Y. Yuan, J. Zhang, Y. Liu, Blockchain in smart park: Application scheme design, in *Proceedings of the 2019 International Electronics Communication Conference*, IECC '19 (Association for Computing Machinery, New York, 2019), pp. 76–83
57. ABC News. $30 million art heist at stockholm museum. https://abcnews.go.com/International/story?id=81873&page=1 (2006). [Accessed on 13-October-2020]
58. The Guardian. Brazil museum fire: 'incalculable' loss as 200-year-old rio institution gutted. https://www.theguardian.com/world/2018/sep/03/fire-engulfs-brazil-national-museum-rio (2018). [Accessed on 13-October-2020]
59. Inês Faria. Trust, reputation and ambiguous freedoms: financial institutions and subversive libertarians navigating blockchain, markets, and regulation. J. Cult. Econ. **12**(2), 119–132 (2019)
60. Avi Spielman. *Blockchain: Digitally Rebuilding the Real Estate Industry*. PhD thesis (Massachusetts Institute of Technology, Cambridge, 2016)
61. P. Chen, B. Jiang, C. Wang, Blockchain-based payment collection supervision system using pervasive bitcoin digital wallet, in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (2017), pp. 139–146
62. N. Chowdhury, An IoT and blockchain-based approach for ensuring transparency and accountability in regulatory compliance, in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct (Association for Computing Machinery, New York, 2019), pp. 957–962
63. F. Tian, An agri-food supply chain traceability system for China based on RFID blockchain technology, in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)* (2016), pp. 1–6
64. C. Visser, Q.A. Hanich, How blockchain is strengthening tuna traceability to combat illegal fishing. Technical report (University of Wollongong, Wollongong, 2017)
65. R.A. Michelin, A. Dorri, M. Steger, R.C. Lunardi, S.S. Kanhere, R. Jurdak, A.F. Zorzo, Speedychain: a framework for decoupling data from blockchain for smart cities, in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '18 (Association for Computing Machinery, New York, 2018), pp. 145–154
66. J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, Y. Zhang, Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4660–4670 (2019)
67. A. Dorri, M. Steger, S.S. Kanhere, R. Jurdak, Blockchain: a distributed solution to automotive security and privacy. IEEE Commun. Mag. **55**(12), 119–125 (2017)
68. X. Zhang, X. Chen, Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network. IEEE Access **7**, 58241–58254 (2019)

69. L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, Z. Zhang, CreditCoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. IEEE Trans. Intell. Transp. Syst. **19**(7), 2204–2220 (2018)
70. M. Cebe, E. Erdin, K. Akkaya, H. Aksu, S. Uluagac, Block4forensic: an integrated lightweight blockchain framework for forensics applications of connected vehicles. IEEE Commun. Mag. **56**(10), 50–57 (2018)
71. C. Oham, R. Michelin, S.S. Kanhere, R. Jurdak, S. Jha, B-FERL: blockchain based Framework for Securing Smart Vehicles. arXiv preprint arXiv:2007.10528 (2020)
72. T.H. Nguyen, J. Partala, S. Pirttikangas, Blockchain-based mobility-as-a-service, in *2019 28th International Conference on Computer Communication and Networks (ICCCN)* (2019), pp. 1–6
73. D. Swinhoe, The 15 biggest data breaches of the 21st century. https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html (2020)
74. C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
75. M. Baza, N. Lasla, M. Mahmoud, G. Srivastava, M. Abdallah, B-Ride: ride sharing with privacy-preservation, trust and fair payment atop public blockchain. IEEE Trans. Network Sci. Eng. **8**(2), 1214–1229 (2019)
76. P. Pal, S. Ruj, BlockV: a blockchain enabled peer-peer ride sharing service, in *2019 IEEE International Conference on Blockchain (Blockchain)* (2019), pp. 463–468
77. C. Jaffe, C. Mata, S. Kamvar, Motivating urban cycling through a blockchain-based financial incentives system, in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17 (Association for Computing Machinery, New York, 2017), pp. 81–84
78. Park Me. *Park Me—find & Pay for the Perfect Spot*. https://www.parkme.com/ (2020)
79. Park Whiz. *Park Whiz—parking Just Got a Lot Simpler*. https://www.parkwhiz.com/ (2020)
80. Spot Hero. *Spot Hero—Find and Book Parking in Seconds*. https://spothero.com/ (2020)
81. W.A. Amiri, M. Baza, K. Banawan, M. Mahmoud, W. Alasmary, K. Akkaya, Privacy-preserving smart parking system using blockchain and private information retrieval, in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)* (2019), pp. 1–6
82. J.C. Ferreira, A.L. Martins, F. Gonçalves, R. Maia, A blockchain and gamification approach for smart parking, in *Intelligent Transport Systems, From Research and Development to the Market Uptake* (Springer International Publishing, Cham, 2019), pp. 3–14
83. Z. Su, Y. Wang, Q. Xu, M. Fei, Y. Tian, N. Zhang, A secure charging scheme for electric vehicles with smart communities in energy blockchain. IEEE Internet Things J. **6**(3), 4601–4613 (2019)
84. C. Liu, K.K. Chai, X. Zhang, E.T. Lau, Y. Chen, Adaptive blockchain-based electric vehicle participation scheme in smart grid platform. IEEE Access **6**, 25657–25665 (2018)
85. S. Jeong, N. Dao, Y. Lee, C. Lee, S. Cho, Blockchain based billing system for electric vehicle and charging station, in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)* (2018), pp. 308–310
86. M. Pustišek, A. Kos, U. Sedlar, Blockchain based autonomous selection of electric vehicle charging station, in *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)* (2016), pp. 217–222
87. N. Ahmed, R.A. Michelin, W. Xue, S. Ruj, R. Malaney, S.S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, S.K. Jha, A survey of COVID-19 contact tracing apps. IEEE Access **8**, 134577–134601 (2020)
88. A.A. Vazirani, O. O'Donoghue, D. Brindley, E. Meinert, Implementing blockchains for efficient health care: systematic review. J. Med. Internet Res. **21**(2), e12439 (2019)

89. A.R. Lee, M.G. Kim, I.K. Kim, Sharechain: healthcare data sharing framework using blockchain-registry and FHIR, in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2019), pp. 1087–1090
90. A. Shahnaz, U. Qamar, A. Khalid, Using blockchain for electronic health records. IEEE Access **7**, 147782–147795 (2019)
91. T.K. Dasaklis, F. Casino, C. Patsakis, Blockchain meets smart health: Towards next generation healthcare services, in *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)* (2018), pp. 1–8
92. K.N. Griggs, O. Ossipova, C.P. Kohlios, A.N. Baccarini, E.A. Howson, T. Hayajneh, Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. J. Med. Syst. **42**, 1–7 (2018)
93. R.W. Ahmad, K. Salah, R. Jayaraman, I. Yaqoob, S. Ellahham, M. Omar, The role of blockchain technology in telehealth and telemedicine. Int. J. Med. Inform. **148**, 104399 (2021)
94. R. Gupta, A. Shukla, S. Tanwar, Aayush: a smart contract-based telesurgery system for healthcare 4.0, in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)* (2020), pp. 1–6
95. R.C. Lunardi, H.C. Nunes, V. Branco, B. Lippert, C.V. Neu, A.F. Zorzo, Performance and cost evaluation of smart contracts in collaborative health care environments, in *Proceedings of the 14th International Conference for Internet Technology and Secured Transactions (ICITST-2019)* (2019), pp. 1–6
96. M. Mettler, Blockchain technology in healthcare: the revolution starts here, in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)* (2016), pp. 1–3
97. N. Saxena, I. Thomas, P. Gope, P. Burnap, N. Kumar, Pharmacrypt: blockchain for critical pharmaceutical industry to counterfeit drugs. Computer **53**(7), 29–44 (2020)
98. World Health Organization, Medical product alert nº 3/2020. https://www.who.int/news-room/detail/31-03-2020-medical-product-alert-n-3-2020 (2020)
99. Y. Huang, J. Wu, C. Long, Drugledger: a practical blockchain system for drug traceability and regulation, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (2018), pp. 1137–1144
100. D. Résière, D. Resiere, H. Kallel, Implementation of medical and scientific cooperation in the caribbean using blockchain technology in coronavirus (covid-19) pandemics. J. Med. Syst. **44**, 1–2 (2020)
101. S. Bhattacharya, A. Singh, M. Hossain, Strengthening public health surveillance through blockchain technology. AIMS Public Health **6**, 326–333 (2019)
102. L.J. González-Soler, M. Gomez-Barrero, L. Chang, A. Pérez-Suárez, C. Busch, Fingerprint presentation attack detection based on local features encoding for unknown attacks. IEEE Access **9**, 5806–5820 (2021)
103. E. Davis, *Global Black Market for Fake covid-19 Vaccination Cards Flourishing* (2021)
104. United Nations. Global waste management outlook. https://www.unenvironment.org/resources/report/global-waste-management-outlook (2020)
105. M. Lamichhane, O. Sadov, A. Zaslavsky, Smart waste management system: A fusion of IoT and blockchain technology, in *Dependable IoT for Human and Industry: Modeling, Architecting, Implementation* (2018), p. 351
106. N. Gupta, P. Bedi, E-waste management using blockchain based smart contracts, in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2018), pp. 915–921
107. D. Hoornweg, L. Sugar, C.L. Trejos Gámez, Cities and greenhouse gas emissions: moving forward. Environ. Urban. **23**(1), 207–227 (2011)
108. D. Patel, B. Britto, S. Sharma, K. Gaikwad, Y. Dusing, M. Gupta, Carbon credits on blockchain, in *2020 International Conference on Innovative Trends in Information Technology (ICITIIT)* (IEEE, New York, 2020), pp. 1–5

109. L. Johnson, A. Isam, N. Gogerty, J. Zitoli, *Connecting the Blockchain to the Sun to Save the Planet*. Available at SSRN 2702639 (2015)
110. G. Stromire I. Potoczny-Jones, Empowering smart cities with strong cryptography for data privacy, in *Proceedings of the 1st ACM/EIGSCC Symposium on Smart Cities and Communities* (ACM, New York, 2018)
111. D. Popescul, L.-D. Genete, Data security in smart cities: challenges and solutions. Informatica Economică **20**(1), 29–36 (2016)
112. L. Cui, G. Xie, Y. Qu, L. Gao, Y. Yang, Security and privacy in smart cities: Challenges and opportunities. IEEE Access **6**, 46134–46145 (2018)
113. R. Czekster, C. Morisset, A. van Moorsel, J. Mace, W. Bassage, J. Clark, *Security of Active Buildings* (Springer, New York, 2020)
114. R. Kitchin, Getting smarter about smart cities: Improving data privacy and data security. Technical report, (Data Protection Unit, Department of the Taoiseach, Dublin, 2016)
115. T. Gong, H. Huang, P. Li, K. Zhang, H. Jiang, A medical healthcare system for privacy protection based on IoT, in *2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)* (2015), pp. 217–222
116. S. Finster, I. Baumgart, Privacy-aware smart metering: A survey. IEEE Commun. Surv. Tutorials **17**(2), 1088–1101 (2015)
117. Z. Ning, F. Xia, N. Ullah, X. Kong, X. Hu, Vehicular social networks: Enabling smart mobility. IEEE Commun. Mag. **55**(5), 16–55 (2017)
118. T.A. Butt, M. Afzaal, Security and privacy in smart cities: issues and current solutions, in *Smart Technologies and Innovation for a Sustainable Future* (Springer, Berlin, 2019), pp. 317–323
119. U.H. Rao, U. Nayak, *Cryptography* (Apress, New York, 2014), pp. 163–181
120. M. Alharby, A. Aldweesh, A. van Moorsel, Blockchain-based smart contracts: a systematic mapping study of academic research (2018), in *Proceedings of the 2018 International Conference on Cloud Computing, Big Data and Blockchain* (IEEE, New York, 2018), pp. 1–6
121. M. Alharby, A. van Moorsel, Blockchain-based smart contracts: A systematic mapping study, in *Proceedings of the 4th International Conference on Computer Science and Information Technology* (AIRCC Publishing Corporation, Chennai, 2017), pp. 125–140
122. Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in *Proceedings of the 2017 IEEE International Congress on Big Data* (IEEE, New York, 2017), pp. 557–564
123. A. Maher, *Models and Simulation of Blockchain Systems*. PhD thesis, (School of Computing, Newcastle University, Newcastle upon Tyne, United Kingdom, 2020). Unpublished thesis
124. T.-T. Kuo, H. Zavaleta Rojas, L. Ohno-Machado, Comparison of blockchain platforms: a systematic review and healthcare examples. J. Am. Med. Inform. Assoc. **26**(5), 462–478 (2019)
125. A. Reyna, C. Martín, J. Chen, E. Soler, M. Díaz, On blockchain and its integration with IoT. challenges and opportunities. Futur. Gener. Comput. Syst. **88**, 173–190 (2018)
126. G.O. Karame, E. Androulaki, S. Capkun, Double-spending fast payments in bitcoin, in *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (2012), pp. 906–917
127. A.M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies* (O'Reilly Media, Inc., New York, 2014)

Decentralized Identity Management and Blockchains: Design Patterns and Architectures

Hye-young Paik, Yue Liu, Qinghua Lu, and Salil S. Kanhere

1 Introduction

The Internet, in essence, is a network of machine addresses, which allows the people or organisations who operate the machines to remain anonymous. Although the anonymity provided by the Internet was much celebrated initially in the 1990s, as the user base of the Internet grew exponentially, not knowing "who" is behind the machine has contributed to the increasing problems of cyber-crime, security and privacy breaches. The recent advances in Internet of Things and Artificial Intelligence further add to the challenges as the "smart things" can be thrown into the mix of the potential users, creating even more confusion around the idea of an identity behind the machines. Hence, establishing and managing user identities have become one of the central issues in the modern hyper-connected society as a pre-requisite to enabling prosperous digital service economy built on trust.

Ultimately, a digital identity should allow the users to prove their identity without having to present physical identity documents, enabling provision of and access to all forms of digital services. The idea of decentralised identity management stems from the observations that people, throughout their life, tend to carry many different senses of "identities" (or personas), and each identity is fused with different sets of personal artefacts overtime. For instance, Alice may be a college student, a

H.-y. Paik (✉) · S. S. Kanhere
University of New South Wales, Sydney, NSW, Australia
e-mail: h.paik@unsw.edu.au; salil.kanhere@unsw.edu.au

Y. Liu
University of New South Wales, CSIRO/Data61, Sydney, NSW, Australia
e-mail: yue.liu13@unsw.edu.au

Q. Lu
CSIRO/Data61, Sydney, NSW, Australia
e-mail: Qinghua.Lu@data61.csiro.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_15

patron at a public library, as well as an avid online gamer. Currently, Alice will have to manage these identities and associated data through multiple accounts and passwords on her own. The so-called "social login" schemes, the OpenID or OAuth protocols, have recently become popularised to alleviate the multiple passwords problem. However, such schemes still present many issues. The main identity providers of these schemes are the most prominent online service platform providers such as Facebook or Google. First, this creates a typical vendor lock-in situation where the user will not be able to leave the platform without losing the access to all other accounts. Furthermore, it elevates and concentrates the privacy and security issues for the user as these platforms become the centralised point of risks, not only from the external, cyber attackers but also from the data-hungry platform providers themselves as they are able to collate and hold control of the user's personal data over all identities.

The rest of the chapter is organised as follows: In Sect. 2, we will first introduce the definition of identity, identity management and its context in the age of digitised society. We will also present an overview of the evolution of the concept over the last few decades along with the development of the Internet. Section 3 presents available identity standards and relevant technology. Section 4 outlines the role of blockchains and implementation options for SSI systems. Section 5 then presents a systematic design analysis of SSI systems through design patterns, which is followed by a reference architecture proposal in Sect. 6. We summarise and conclude the chapter with discussion of future directions in Sect. 7.

2 Evolution of Identity Management

A thorough discussion on the fundamentals of *digitised* identities would require views from the disciplines beyond computer science and technology. In the age of "smart" Internet enabled by Internet of Things and Artificial Intelligence, ongoing and inclusive discussions around what a digital identity should mean and how it would operate in the digitised society are important to understand how the technology can lead the digital transformation process, but be still guided by the expectations of the society as a whole.

In this chapter, however, we use the terms and definitions as defined in the ISO/IEC information technology security and privacy document[1] to focus the discussion strictly on technological implementation of the concepts based on a common understanding in the field of identity management. We also use the terms *identity* and *digital identity* interchangeably.

[1] International Organization for Standardization/International Electrotechnical Commission, 24760-1, 2019-5 edition, https://www.iso.org/standard/77582.html

In what follows, we will introduce some definitions of identity and identity management, as well as how the concepts have evolved over time along with the development of the Internet.

2.1 What Is Digital Identity

First to discuss identity, we need to discuss the notion of *entity*. An entity has a recognisably distinct existence and can have a physical or logical embodiment. For example, an entity could be a person, organisation, a device or a SIM card or a web site. Entities are important because they are necessary to operate a system.

An *identity* is a set of attributes related to an entity. The attributes describe the properties of an entity such as address, telephone number, a MAC address or a domain name. An entity can have many identities and several entities can have the same identity.

For a system to make an identity-based decision, it needs to distinguish which identity the entity is using. For this purpose, the term *identifier* is used to refer to a set of attributes that uniquely characterise an identity within the system domain.

A *credential* is a representation of an identity and can act as an identifier for an entity. It is typically used for authentication to grant entry to a system domain for the entity. However, a credential can accommodate a wide range of identity attributes and can also be used by the entity to prove existence and uniqueness of the identity attributes contained in a credential. Some of the types of the attributes are as follows:

- On physical existence – biographical details, home address or device location
- The entity's evolution over time – skill certificates, awards, proof of installed applications
- Information assigned to the entity – title, role, social security number or passport number
- An object that represents identity information for the entity – passport, business card, vehicle registration

A credential can be associated with a unique identifier, and issuance of a credential can be recorded in a registry against its associated entity. Ultimately, managing of identities involves managing the lifecycle of identity information from its conception to deletion.

2.2 Evolution of Identity Models

As mentioned, the Internet was created with no explicit provision for identifying the entities for communication. It should be noted that this issue equally concerns both directions, that is, the *user* entity authenticating the *service provider* entity

and vice versa. To compensate this, a collage of ad hoc, partial solutions have been developed to create the identity layer. Hence, the ways to represent and manage identities have evolved along with the growth of the Internet in the last few decades. In this section, we briefly give an overview of the historical development of the identity management schemes and their shortfalls. We will then introduce the idea of decentralisation and sovereignty of identity.

2.2.1 The Centralised Identity Model

In the centralised model, depicted in Fig. 1, also known as the account-based model, identities are created, stored and managed by a central authority of one system domain (e.g. the company for which the entity works).

The entity shares a credential with each central authority to establish trusted identity. Hence, it is also referred to as the account-based or "silo-ed" model. The credential contains, typically, a user name and password. However, some authorities may solicit more, often very private attributes, such as date of birth or maiden name.

As an early and simple identity management model for the Internet, it is the most common and widely used method still today. However, this model presents many issues. First, it is not scalable, especially from the entity's viewpoint. The silo-ed identities lead to hundreds of accounts and lost or easy-to-guess passwords, which result in increased security and privacy risks.

Also, establishing a relationship between each central authority and the entity over the Internet this way does not truly authenticate both parties. Typically, the user entity is asked to trust the digital certificates issued by a third-party certificate authority as the identity of the service provider, which is open to a man-in-the-middle attack. The service provider entity can only rely on their own due diligence

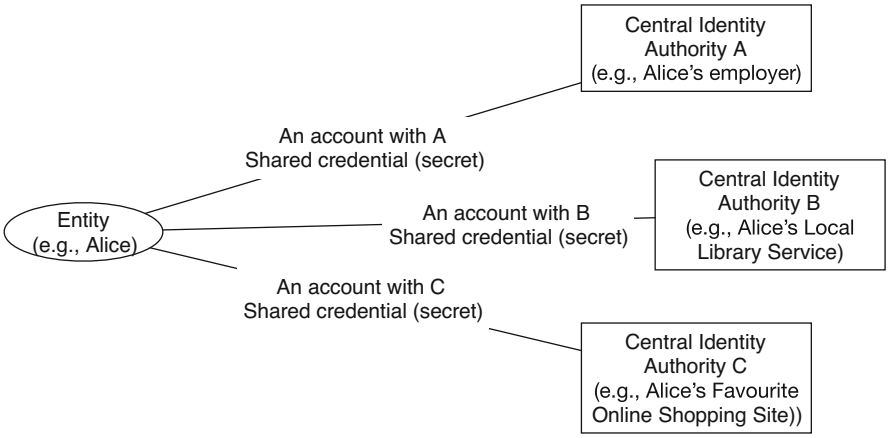


Fig. 1 The centralised identity model

and security expertise to trust the credential provided by the user. Furthermore, the problem of scalability is magnified if the model needs to incorporate a large number of smart devices.

From the privacy viewpoint, the centralised model does provide the entity much control of its own identity attributes, as any credentials collected are stored and managed by the central authority.

2.2.2 The Federated Identity Model

In a federated model, depicted in Fig. 2, identity owners can use the same identity to acquire services across different service provider(s) within a federation (e.g. National Science Foundation's FastLane allows users from participating organizations to log in to the system using their organization credentials). This reduces the number of digital identities that users have to manage and lowers the administrative costs of the service providers.

The core challenge of this model is how and by whom user identities are aggregated and resolved. The model introduces the concept of *Identity Providers* (IdP). Multiple identity providers are supported, allowing a distributed and partitioned storage of identity attributes for a single entity. Under this model, thus, a set of service providers and identity providers would form an alliance, referred to as a "circle of trust". Over the years, several schemes have been developed such as Shibboleth, Web service federation or SAML (Security Assertion Markup Language), each imposing different standards and system architectures to implement the model [1].

Although the user experience of authentication process is simplified under the federated model, much of the issues with the centralised model still persists. For

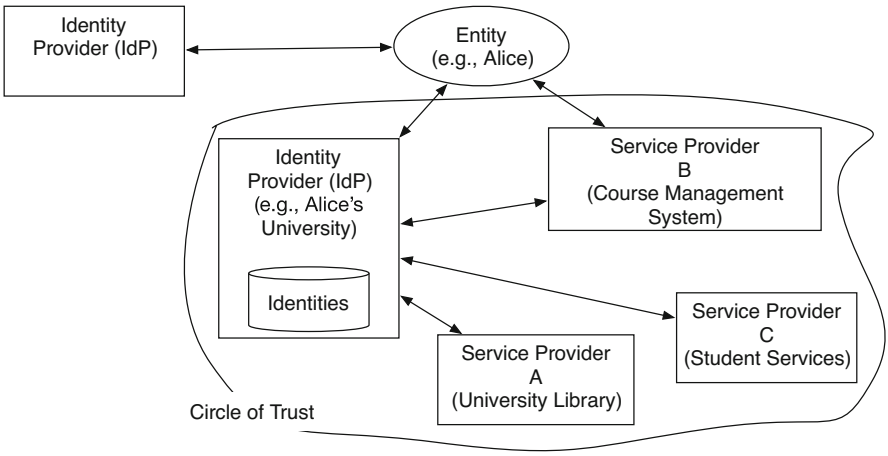


Fig. 2 The federated identity model

instance, an entity still has to have multiple accounts across the federated system domains. The authentication process is not bi-directional and remains open to man-in-the-middle attacks. Also, from the privacy viewpoint, although the identity information is distributed, the identity providers can correlate and link partial identity attributes on the same entity.

2.2.3 The User-Centric Identity Model

The *user-centric* model aims to shift the identity management paradigms from the identity providers to the users. The model recognises the fact that the existing solutions, either centralised or federated, do not give control of the identities and credential data (i.e. information shared with various identity providers for authorisation purposes) to the users. Many different standards and systems make the users not able to transfer one identity from one service provider to another, nor they are able to decide which identity attributes are shared with other service providers.

One of the most commonly used forms of the user-centric model is by the major social media services called "social log-in". This is when credentials from a social media site (e.g. Google, Twitter) can be used to log into other sites, as shown in Fig. 3. The protocols such as OAuth[2] provide the entity a way to authorise a third-party service provider to access the resources owned by the entity without sharing the credentials with the third party.

Despite the proposition, the current implementations of the models are still close to the federated model in that the identities are still stored by service providers. The emphasis is more on the data privacy and user experience of identity owners, allowing them to decide what kind of identity data can be shared to the service providers to some extent. Still the main identity providers of these schemes are the most prominent online service platform providers such as Facebook or Google; hence the typical issues with centralisation of identities still persist.

2.2.4 The Decentralised Identity Model

The latest view on identity models is centred around decentralisation in order to provide true and complete control of the identities and associated private data for an entity. In a radical paradigm shift from the existing and previous models, the decentralised identity model is not account-based; rather it introduces the concept of identity formed based on a direct relationship between two entities (i.e. peer-to-peer) without an involvement of centralised identity providers. As an equal participant of the relationship, the authentication process is bidirectional between the user entity and the service provider entity.

[2] OAuth/OAuth 2, https://oauth.net/2/

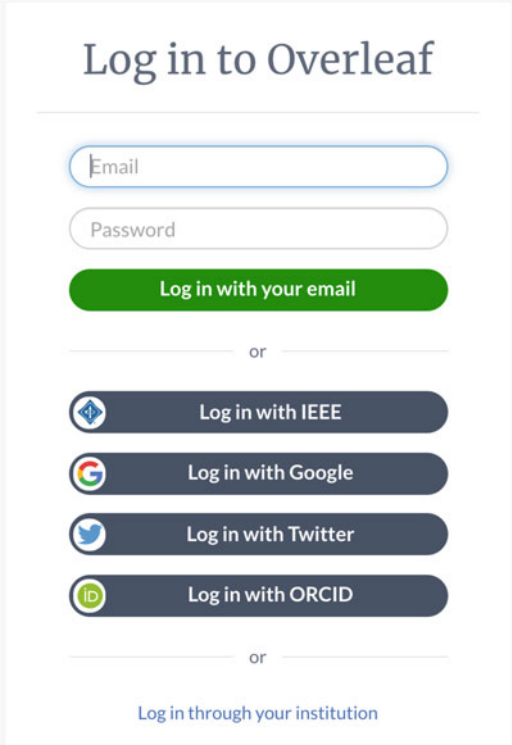


Fig. 3 The user-centric identity model – a social login example

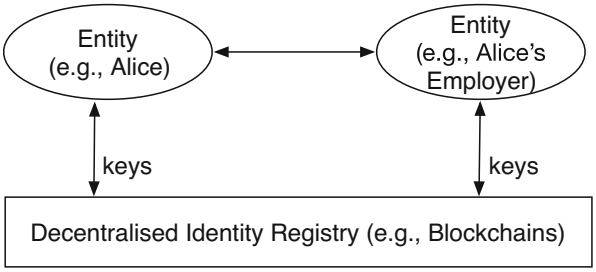


Fig. 4 The decentralised identity model

Unlike the previous models where identity is established per interaction session and lasts only during those interactions, in the decentralised model, the identities persist as long as the relationship persists. As depicted in Fig. 4, typically, identifiers for an entity are created using cryptographical means (e.g. public/private keys) rather than an identity attribute (e.g. email addresses, user names) and stored in a distributed, secured data store such as distributed ledgers. There are a few standards being developed to support the decentralised identity model. One of the prominent combinations of the technical implementation choices for the model is

DID (Decentralised Identifiers, by W3C) over blockchains. The later sections of the chapter discuss DID and blockchains, as well as other possible implementation choices.

2.3 The Self-Sovereign Identity Model

The term *self-sovereignty* in identity is used often and popularised by the blog post by Chris Allen [3] in 2016 and has become a moniker for the decentralised identity management concept. The objective of self-sovereign identity is to allow identity holders to keep real ownership of their identities and control over how their identity data is used. The concept of self-sovereign identity is increasingly popular as there is a growing attention in protecting data privacy (e.g. General Data Protection Regulation – GDPR [4]). Recently, both industry and academia have put effort in formalising the definitions and system requirements for self-sovereign identity (SSI) [6].

To define how the concept of "Self-Sovereignty" manifests itself in software systems with regard to managing identities, we summarise the requirements of self-sovereign identity systems based on Allen's principles [3]. The list below shows the overall functional requirements.

- Holders need to be able to register IDs for particular relationships with others.
- Holders need to be able to control their identities by updating the information associated with IDs and revoking IDs.
- Holders need to be able to control the disclosure of their identity facts on both the content and valid time period.
- Holders need to be able to access their identity data.
- The access to holders' identity data must be approved by the holders.

The list below shows the core quality properties that are required by self-sovereign identity systems.

- Data integrity: IDs and their ownership must not be changed by anyone.
- Transparency: All the historical operations on identity data must be transparent and traceable. Also, the code for business logic must be transparent.
- Persistence: All the identity data must be long lived as the holders wish until they decide to revoke the ID.
- Availability: The availability of identity data must be ensured.
- Security: The system shall ensure that the holders do not lose control of their IDs.

This book chapter focuses on introducing self-sovereign identity as an emerging form of decentralised identity model. As suggested in the requirements, in self-sovereign identity, identity owners must be central to the administration of identities. Implementing the SSI concept will require first establishing decentralised identifiers that do not rely on a centralised registry or identity provider and provide total

ownership of the user's identity. The next section explores digital identity standards including current options for representing decentralised identities.

3 Digital Identity and Related Standards

This section introduces some of the main digital identity-related standards that are applicable to realising the SSI concept. From the technical point of view, digital identity fundamentally is represented by digital identifiers, i.e. strings or tokens associated with an identity that are guaranteed to be unique within a given scope (e.g. within a country, within a university). Therefore, we will mainly refer to available digital identifier schemes and related standards.

3.1 Decentralised Identifiers (DID)

Self-sovereign identity continues the adoption of key pair mechanism, but in a more decentralised context, as a distributed system is needed for the registration of identities. To implement self-sovereign identity, the W3C Community Group specifies a standard of decentralised identifier (DID) [12], which contains human-readable information and can be used across different platforms. A DID is a URL that refers to an entity for trusted interactions, and each DID bonds to a DID document (DDO), which contains the data describing the DID subject, entity identified by corresponding DDO, with pre-defined mechanisms, such as public keys and pseudonymous biometrics. The DID subject can use these data to authenticate itself and prove its association with the DID. A simple DDO example given by W3C is presented, which has five properties: *context* expresses the environment of interaction between two DIDs; *id* is the DID described by this DDO; *verificationMethod* contains the public key for digital signatures and cryptographic operations; *service* denotes service endpoints that are used for interactions among DIDs; and *authentication* represents a kind of verification relationship to prove the current DID subject is valid. A self-sovereign identity implementation should provide DID methods for precise operations, including creation, verification, update and deactivation.

A DID and its corresponding DDO do not contain any identity information, which are stored in personal devices or database owned by the identity service providers. A credential [13] is a verifiable claim, carrying particular identity information attributes that an issuer provides to attest to some specific characteristics of an entity. Within self-sovereign identity, entities are able to issue digitally signed credentials about themselves and others linked to their DIDs, and these credentials can be verified by anyone else. A successful verification can append the trust of verifier on the identity attributes, which eventually forms a chain of trust.

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "verificationMethod": [{
    "id": "did:example:123#ZC2jXTO6t4R501bfCXv3RxarZyUbdP2w_psLwMuY6ec",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
    "id": "did:example:123456789abcdefghi#vcr",
    "type": "CredentialRepositoryService",
    "serviceEndpoint": "https://repository.example.com/service/8377464"
  }],
  "authentication": [{
    "id": "did:example:123#z6MkecaLyHuYWkayBDLw5ihndj3T1m6zKTGqau3A51G7RBf3",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123",
    "publicKeyBase58": "AKJP3f7BD6W4iWEQ9jwndVTCBq8ua2Utt8EEjJ6Vxsf"
  }]
}
```

Listing 1: DDO for DID did:example:123456789abcdefghi

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<> a foaf:PersonalProfileDocument ;
   foaf:maker <#me> ;
   foaf:primaryTopic <#me> .

<#me> a foaf:Person ;
   foaf:name "Bob" ;
   foaf:knows <https://example.edu/p/Alice#MSc> ;
   foaf:img <https://bob.example.org/picture.jpg>
```

Listing 2: WebID https://bob.example.org/profile#me

3.2 Web Identity and Discovery (WebID)

A WebID, a currently W3C standard working draft, is essentially an HTTP URI which refers to an entity (person, organization, device, etc.). Similarly to the DID scheme, a WebID can be associated with a profile document which describes the relevant entity. The operation and management of WebIDs rely on a set of standard, widely used Web components such as URI for the identifiers, TLS protocol for authentication and RDF (Resource Description Framework) for data model. Take, for example, the WebID https://bob.example.org/profile#me, for which the WebID Profile describes the name, a friend (via WebID), a picture.[3]

Using already common Web standards has many benefits. For example, as shown in the example above, WebIDs can also utilise existing Web vocabularies such as

[3] The example is taken from the WebID specification document, https://www.w3.org/2005/Incubator/webid/spec/identity/

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1234",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu/issuers/1a4",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "expirationDate": "2020-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:123456789abcdefghi",
    "degree": {
      "type": "PhD Degree",
      "name": "PhD in Computer Science"
    }
  },
  "proof": { // digital signatures snipped ...
  }
}
```

Listing 3: Verifiable credential example

FOAF[4] to build "a Web of trust" to link together WebID profiles (family members, group of friends, work colleagues, etc.). Such a web of trust can then be used by a service to make authorisation decisions.

3.3 Verifiable Credentials

A Verifiable Credential is a Web standard digital credential model currently being developed by W3C.[5] Its aim is to promote digitisation of credentials so that more services can support seamless and secure credential sharing. The data models for a verifiable credential are based on standard Web components such as URI and JSON. Each credential consists of some metadata, claims for the credential and proofs (usually a digital signature) that help securely authenticate and verify the credential content.

Verifiable credentials provide a standardised approach to representing and operating on credential data. Along with a decentralised identifier such as DID, and blockchain-based credential storage or verification, verifiable credentials could play an important role in building a secure, tamper-evident and privacy-preserving identity and credential management solution. In the example, the first ID refers to the identifier of the credential, and the second ID under CREDENTIALSUBJECT refers to the DID of this credential holder.

[4] FOAF is a set of description standards for linking people and information using the Web vocabularies, http://xmlns.com/foaf/spec/

[5] Verifiable Credentials, https://www.w3.org/TR/vc-data-model/

3.4 X.509 and PGP

Besides the open digital identifier schemes, we mention two digital identity standards that are widely in operation and accepted by Internet service providers and organisations today: X.509[6] certificate and PGP.[7]

X.509, first proposed in 1988, is a public key infrastructure standard for both identity verification and asymmetric key cryptography. An X.509 certificate is a digital certificate that accepted this standard to conduct verification of an entity, including data as public key, digital signature and other necessary identity information related to this entity. Applying this standard, a certificate authority distributes key pairs and certificates to each entity. The private key is kept securely by the entity itself, while public key is included in the certificate. The key pair of an entity can be utilised for two goals: (a) an entity can digitally sign the documents which then are verified using the public key; (b) a message can be encrypted by an entity's public key, and only the entity can read the message using its private key for decryption.

PGP (also known as "Pretty Good Privacy") is a protocol for email encryption/decryption, which was proposed in 1991 and now has become a de facto standard worldwide. This standard adopts the techniques of data compression, hashing, and public/private key mechanism to protect entities' privacy. Generally, a message is first compressed to save transmission time and storage and then encrypted by the sender's private key, which can be decrypted using the pre-shared corresponding public key.

Nevertheless, most of the existing identity management solutions rely on a centralised authority, including the above two standards, either for identity registration or credential verification. For instance, a server is required to act as the certificate authority for key pair distribution, which may cause single-point failure. Consequently, entities often do not have full control over their data which may be compromised without their awareness.

4 Blockchains in Self-Sovereign Identity Systems

4.1 The Role of Blockchains in SSI

Blockchain has been widely recognized as a viable technology for enabling DID in terms of data integrity and confidentiality. As self-sovereign identity needs to operate with no intermediaries, it aligns with the inherent peer-to-peer nature of blockchain. Therefore, blockchain is considered a base infrastructure for imple-

[6] https://www.itu.int/rec/T-REC-X.509

[7] https://tools.ietf.org/html/rfc4880

menting self-sovereign identity, providing decentralised identity registration and data storage. The codified public key infrastructure can be utilised as the certificate authority, while smart contract supports the implementation of the standard DID method specification.

In fact, as W3C has published the DID and verifiable credential specifications, SSI applications would prefer to follow these standard specifications, utilising blockchain as a shared registry of DIDs (i.e. containing the indexes of all DIDs). This registry function is crucial in the ecosystem of digital credentials. In the DID and verifiable credential standard specifications, such a registry is referred to as "Verifiable Data Registries" where registered DIDs are resolved to DID documents. In the specifications, the role of a verifiable data registry can be allocated to distributed ledgers, decentralised file systems, databases of any kind, peer-to-peer networks and other forms of trusted data storage.

4.2 Existing Blockchain-Based SSI Systems

Many organizations and companies have been developing self-sovereign identity platforms using blockchain. Some of the major systems include Sovrin [14], Serto [9] and Stacks (previously known as Blockstack) [2].

Sovrin[8] provides a global public, *permissioned* SSI platform for exchanging digital identity which is represented by the DID scheme. Sovrin leverages blockchain, IBM Hyperledger, as the third-party registry for resolving identities and securely establishing relationships between them. No personally identifiable information (PII) is stored on the public ledger.

Serto, formerly uport,[9] is another SSI platform that utilises Ethereum as an identity certification authority. An identity, represented by a DID, is associated with a smart contract that allows identity resolution, revocation or replacement. The personal information is stored on-device and off-chain with IPFS (InterPlanetary File System).

Stacks is considered an ecosystem of a decentralised computing network and decentralised applications (DApps), which builds and extends on the Bitcoin blockchain to run general purpose smart contracts. It uses a unique decentralised domain name scheme (Blockchain Name System) to manage identifiers (e.g. a user's public key). A user can securely and privately store the data on Stacks, and a decentralised app will utilise the data with the user's permission.

In terms of managing the keys, each system employs different scheme. For instance, registering a DID in Sovrin does not require an on-chain address as the underlying platform, Hyperledger Indy, operates specifically for identity management domain. It creates a new DID via a "NYM" transaction (a type of

[8] Sovrin, https://sovrin.org/

[9] Serto, http://www.serto.id

Hyberledger Indy transaction that is designated to creating a DID and managing keys) along with a key pair for signatures and verification. These three values are then stored in the user's local application (e.g. a digital wallet). Serto may operate in several Ethereum networks as long as ERC1056 is deployed. ECR1056 (Etherium Lightweight Identity) specifies how on-chain identities are managed on Ethereum. By default, creating a Serto DID needs an Ethereum address as the underlying key. This simplifies the storage issue of the keys in that only the address is kept in a wallet, while the DID and corresponding DID documents are recorded in the ERC1056 smart contract. In Stacks, both on-chain and off-chain DID management schemes are enabled. In the on-chain scheme, a registration of a new DID sends two transactions to Bitcoin, while an off-chain registration requires an entity to have an existing on-chain DID and submits a request to an off-chain registrar. DID owners take the responsibility of uploading corresponding DID documents to relevant locations for subsequent workflows.

Most SSI applications in use or development today can be found in the DID specification registry,[10] as most of them make an effort to incorporate the available W3C standards on decentralised identity such as DID.

4.3 Non-Blockchain-Based Solutions

Currently, most SSI solutions employ blockchains as the means to achieve a decentralised trusted authority to create, manage and verify identities. Although blockchain has many features (e.g. transparency, persistent/tamper-proof data storage) that are inherently appealing to realising SSI core principles, there are alternative options to implement the idea of self-sovereignty over identities and associated data.

For instance, a new paradigm, collectedly known as Personal Data Store (PDS) systems [5], aims to decentralise data and its processing and give the users total self-governance over their own data. A leading example of this paradigm is Social Linked Data (SOLiD)[11] within which a user is represented a WebID, a W3C standard for identities, and maintains a POD (Personal Online Data store) to store their own data and can fully control who can access to the data within their PODs.

IRMA [11], which stands for "I Reveal My Attributes", does not use blockchain but implements identity and credential management concepts similar to SSI. To avoid revealing too many uniquely identifiable credential attributes such as passport numbers or email addresses over online interactions, IRMA uses attribute-based authentication. Credentials are securely stored on a user's mobile phone via a personal secret cryptographic key. The IRMA app on the user's device allows the user to sign and share credentials. Only those attributes which are relevant for

[10] https://w3c.github.io/did-spec-registries/#did-methods

[11] SOLiD, http://solid.mit.edu

a transaction are disclosed, and it provides the user transparency about who is requesting to access which attributes.

5 SSI Lifecycle and Design Patterns

Still in the early stages of maturity, SSI implementation approaches using blockchain lack systematic software architecture design proposals that could guide software architect and developers through important software quality concerns such as scalability, maintainability, usability, etc.

As a step towards proposing a systematic architectural view of blockchain-based SSI systems, this section presents an analysis of the lifecycles of three major SSI objects (i.e. key, identifier and credential). We annotate the state transitions with software design patterns that we consider critical to SSI application development. This analysis becomes the basis for a reference SSI platform architecture that supports the idea of Design-Pattern-as-a-Service (DPaaS) in Sect. 6.

5.1 SSI Objects and Their Lifecycles

Figure 5 illustrates the lifecycles of the main SSI objects and their interconnections: *key*, *identifier* (represented by DID) and *credential*. The grey boxes indicate the SSI design patterns relevant for the states and transitions concerning the SSI objects. The interconnections (dotted line with empty arrow heads) show how certain states in each lifecycle affect the objects (e.g. a key is created, with which an identifier is registered).

- **The lifecycle of a key** starts when a key-pair is created for an entity. Once created, it is stored in a location the entity prefers. If the private key is lost or compromised, it might be recovered later or replaced with a new private key. The key pair could also be deleted if the entity does not need it anymore.
- **The lifecycle of an identifier** starts when an identifier, a DID in this instance, is registered on blockchain using a key. Each DID is mapped to a blockchain account's public key to ensure uniqueness and linked to a DDO (DID Document) which specifies the public keys, authentication protocols and service endpoints. Entities resolve (i.e. verify) each other's identity using the public key kept in DDO. Once a DID is registered, its details contained in the associated DDO can be updated. A DID could also be revoked.
- **The lifecycle of a credential** starts when a holder receives a credential request from a verifier. The holder then sends credential requirements to an issuer. The issuer generates the required credential and signs it. The holder receives the credential and discloses it to the verifier. The verifier checks the signature of

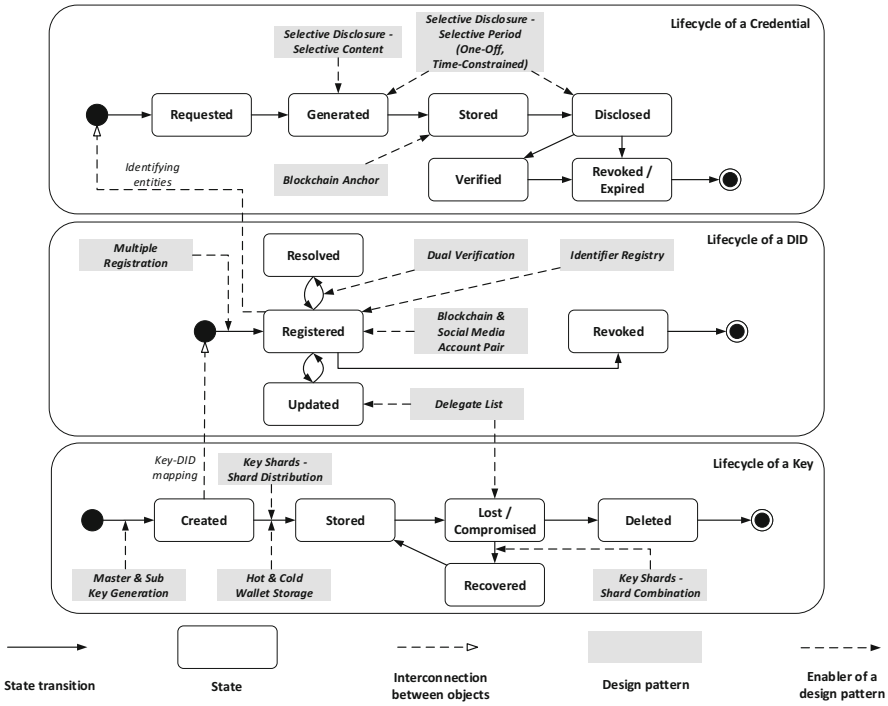


Fig. 5 Lifecycles of three main objects in self-sovereign identity

the issuer to verify the authenticity of the credential. The credential could expire or be revoked by an issuer.

5.2 *SSI Design Patterns*

We now present design patterns associated with the SSI objects. Figure 5 indicates the state or state transition with which each pattern is associated. We list three representative patterns for each object in a summary format. [7] reports a full list of the design patterns with known usage examples.

5.2.1 Key Object Related Patterns

Master and Sub-key Generation

Problem and Forces Public key cryptography and digital signatures are used to identify accounts and authorise transactions submitted to a blockchain. Using a

single key for all transactions has serious privacy implication for an identity owner since transactions can be correlated to expose all the identities an entity holds.

Solution Each entity can have a master-key to manage sub-keys which are used for signing messages under different identities. For example, a person can have a sub-key for the student identity and another sub-key for the company intern identity. Each sub-key is linked to a unique identifier and stored as part of the identifier's data in the identifier registry, which can be updated using the master-key. The use of master-key must be minimised (i.e. only used for controlling sub-keys) due to its importance. Figure 6a is a graphical representation of the pattern.

Consequences The solution preserves identifiability as each transaction is signed by a private key, and privacy is improved as each sub-key maintains its own identity and transactions are less likely to be correlated to the user. Also, if a sub-key is lost, the master-key can be used to revoke the lost key and update it with a new sub-key, which increases the availability of the system. However, if the master-key is lost, the identity owner loses control of all the sub-keys which means loss of control of all the owned identities.

Hot and Cold Wallet Storage

Problem and Forces As a blockchain network participant, one entity can rely on so-called wallets to manage its accounts and interact with blockchain. An entity's wallet may suffer malicious attacks, leading to key theft. The attacker can send transactions under that entity's name to blockchain using a compromised key.

Solution Users can choose to store keys in two types of wallets, namely, hot wallet and cold wallet. Hot wallet refers to the blockchain gateways that are connected to the Internet. Through a hot wallet, a user is able to directly conduct specific operations (e.g. generation) to its accounts and related decentralised identifiers stored on-chain. Cold wallet refers to key storage that is off-line, keeping the accounts from being hacked. A cold wallet can be any device disconnected from

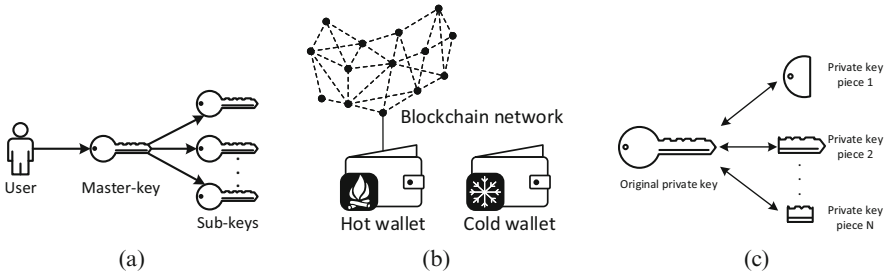


Fig. 6 Some of the *key* related design patterns. (**a**) Master & sub keys. (**b**) Hot & cold wallet storage. (**c**) Key shards

the Internet or even a paper recording an entity's keys. When the keys stored in a cold wallet are required for signing transactions, the user needs to connect the cold wallet device to a computer and copy-paste the key in the relevant field. A user can combine these two wallets: storing accounts that are frequently used in a hot wallet while using a cold wallet to keep those that are not used often. Figure 6b is a graphical representation of the pattern.

Consequences Cold wallets are separated from the Internet, which provide secure storage for entities' keys, but they are less convenient to use. Hot wallets store one's secret keys online, which is still vulnerable to theft.

Key Shards

Problem and Forces In self-sovereign identity, a participant may have multiple keys, for instance, signing key for transaction authorisation, public/private key pair for encryption/decryption, etc. Consequently, key management is significant to the users, especially credential issuers and holders. A user may lose or forget his/her secret keys under some circumstances, e.g. the device containing the keys is lost or broken. Losing the keys denotes that the owner could lose control over its blockchain accounts in self-sovereign identity and the related identities.

Solution To protect the security of a secret key, one can spilt that key into several pieces as its requirement and define a regrouping threshold. The key pieces can be kept in any way the user prefers, e.g. wrote on a paper and locked in a safe box, given to family and friends, etc. When a key is lost, the user needs to regain enough key pieces (more than the preset regrouping threshold), and these pieces can help rebuild the complete key. Figure 6c is a graphical representation of the pattern.

Consequences A lost key can be recovered by its shards, which provides great convenience to blockchain users. Also, the shards are stored in a decentralised way, which reduces the risk of losing all shards in an attack. However, maintaining key shards needs extra vigour in that if a key shard is lost, there is no way to restore it and having multiple key shards provides multiple targets to attack.

5.2.2 Identifier Object Related Patterns

Identifier Registry

Problem and Forces Identity is defined as sets of attributes related to an entity.[12] In software applications, identity attribute data needs to be accessed for a specific purpose. An identifier is a globally unique persistent series of digits and/or char-

[12] https://www.iso.org/standard/77582.html

acters that is used to uniquely identify an entity (e.g. human, organisation, device) within one domain and can be used to retrieve the storage location of the identity attribute data. A Decentralised Identifier (DID) is a new type of identifier which is designed for cryptographically verifiable self-sovereign identity. In traditional centralised software systems, mappings between an identifier and the identity data storage location are maintained by a centralised single authority which may become a potential single point of failure.

Solution Implementing an identifier registry designed as a smart contract to maintain bindings between an identifier and the location of associated off-chain identity data attributes. This identifier registry smart contract is the main entry point for accessing the attributes of an identity, which can map each identifier to a storage (e.g. IPFS, Dropbox, etc.) location for the respective identity attributes (e.g. an IPFS hash linking to the IPFS storage location containing the user's identity attributes). Only the identifier owner is allowed to update the storage location of identity attributes. Each identifier points to an identifier document which describes how to use that specific identifier, e.g. public keys used for digital signatures and service endpoints for interaction. Figure 7a is a graphical representation of the pattern.

Consequences Through the cryptographic access controls built into the blockchain, it is guaranteed that only the owner of the identity (i.e. the key owner) has the right to modify the data reference in identifier registry. Also, the data structure of identifier registry only stores identifiers and locations of identity attributes, which makes the solution scalable. When a public blockchain is used, the cost can be kept low as the data size of identifiers and identity storage locations is fixed. However, the off-chain identity data store might not be as secure as blockchain. The raw data may be changed or deleted without authorisation.

Multiple Registration

Problem and Forces An identifier is used to uniquely identify an entity and to retrieve the identity attribute data. Sending all transactions using a single identifier

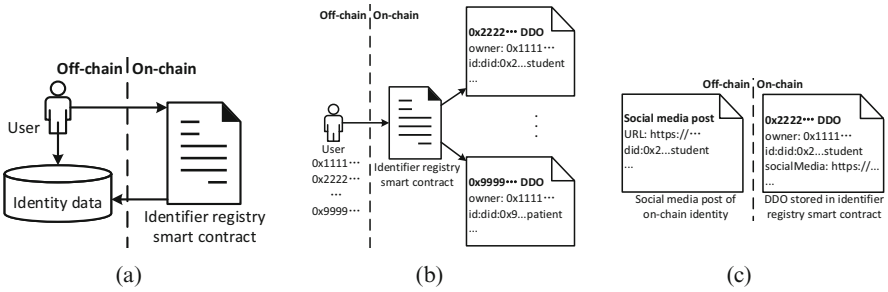


Fig. 7 Some of the *identifier* related design patterns. (**a**) Identifier registry. (**b**) Multiple registration. (**c**) Blockchain & social media account pair

has serious privacy implication for an entity since these transactions can be correlated to expose all the identities this entity holds.

Solution Each entity can establish a unique identifier for every relationship (i.e. every identity) they have, which allows keeping interactions with one entity entirely separate from any other entity. For example, the relationship a person builds with a hospital is completely separate to the one that is established with a university. Neither the hospital nor the university could proactively use the identifiers to correlate this person's activities. Figure 7b is a graphical representation of the pattern.

Consequences This pattern improves privacy; an entity's activities in different identity relationships can hardly be correlated. Also, the loss of signing key of one identifier does not affect the other identifiers. However, compared to a single identifier, multiple identifiers require more effort to manage.

Blockchain and Social Media Account Pair

Problem and Forces Social media profiles can be considered as one of the most important assets, which are critical to achieve more exposure on the Internet, attract more attention or improve online reputation. The trustworthiness of a social media profile can be improved by verifying the account using traditional identity issued by some central authority. On the other hand, blockchain provides a decentralised infrastructure for self-sovereign identity, where entities are in control over their own identities. To ensure the trustworthiness, some identity blockchains (e.g. Hyperledger Indy) are designed as public permissioned blockchains, which are governed by a group of participants. In addition to verification by some certain people or central authorities, a user can link his/her social media profile (e.g. Twitter) to his/her identity registered on blockchain to improve the trustworthiness of both social media profile and blockchain-based identity. The problem here is how to bind a social media profile with the corresponding blockchain-based identity to ensure mapping.

Solution An entity can create an attribute of social media in the identifier document. Signing the attribute with the blockchain signing key creates a claim that the blockchain-based identity controls the social media account. The attribute also contains a URL which links to a social media post stating that the social media account also controls this particular blockchain identity. Thus, a two-way link is established for connecting the blockchain identity with the social media profile. The two directional binding makes sure that that the social media profile and blockchain-based identity have a one-to-one mapping. Figure 7c is a graphical representation of the pattern.

Consequences Using this pattern, the trustworthiness of both social media account and blockchain-based identity can be improved by binding them together, and blockchain provides a secure data store. However, it should be noted that the

trustworthiness of social media account itself relies on the verification process, and the trustworthiness of blockchain-based identity depends on the trustworthiness of trusted participants.

5.2.3 Credential Object Related Patterns

Selective Content Generation

Problem and Forces A verifier requires certain information to prove a holder's identity; thus, a holder only needs to share a credential with necessary data to the verifier. If issuers publish general credentials to holders, a verifier can learn all identity data involved when only some particular attributes are needed. For instance, if a person shows his/her ID to identify the age, his/her address is presented either. This may cause data leak as extra information is provided.

Solution *Selective Content Generation* allows issuers to decide what identity attributes are contained in a credential. An issued credential needs to satisfy the target verifier's specific requirements of holder's identity, without revealing extra data. A credential with selective content disclosure can be generated via the following approaches.[13] Figure 8a is a graphical representation of the pattern.

- *Atomic credentials.* An issuer generates multiple credentials, and each one only contains exactly one identity attribute about the holder. Consequently, the holder can flexibly disclose those required credentials to a verifier.
- *Selective disclosure signatures.* A general credential is issued to a holder, but some special signature schemes (e.g. Camenisch-Lysyanskaya signatures) allow them to only reveal necessary information.
- *Hashed values.* A general credential consists of multiple identity attributes, but each one is hashed with different nonce. When verifying a credential, a verifier can only validate those with the actual values provided by the holder, but cannot determine other hashed values.
- *Zero-knowledge proof.* When proving certain identity attributes, a holder can protect its information by giving a range instead of precise value (e.g. age is over 18).

Consequences A credential with selective content can disclose the identity data which satisfy the verifier's individual requests while keeping other identity data private. However, creating and maintaining multiple credentials with different contents can incur extra costs.

[13] https://w3c.github.io/vc-imp-guide/

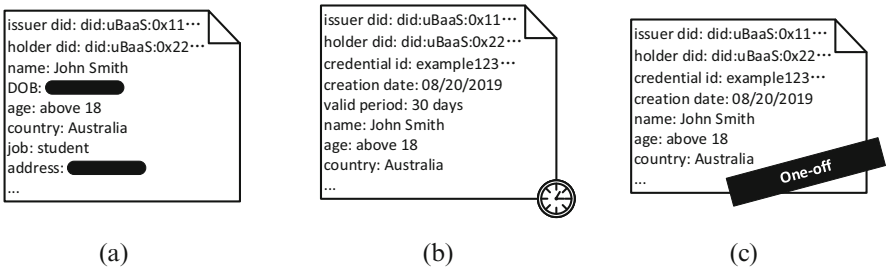


Fig. 8 Some of the *Credential* related design patterns. (**a**) Selective content. (**b**) Time-constrained access. (**c**) One-off access

Time-Constrained Access

Problem Usually an identification process lasts for a certain time period. After proving the identity of an entity, the presented credential has accomplished its mission and should not be accessed again. After receiving a credential, a verifier then has the ability to access, read and verify certain identity data of the holder. If the credential is long-term or even permanently effective, the verifier then can verify the credential after current identification process, which means that it can still access and check the holder's identity data when there is not a legitimate permission for proving the identity, resulting identity data leak.

Solution A holder is able to generate an identifiable link and define its accessible period (e.g. certain days). The link can redirect to a page presenting credential content. Afterwards, the holder can share the time-constrained link to verifiers instead of the original credential itself. Within the predefined accessible period, a verifier can visit and verify the credential for identification without limit. Nevertheless, when the link is expired, there is no approach for the verifier to obtain credential content again. Figure 8b is a graphical representation of the pattern.

Consequences A holder can determine the accessible period of a shared link, which ensures that the holder's identity information can only be fetched within a particular identification process. It also is also flexible as links with different accessible periods can be sent to different verifiers. However, storing the credential replicas requires more storage. A malicious verifier may take a photo of the credential when accessing it; then it has the credential content even if the shared link is expired. Although the compromised credential cannot reveal up-to-date information of the holder, the attacker still maintains historical identity attributes of the holder to some extent.

One-Off Access

Problem and Forces A verifier does not require a long-term effective credential but only needs to check the identity of a holder once for a specific purpose.

Sometimes an identification process does not require a strict verification procedure, but only needs to check the identity for once. For instance, travelling by train/airplane or going to a theme park only asks for checking credentials before entering. If a holder presents a long-term effective link redirecting to the credential content, a malicious verifier may access the holder's data illegally after identification process. This can be considered as an extreme version of *time-constrained access*.

Solution A holder is able to generate an identifiable link, which redirects to a one-off page presenting the credential content. One-off links can be shared with verifiers on some special occasions. After being visited once, the link becomes invalid that no one can use it to access the credential content. Figure 8c is a graphical representation of the pattern.

Consequences *One-Off Access* defines an extremely short period of access to the credential content, satisfying temporary identification requests. However, once issued, a malicious verifier can still take a photo of the credential when accessing it, which can cause a privacy issue for the holder.

6 A Service-Oriented Reference Architecture Proposal for SSI

The analysis of the SSI objects, their lifecycles and the design patterns in the context of blockchains allows us to conceive a systematic view of an SSI architecture that broadly covers the SSI requirements and software quality concerns.

In proposing a reference SSI architecture based on the analysis, we also take advantage of the service oriented architecture paradigm where modular and composable *software services* ensure extensible and portable system designs.

We name our architecture Design-Pattern-as-a-Service (DPaaS) for SSI. As the name suggests, the architecture aims to highlight the core SSI objects and the design patterns as functional service operations of the objects. Figure 9 illustrates the overall three-layer architecture: (1) service layer, (2) off-chain data layer, and (3) on-chain data layer. The on-chain data layer represents the data store maintained within the blockchain, most notably the DID registry for storing and accessing DIDs. The off-chain data layer refers to any other data stores in the system that are outside of the blockchain, notably databases maintained by the issuer or other registry data that are kept by the system operators.

The service layer is comprised of key services, DID services and credential services, each group of services representing functional operations as a set of APIs that interact with the relevant SSI object.

[8] lists the details of each group of services in full. Here, we briefly provide an overview of each service class.

Key Services: Key management related services are provided in wallet applications for the users. A user (i.e. an entity) is required to create at least one blockchain account which contains a key pair for sending/signing transactions

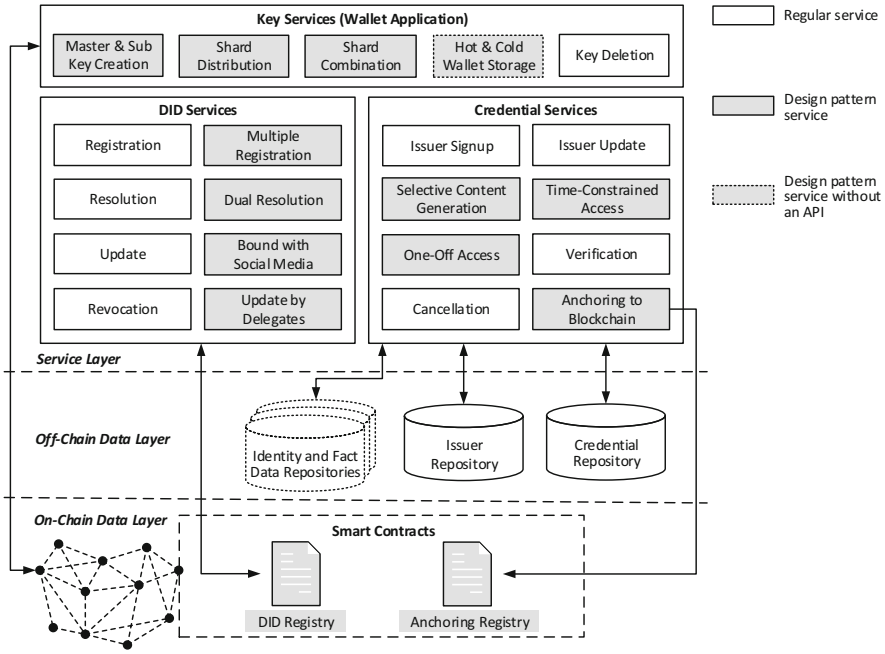


Fig. 9 Architecture of Design-Pattern-as-a-Service Platform for self-sovereign identity

and registering a DID. The services include the functional operations supported by the key related patterns such as *Master and Sub-key Generation* to split the management key (i.e. master-key) and signing keys (i.e. sub-keys) and *Shard Distribution* and *Shard Combination* which help the user manage the private key or recover a lost key. Once a key pair is created, users can choose to store the key in online wallet applications (i.e. hot wallets) or manually write down on a piece of paper (i.e. cold wallet) via *Hot and Cold Wallet Storage*.

DID Services: The DID management services provide implementations for all identifier management functionality offered by patterns such as *Multiple Registration* which allows each key pair to register a DID for every identity an entity has to avoid privacy disclosure caused by transactions correlation, *Resolution/Dual Resolution* which allows entities to verify each other's identity, or *Bound with Social Media* to combine social media identities with DID identities. To ensure identity data integrity and address blockchain's scalability issue, our design suggests creating a lightweight *Identifier Registry* as a smart contract on blockchain, which only maintains the mappings between registered DIDs and their associated descriptions (i.e. DDOs). The identity data and facts about a holder are stored in off-chain *Identity and Fact Data Repositories* hosted by the respective issuers (since we focus on legal identities).

Credential Services: Credentials are maintained via credential services. We assume that, to become a valid issuer, an entity needs to first register via *Issuer*

Signup, which requires a super account's (i.e. the system owner) approval. All the certified issuers' DIDs are maintained in *Issuer Repository*. However, the suggested design could be further adapted in a decentralised way, by having a group of super issuers register and maintain a new issuer.
The services mainly represent the credential content management patterns such as *Selective Content Generation*, fetching relevant data from the local *Identity and Fact Data Repository* and generating a signed credential and *Time-Constrained Access* or *One-Off Access* which can be implemented with a security token (e.g. Jason Web Tokens (JWT)[14]). An issuer can revoke an existing credential via *Cancellation* if the holder no longer meets the prerequisites.

7 Conclusion

Digital identity is central to a successful digital economy. However, most of the technical solutions to identity and related credential management have relied on identity data and its processing to be centralised for the identity and service providers.

Decentralised identities are digital identities and associated credential data that are created and managed by an individual user, and the user maintains the control of the identity. This chapter introduced self-sovereign identity (SSI) as an emerging digital identity management scheme that aims to systematise and standardise the core principles of decentralised identities.

Blockchain is considered a promising solution that could provide the system properties required by SSI to realise its core principles such as decentralised identity authority, persistent storage or high availability. There have been increasing efforts in both industry and academia to address many of the identity management challenges using blockchain technology in the past few years [6, 10].

Despite the efforts so far, there are still many open challenges in advancing SSI. For example, as the central idea of the identity management scheme is to have no central authority, there is an issue of effective decentralised governance structure that would be appropriate for SSI (e.g. revoking an identity or introducing a new credential issuing authority). The cost of running the underlying blockchain systems could be a potential barrier for wider adoption of blockchain-based SSI solutions. Although it is possible to devise a private or permissioned scheme to eliminate the monetary cost, a private or permissioned scheme may lead to a vendor lock-in solution that is not ideal for truly decentralised identity platforms. Another issue is its usability. Many SSI solutions are in early stages, and much of the focus has been building the infrastructure and internal protocols, and less has been done on how to integrate the users into the system with good usability. Relating to this

[14] JWT is an open, industry standard method for representing claims securely between two parties. http://jwt.io

issue, SSI still lacks a software architectural view that systematically addresses common software quality concerns such as scalability, maintainability, usability, etc. This chapter introduced an analysis of SSI main objects, their lifecycles and design patterns drawn from their interactions. The design patterns highlight, among other things, some of the key usage patterns of private/public key-pair management and scalable and flexible identifier/credential management. Based on the design patterns, a reference architecture is proposed. The architecture promotes a reusable and extensible service-based design of system functionality which is modularised by the patterns.

Given that SSI requires a fundamental paradigm shift in how users' identity and their data are managed in a software system, one of the practical challenges in the future development of SSI is to be able to gather critical support from practitioners through continuing development of the standards and their applications. In terms of future research directions in the field, establishing effective governance models for SSI would be essential to promote broader adoption of SSI-based applications. In particular, issues such as defining and regulating accountability for SSI roles or a sustainable incentive model for maintaining "good" SSI governance would be important in realising the potential of SSI in real-world applications. A solution towards good governance should consider how to integrate relevant existing regulations into SSI operational workflows (e.g. Australia's regulation and standards on digital identity in blockchain[15]). Another important area of research in SSI is trust management. In the era of AI and deep learning-driven applications, trustworthiness of a modern software system will depend on its ability to instil innate human values such as trust into the identity of all interacting entities in the system. The topics of trust in SSI could look into how SSI may contribute towards improving trust in future software systems, as well as investigate operational models to enhance trust in SSI platforms from within.

References

1. F. Alaca, P.C.V. Oorschot, Comparative analysis and framework evaluating web single sign-on systems. ACM Comput. Surv. **53**(5) (2020). https://doi.org/10.1145/3409452
2. M. Ali, J. Nelson, R. Shea, M.J. Freedman, Blockstack: a global naming and storage system secured by blockchains, in *2016 USENIX Annual Technical Conference* (2016), pp. 181–194
3. C. Allen, The path to self-sovereign identity. http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html. Accessed 15 Feb 2020
4. Council of the European Union: General data protection regulation GDPR (2019). https://gdpr-info.eu/
5. H. Janssen, J. Cobbe, C. Norval, J. Singh, Decentralised data processing: personal data stores and the GDPR. Social Science Research Network (SSRN) (April, 2020). https://doi.org/10.2139/ssrn.3570895

[15] https://www.industry.gov.au/data-and-publications/national-blockchain-roadmap/regulation-and-standards

6. S.Y. Lim, P.T. Fotsing, A. Almasri, O. Musa, M.L.M. Kiah, T.F. Ang, R. Ismail, Blockchain technology the identity management and authentication service disruptor: a survey. Int. J. Adv. Sci. Eng. Inf. Technol. **8**(4–2), 1735–1745 (2018)
7. Y. Liu, Q. Lu, H.Y. Paik, X. Xu, Design patterns for blockchain-based self-sovereign identity, in *Proceedings of the European Conference on Pattern Languages of Programs 2020*, EuroPLoP'20 (Association for Computing Machinery, New York, NY, 2020). https://doi.org/10.1145/3424771.3424802
8. Y. Liu, Q. Lu, H.Y. Paik, X. Xu, S. Chen, L. Zhu, Design pattern as a service for blockchain-based self-sovereign identity. IEEE Softw. **37**(5), 30–36 (2020). https://doi.org/10.1109/MS.2020.2992783
9. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, M. Sena, uPort: a platform for self-sovereign identity (2017) https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221. pdf
10. A. Mühle, A. Grüner, T. Gayvoronskaya, C. Meinel, A survey on essential components of a self-sovereign identity. Comput. Sci. Rev. **30**, 80–86 (2018)
11. Privacy By Design Foundation: IRMA. https://irma.app/docs/what-is-irma/. Accessed 10 Jan 2021
12. D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, Decentralized identifiers (data model and syntaxes for decentralized identifiers). https://w3c-ccg.github.io/did-spec/. Accessed 15 Feb 2020
13. M. Sporny, D. Longley, D. Chadwick, Verifiable credentials data model (expressing verifiable information on the web). https://www.w3.org/TR/verifiable-claims-data-model/. Accessed 15 Feb 2020
14. P. Windley, D. Reed, Sovrin: a protocol and token for self-sovereign identity and decentralized trust. https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf. Accessed 15 Feb 2020

From Centralized to Decentralized Remote Electronic Voting

Christian Killer, Bruno Rodrigues, Eder John Scheid, Muriel Figueredo Franco, and Burkhard Stiller

1 Introduction

Electronic Voting (EV) is straightforward in an ideal and secure world: voters send their votes to a trusted counter, which tallies the votes and publishes the final result. However, that is far from being realistic when the opposite is the case: entities involved in EV are typically not fully transparent and trusted, and the context is adversarial. In a traditional EV setting, every participant in an election system is potentially malicious, for example, voters selling votes, dishonest election officials, and corrupt equipment vendors [42].

Therefore, opportunities exist to decentralize electoral process stages and promote transparency to reduce dependency on centralized entities. While decentralization reduces the control of entities that may be directly interested in the election result, the challenge lies in guaranteeing verifiability and privacy of all processes. This challenge directly addresses trust assumptions that a centralized system explicitly places on a single entity for (i) the execution of the voting protocol, (ii) the verifiability and privacy properties, and (iii) the audit trail's storage. Moreover, voting systems' essential privacy requirements oppose the need for convincing evidence that the election's outcome is correct while assuring ballot secrecy at the same time [6].

Voting research is interdisciplinary and spans decades and many fields, ranging from technical concepts, such as network security and cryptography, to socioeconomic aspects (e.g., statistics and law). Over the years, advances in research allowed for the deployment of EV systems, previously only formulated in theory.

C. Killer · B. Rodrigues · E. J. Scheid · M. F. Franco · B. Stiller (✉)
Department of Informatics IfI, Communication Systems Group CSG, Universität Zürich UZH, Zürich, Switzerland
e-mail: killer@ifi.uzh.ch; rodrigues@ifi.uzh.ch; scheid@ifi.uzh.ch; franco@ifi.uzh.ch; stiller@ifi.uzh.ch

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_16

For example, [54] categorizes existing EV approaches into (i) paper-based processes with electronic counting and (ii) digital systems that use direct recording systems (i.e., an electronic ballot box) and remote voting either in a dedicated network or remotely over the Internet. Advances in these different approaches enabled the practical evaluation of cryptographic schemes in real-world scenarios. Poll site systems (i.e., systems where voters are required to vote in a predetermined voting location with an electronic ballot) were extended with EV to provide a verifiable audit trail based on cryptographic voting schemes [2, 8, 57].

As a special case in EV approaches, Remote EV (REV) systems were designed and implemented to allow for vote casting in a remote and uncontrolled environment (e.g., from a personal device, typically today over the Internet). For that reason, REV is often referred to as "Internet Voting" (or I-Voting). Depending on the specific legal and country-specific context, REV and I-Voting may be considered synonymous. So far, various electoral processes used REV systems in the last few years, e.g., Switzerland [76], Estonia [43, 83], Norway [34], and Australia [41]. However, these systems are in major revision processes today due to security concerns raised, resulting in REV attempts being put on hold. Therefore, a secure implementation and deployment of a scalable and operational REV system still requires utmost scrutiny of every aspect of the system itself, its stakeholders involved, and its deployment and ultimately requires trust from the electorate.

All REV systems require Public Bulletin Boards (PBB), which are functionally comparable to final tally boards of traditional voting approaches and immutably store ballots to provide a verifiable audit trail to the public. In state-of-the-art REV implementations, PBBs (which do determine a central component for any REV system) are formulated as "trusted centralized components" and are assumed to provide the verifiable audit trail [1, 14, 19]. As further prior work outlined, the use of permissioned Blockchains (BC) or permissioned Distributed Ledgers (DL) provides an immutable, publicly shared, and append-only PBB [49, 51]. To reach these requirements, N trusted authorities determine a permissioned DL-network, formed by a Proof-of-Authority (PoA) consensus, which measurably lowers trust assumptions from BFT (Byzantine Fault Tolerance)-based networks from $\frac{2}{3}$ to at least $\frac{N}{2} + 1$ honest nodes. The BC's and DL's underlying data structures in place today provide favorable properties for audit trails, and hence, for PBBs. Thus, because of BC's and DL's immutable, append-only, and publicly readable structure, they were used to implement voting systems in related [15, 59] and prior work [48–50].

To establish an objective basis for understanding such distributed approaches, it is fundamental to differentiate between *permissionless* and *permissioned* decentralized networks. By definition, permissionless consensus algorithms, e.g., Proof-of-Work (PoW) or Proof-of-Stake (PoS), are based on an economic incentivization of the honest behavior of the majority of participants while punishing malicious behavior, which can always be detected. The major goal of permissionless networks is to remove the need for Trusted Third Parties (TTP) (e.g., as known for financial transactions without intermediaries), thus, any intermediary. Through logical

deduction, permissionless consensus algorithms are determined to be unsuitable for those use cases that already assume a TTP inherently. For instance, voting and election systems for governments and organizations are, by definition, led by trusted authorities (at least this can be safely assumed for a number of societies across the globe). Therefore, permissioned networks offer a highly suitable framework for use cases, where a set of N trusted authorities collaborate in their roles as defined by law or identity as determined by society without any economic incentive. Unfortunately, so far, the full set of implications (e.g., long-term privacy) are not yet considered when selecting a permissionless BC.

REV Core Challenges With the steady decline of trust in governments and elections [22, 80], it is imperative to address public service provisioning in a decentralized way [71]. In political sciences, efforts toward Electronic Government (E-Government) reverse such declines in trust. Still, whether this has a measurable effect on trust or whether it affects the government's perceived efficiency is disputed [80]. From a technological perspective, security-sensitive software is often not open-sourced [84], because of various interests colliding (e.g., commercial and national security interests). Although it is not guaranteed that open-source software is "automatically" more secure [75], voting systems require certainly and without restrictions always transparency and public scrutiny in order to build trust among the general public.

When it comes to fostering trust in electoral processes, the lack of transparency is very often the major concern. Without transparency, false accusations cannot be debunked, which undermines the trust in the electoral process. This problem can be tackled by publishing all verifiable data on a PBB, which is immutable and publicly readable. BCs and DLs highlight a feasible possibility to minimize or distribute trust among stakeholders. In contrast, in traditional REV systems, voters (i.e., the electorate) and Voting Authorities (VA) are the primary stakeholders. The public trust in the voting process (and its outcome) is the foundation of democracy, and it is based on verifiable evidence, which voters can publicly query. While VAs manage voting procedures, voters expect verifiable proof to conduct audits of these procedures.

Although there are many open challenges and threats in REV systems that will not be solved or mitigated with the usage of decentralized systems, there are two core challenges that REV systems can address in full and proven reliably: (i) achieving a tamper-proof and immutable audit trail in a practical PBB deployment and (ii) achieving a practical decentralization of trust by deploying a suitable consensus algorithm (e.g., Proof-of-Authority, PoA). While many research papers and industry efforts proposed BC- and DL-based REV systems, only very few of them thoroughly considered the detailed trust assumptions required and trade-offs to be made explicit, which need to be evaluated upon deploying a decentralized system in a highly adversarial context [61].

Contributions While both EV and REV systems expose trade-offs between election verifiability and privacy, REV systems add even higher complexity, since they

do not run within a controlled environment, i.e., outside of a prepared voting booth, where eligible voters cast their votes. In this regard, REV's fundamental goal is to achieve privacy while providing a fully verifiable audit trail. However, many of the BC- and DL-based systems proposed did not consider such significant requirements for REV systems, e.g., the trust model and assumptions of the underlying consensus mechanisms. Thus, transparency is a key factor for trust in REV systems. *For example*, the Swiss ordinance regarding REV requires any potential REV system being introduced to publish the source code for the public without exceptions [25].

Therefore, this chapter's major contribution is to address Remote Electronic Voting (REV) in exactly that context and to discuss trade-off settings for clarifying how Distributed Ledgers (DL) can enhance the decentralization of the electoral process by ensuring transparency and the ability to verify results while guaranteeing voter confidentiality at the same time.

Organization This chapter is organized as follows. Section 2 provides major background concepts for REV, such as Public Bulletin Boards (PBB) and protocol requirements, including their associated privacy properties. In a similar direction, Sect. 3 describes cryptographic primitives and underlying assumptions that are essential for REV and PBBs. Section 4 overviews the REV state of the art by discussing how cryptographic elements and mechanisms are leveraged to fulfill these previously outlined protocol requirements. Additionally, Sect. 5 outlines research directions and open aspects for REV systems considering as core principle a privacy-preserving requirement as the principal design driver for both centralized and decentralized REV implementations. Finally, Sect. 6 provides overall considerations based on those significant findings outlined and addressed herein.

2 Remote Electronic Voting (REV) Fundamentals

The core challenge in REV is to balance privacy and verifiability properties based on a transparent and evidence-based voting process. Many early REV systems used trusted central servers as a PBB, which represents a single point of failure operated by trusted stakeholders that may have an interest in the election's outcome [38]. For example, the Swiss Post (SP) Voting system operates in trusted, closed, and hardened datacenters under the control of the SP (a partially state-owned company). Although the protocol assumes only a set of separate control components as trustworthy, they are all technically operated by the SP [39]. The overall system is also dependent on a code-verification system, which assumes a trustworthy printing office that secretly prints and sends code sheets via postal mail to eligible voters. In the case of collusion between the printing office and compromised devices, the device can return correct verification codes, although votes are not cast as intended by the voter [6].

Ideally, a cryptographic voting scheme would allow voters to only trust cryptographic schemes. It would not be required to assume external stakeholders'

trustworthiness (e.g., a print office's printing verification code). In order to minimize these trust assumptions placed in centralized infrastructure, Blockchains (BC) and Distributed Ledgers (DL) are a suitable possibility to (i) minimize and distribute trust among stakeholders and (ii) enable practical implementation of an immutable and append-only PBB for REV systems. However, many of the BC- and DL-based systems proposed did not consider the complete set of requirements for REV systems, e.g., a trust model and assumptions of underlying consensus mechanisms. Persisting election evidence on a BC or DL enhances trust assumptions marginally, mainly because the election evidence's immutability is assured. Further, the practical decentralization of the voting process on an architectural layer can be achieved as well.

> We need to realize popular services in a secure, distributed, and decentralized way, powered by free software and free/open hardware. We need to build systems beyond the reach of super-sized companies and spy agencies. Such services must be based on strong cryptography. (Phillip Rogaway [71])

The design goals are similar and reasonable: increase transparency and provide a publicly readable and fully decentralized audit trail. After all, transparency is a critical factor for trust in REV systems. For instance, the Swiss Federal Ordinance for Federal Chancellery Ordinance on Electronic Voting (VEleS) [25] demands the publication of source code for any REV system that is used to tally more than 30% of the cantonal electorate. In order to assess the feasibility of any system, the architecture and its components need to be assessed. Therefore, the major building blocks of an REV system are key, with the core component of a decentralized REV system being built by the Public Bulletin Board (PBB). Voters use a Voting Client that executes a Voting Software Client (VSC) to interact with the PBB directly through a set of defined communication channels while interacting according to the voting protocol to ensure privacy and verifiability.

2.1 Building Blocks of an REV Architecture

Before detailing the key privacy and verifiability properties of an REV system, Fig. 1 depicts the main building blocks needed to design an REV system. In essence, the *Voter* uses a *Voting Client* to execute *Voting Software*, which was distributed by the *Voting Authorities (VA)*. A voter uses personal hardware capable of executing the voting software, e.g., a smartphone or personal computer. The voter fetches the official voting software from publicly available storage (e.g., a Web site, repository, or maybe a distributed storage). The integrity and authenticity of the voting software executable must be assured (i.e., cryptographically signed with the VA's credentials). Depending on the voting protocol, the *Ballot* is generated, which can include various cryptographic operations that need to be executed on the voting client. For instance, certain Zero-Knowledge Proofs (ZKP) may be required to prove ballot validity. A crucial component of REV systems is *Public Bulletin Boards (PBB)*, which are part

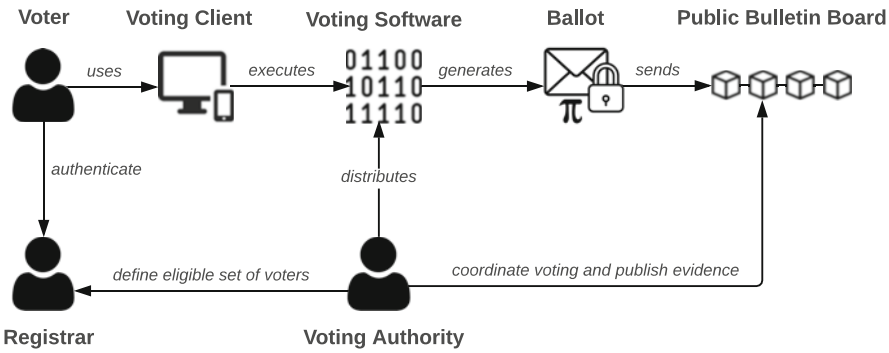


Fig. 1 Overview of major REV building blocks

of any REV system, used to publish an auditable evidence trail during the execution of the voting scheme [49]. According to [44, 45], a PBB shall provide the following properties:

1. It is an append-only data structure, i.e., information cannot be modified or altered.
2. It is public in the sense of being searchable by anyone.
3. It is consistent in its view for anyone accessing its information.

Voting systems require verifiable evidence of a correctly executed voting process. In that context, the PBB is the most relevant component of an REV system that a BC or DL can replace since it requires similar properties: immutability, be append-only, and persist a consistent view to anyone. Voters can query the PBB at any time to verify evidence published by the VA.

Another crucial aspect of REV systems is voter authentication i.e., assuring that only eligible voters can cast a ballot. Often, REV systems use a trusted *Registrar* entity to authenticate and check the eligibility of voters and provide them with the necessary credentials to vote. Hence, Identity Management (IdM) is crucial to give credentials to eligible voters. Usually, the IdM is handled by a TTP, such as the Voting Authorities (VA). The IdM manages voters' identities and provides credentials to the eligible set of voters, which can be used to authorize access to cast valid ballots. It is important that the IdM cannot fake identities because it would enable ballot-stuffing.

Another crucial aspect is the security of the communication channels used between the different entities. These channels need to be secured in order to guarantee various security properties, as well as the integrity of the full voting process. For instance, *sender-anonymous* channels and *untappable communication* channels are required [44]. *Sender-anonymous* channels ensure that a sender cannot be identified by the receiver of a message, whereas *untappable communication* channels ensure that neither the sender nor the receiver "can learn anything about the communication, including whether communication occurred or not" [44]. Depending on the specific voting protocol used, these assumptions need to be

carefully assessed with the practical deployment decisions, e.g., Over which channel does the voting software interact with any of the other components?

2.2 Architectural Models for Public Bulletin Boards

The use of a platform based on DLs for the construction of elections based on REV does not necessarily imply the use of BCs. It is fundamental to note the differences between the architectural models used in the different types of storage architectures (cf. Fig. 2). For instance, a permissionless BC-based architecture implies open participation in the consensus mechanism, i.e., every participant running a full node is able to participate in the consensus process. While traditional permissioned and centralized database storage may result in a lack of transparency in the electoral process, decentralized options may also have drawbacks, such as loss of efficiency and greater overhead in managing the electoral process stages.

Considering Fig. 2 and the distinction between centralized and distributed architectures for decentralized architectures, both are managed by only one organization and can count on one server (or a pool of servers in a datacenter) in a centralized architecture or several servers in a distributed architecture. In these two types of architectures, it is necessary to trust that the entity (i.e., federation) controlling the electoral process is not being influenced by adversaries. Thus, the transparency of the process is a fundamental element that an organization does not always guarantee but can be more easily obtained through N independent entities in a decentralized and transparent architecture.

Therefore, it is crucial to distinguish between BC and DL technologies through which it is feasible to decentralize electoral process where each participating entity is independent, (i.e., the decentralized process is federalized). In a decentralized REV, each entity is independent and can publish its results on the PBB without

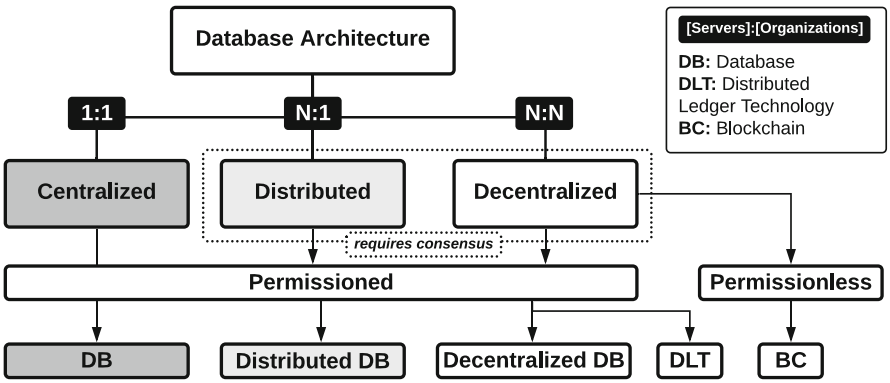


Fig. 2 Taxonomy of data structures and consensus algorithms

relying on another entity. This is a relatively more transparent process than a centralized REV.

2.3 Protocol Requirements

Literature on EV offers various protocols to meet the core requirements of privacy and verifiability, which are in direct opposition to one another and inherently contradictory, e.g., Individual Verifiability (IV) seems to be incompatible with Receipt-Freeness (RF) [36]. Also, [18] has shown that a voting system cannot achieve Unconditional Privacy (UP) and Universal Verifiability (UV) at the same time. They also showed that UV and RF could not coexist if the transcript of the vote depends on *only* public values and values known to the voter. For this reason, REV systems aim to achieve a balance between privacy and verifiability.

2.3.1 Privacy Properties

Vote Privacy is considered a human right and is stated as such in Article 21 of the Universal Declaration of Human Rights [82]. A voting system without Ballot Privacy allows large-scale vote-buying and voter coercion [73]. Thus, privacy is key in democratic voting. In this context, privacy can be further classified in subcategories, which are based on the degrees of privacy and assumptions made, ranging from the weakest notion of Ballot Privacy (BP) to the strongest of Coercion-Resistance [24] (Fig. 3).

Many privacy properties require at least a minimal amount of trust placed in the Voting Authorities (VA) to store the private keys for the election keys safely. Trust in single authorities can be distributed by using threshold cryptography, hence distributing the trust among VAs and other trusted stakeholders. For instance, by using a BC- or DL-based system, it is possible to decentralize the point of trust in a single party through transparent infrastructure in which all participants can

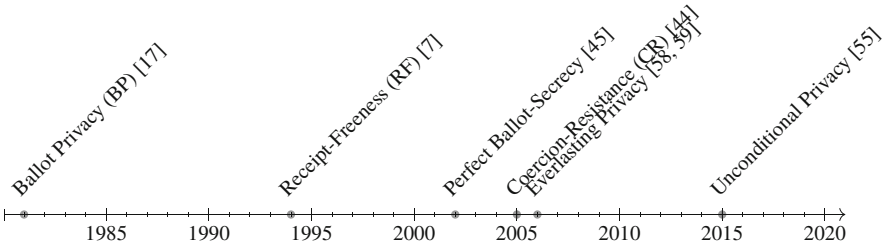


Fig. 3 Timeline of privacy notions

verify the distributed key generation, the confirmation of votes cast, and the vote tallying procedures.

> The will of the people shall be the basis of the authority of government; this will shall be expressed in periodic and genuine elections which shall be by universal and equal suffrage and shall be held by secret vote or by equivalent free voting procedures. (Article 21, (3) of Universal Human Rights Declaration [82])

In 1981, the first notion of *Ballot Privacy (BP)* was defined by [17] as "A voter's vote is not revealed to anyone." In [20], the notion was formally introduced, laying the groundwork for formal notions of voting system properties. Often, the term *Ballot Secrecy* is used interchangeably to describe BP. The National Institute of Standards and Technology (NIST) defined Ballot Secrecy in a more process-oriented fashion as follows: "a voting system is private if it (and the procedures/process for using it) does not make available additional information on an individual voter's ballot choice(s), beyond that contained in the tally" [58]. Later, in 2002, another notable definition followed from [47], called *Perfect Ballot Secrecy*, which "ensures that knowledge about the partial tally of the ballots of any set of voters is only computable by the coalition of all the remaining voters (this property captures strong voter privacy as understood in real world elections)." Further refinements of this definition can be found in literature based on the trust model or the computational power of an adversary [47, 56, 60].

Most early works on voting systems had underlying protocols that produced receipts. These receipts would allow voters to prove how they voted to others. Intuitively, the ability to produce a receipt of how a voter voted can easily be abused for vote-buying or coercion [7]. This brought about the definition of *Receipt-Freeness (RF)*, which is defined as "a voter cannot gain information which can be used to prove, to a vote-buyer, how she voted" [7].

The key difference between RF and *Coercion-Resistance (CR)* is whether the voter is considered malicious or not. In the case of RF, the voter is considered to be a dishonest voter who wishes to sell her vote. For CR, it does not matter whether the voter is honest or not. The voter is coerced by an adversary to disclose sensitive information, such as her vote or voting credentials. As such, CR is a stronger privacy guarantee than RF. Therefore, CR is defined as "a voter cannot interact with a coercer to gain information, which can be used to prove how she voted" [44].

The privacy definitions above do not explicitly define an adversarial model, i.e., do not include any assumptions regarding the adversary's computational capabilities. While current adversaries are bound by hardware and software available as of now, future adversaries may break underlying assumptions, which in turn break the security of cryptographic primitives and compromise the privacy properties. Many voting schemes are based on these hardness assumptions, such as the Decisional Diffie-Hellman assumption (DDH) [11], which is covered in detail in Sect. 3.3. With a focus on these intractability assumptions, further definitions of *everlasting* and *unconditional* BP notions are reviewed.

A voting system with *Everlasting BP (EP)* does not allow even a computationally unbounded coalition of voters or outside observers to determine for whom a voter

voted [60]. Going even further is *Unconditional BP (UP)*. A voting system with UP does not allow anyone, neither voters, outside observers, nor VAs, be they computationally unbounded or not, to determine for whom another voter voted [56].

Remembering the definition of perfect BP [47], even if VAs conspire, they are unable to link votes to voters and break BP. Therefore, a voting scheme with perfect BP does not require trust in the VAs or other parties. However, perfect BP does not mention any computational hardness assumptions and, hence, is not everlasting. Therefore, by combining everlasting BP (i.e., no computational hardness assumptions for privacy) and perfect BP (i.e., no trust in TTPs needed for BP), it is possible to achieve the Unconditional BP.

2.3.2 Verifiability

In real-world systems, voters have to place implicit trust in voting machines and polling staff for the integrity of the elections. In the pursuit of minimizing the requirement for trusted personnel and hardware, REV research proposed cryptographically verifiable voting schemes [3].

Once all votes have been cast, the system must be able to prove that the result was tallied correctly and that the single ballots contain the actual voters' choices. Prior work proposed many different notions of verifiability, and novel, formal definitions are being researched as well. Sako and Kilian [74] defined *Individual Verifiability (IV)* as "a voter can verify that her vote is included in the set of all votes cast."

However, IV is not sufficient to provide a verifiable REV system because the system then relies on each voter to verify that each ballot was correctly published and unaltered. Since that is an unrealistic assumption, [74] also defined Universal Verifiability (UV) as being achieved if "an observer can verify that the tally has been correctly computed from the set of all votes cast" [74].

Additional informal definitions of UV by Kremer et al. [53] are defined as being achieved if "anyone can verify that any vote in the set of "all" votes belongs to an eligible voter, and each eligible voter voted only once [53]." This notion already includes eligibility verifiability, highlighting the challenges of ensuring that only eligible voters vote and only vote once.

Further research in the realm of verifiability properties refined a more procedural perspective on the *end-to-end* voting process, called *End-to-End Verifiability (E2E-V)*, which is only fulfilled if a voter can verify the following three properties [44]:

- *Cast-as-Intended* (CaI): "her choice was correctly denoted on the ballot by the system."
- *Recorded-as-Cast* (RaC): "her ballot was received the way she cast it."
- *Counted-as-Recorded* (CaR): "her ballot counts as received."

The last property of E2E-V, CaR, is often also referred to as *Tallied-as-Recorded* (TaR) as well. Therefore, voters first confirm to the system that the vote has been correctly encrypted, and thus CaI is achieved (by verifying the encryption operation

on a second device, for instance). Then, the voter tracks her vote on the PBB by using her receipt and can verify that the vote was RaC. Lastly, the result's integrity is ensured by auditing the tallying process, which means that the REV system publishes cryptographic proofs that prove correct decryption and operation [3].

However, these verifiability properties rely on a PBB as an audit trail; otherwise, there is no data to verify. A PBB serves as an immutable, append-only audit trail. Therefore, verifiability properties *assume* such a PBB exists. In practice, a fully decentralized REV system requires a shared PBB. A public permissioned DL can serve as such and enable these different verifiability properties, allowing voters, auditors, and the general public to query the persisted data. It is crucial that the deployment of said BC or DL is carefully parameterized and configured to the use-case and voting protocol. For instance, there is a huge difference between simply persisting hashes on a public permissionless BC and having actual VAs perform Multi-Party Computation within a permissioned DL network. These differences make it necessary to look at the properties within the specific context and mode of deployment of the PBB.

Küsters and Müller [55] drove efforts forward to elevate the understanding of dependencies between various privacy and verifiability properties in their formal and informal definitions, as well as misconceptions and misunderstandings of these properties.

1. For instance, it is commonly believed that IV and UV are the same as E2E-V. However, IV and UV are neither sufficient nor necessary to achieve E2E-V.
2. Further, E2E-V alone is typically insufficient for practical purposes. REV systems should really be designed with accountability in mind.
3. Further, CR does not imply general privacy, but increasing the level of privacy can lead to a lower level of coercion-resistance.

These confusions and misconceptions highlight the importance of clear definitions and a clear assessment of these definitions. The privacy and verifiability notions are not fixed but are ever-evolving into different formalized approaches. An overview is provided in [55]. Also, privacy and verifiability are not the *only* protocol requirements. Two notable properties include the notions of Software Independence and Accountability in the context of REV systems.

Additionally, the requirement *Software Independence* can claim for a voting system *if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome* [70]. REVs are complex hardware/software systems. Minor errors or manipulations can lead to unpredictable errors, which adversaries might exploit. As such, Software Independence follows the approach of: *Verify the election results, not the voting system.*

Furthermore, *Accountability* is a more robust, extended form of verifiability, i.e., if the final tally does not correspond to how the voters actually voted, then VAs can be held accountable for the misbehavior. The misbehavior can be intentional or unintentional. Accountability is important for multiple practical reasons because it intensifies the incentive all parties have to follow their roles because they can be singled out in cases of misbehaviour [55].

2.4 General System Requirements

While these protocol requirements are essential for the formulation of voting protocols, the REV system also needs to provide multiple facets, many of which are not specific to REV systems but to information systems in general. These general security system requirements are less specific to REV, but still fundamental for REV in practice, as REV follows a clear software architecture. These general requirements may contradict protocol requirements [36]. For instance, software architecture characteristics usually refer to the *"-ilities"* that the system must provide: availability, reliability, testability, scalability, and deployability [68]. Also, besides these general requirements, convenience, flexibility, and usability are also desirable properties. Further requirements that address political, administrative, or even juridical questions are also crucial for REV in practice.

In the context of BC- or DL-based REV systems, the most crucial aspect is the definition of the security goals and the threat model because these build the basis for the system's risk assessment. For instance, when using a public permissionless BC as a basis, the system's architecture depends on the assumption that the public network is reliable, available, and scalable. Public permissionless networks are, by design, not controlled by known stakeholders. Therefore, it is hard to challenge or prove the properties and assumptions made. However, if a voting system relies on a known set of permissioned nodes, i.e., a public permissioned DL, then the assumptions, properties, and performance can be measured more precisely while still providing the benefits of transparency.

3 Cryptographic Primitives and Assumptions

This section focuses on the fundamental cryptographic primitives and their underlying assumptions. The following background is necessary to understand REV protocol approaches; how they work technically, i.e., how the underlying cryptographic mechanisms support the voting scheme itself; and based on which significant assumptions these primitives hold their security notions. Table 1 shows the most relevant cryptographic primitive schemes to understand REV systems and provides a brief description of each of them. Further, the most important cryptographic primitives are introduced, accompanied by the most essential *computationally* hard problems. Also, ElGamal message encodings are detailed in Sect. 3.1 because many REV protocols and systems are based on the ElGamal public-key cryptosystem [28]. This section outlines the most fundamental cryptographic primitives used in voting schemes and relates for an in-depth view of selected topics (e.g., hash functions) to the respective general security-related literature.

Chaum [16] first proposed Blind Signatures (BS) to enable untraceable electronic mail and payments, and BS find application in early REV schemes as well, which would work as follows: Suppose a voter v encloses her $ballot_v$ which is written on

Table 1 Overview of cryptographic primitives

Scheme	Description
Blind Signatures (BS)	Digital signature scheme to sign blinded messages
Homomorphic Encryption (HE)	Allows operations on ciphertext, manipulating the plaintext
Zero-Knowledge-Proofs (ZKP)	Prove knowledge of v interactively, without revealing v
Non-Interactive ZKP (NIZKP)	Prove knowledge of v non-interactively, without revealing v
Multi-party Computation MPC	Allows for removal of TTPs from distributed computation
Distributed Key Gen. (DKG)	Allows multiple parties to collaboratively generate a public key
Fiat-Shamir Heuristic (FS)	Technique to render ZKPs non-interactive using hash functions
Commitment Schemes (CS)	Commit to value v now and reveal commitment and v later
Mix Nets (MN)	Mix ciphertext to remove relation to previous ciphertexts

carbon paper, within an envelope, and seals it. The sealed envelope is given to a trusted registrar r. The registrar verifies (i) the identity and eligibility of v and (ii) verifies that v has not cast a ballot yet. Then, r signs the envelope, which includes the ballot on the carbon paper, and returns it to v. Now, the voter v can *unblind* the ballot and cast the signed carbon paper $ballot_v$ separately. In short, the use of BS [16] allows a registrar r to sign a message m without knowing its contents [44].

Another important cryptographic primitive is Homomorphic Encryption (HE). HE enables the use of encrypted data in such a way that its decrypted result is the same as if the operation was performed on the original plaintext data. Confidential data remains private while still being processed in an untrusted environment (e.g., a public cloud provider) [4, 31]. While fully HE is considered the holy grail of modern cryptography, partial HE is also useful for specific use cases, such as voting. HE is based on a homomorphism (i.e., a structure-preserving mapping) existing between plaintext and ciphertext operation, which can be defined with equality as follows:

$$\mathsf{Enc}_{\mathsf{pk}}(m_1) \otimes \mathsf{Enc}_{\mathsf{pk}}(m_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 \oplus m_2)$$

HE allows for combining two encrypted messages into one without the use of pk. The operators $\oplus$ and $\otimes$ represent arbitrary operations that depend on the implementation of the cryptosystem. This homomorphic property can be used to tally votes in a voting system. Every voter V can encrypt his or her ballot $ballot_V$ using the public election key pk, resulting in the encrypted vote: $e_{\mathsf{pk}}(ballot_V)$ [44].

Suitable cryptosystems that provide a homomorphism partially under a single operation are ElGamal [28] (cf. Sect. 3.1), the Paillier cryptosystem [64], and RSA [69]. However, ElGamal and Paillier are the most popular choice for REV schemes because both systems are non-deterministic, which is one reason why RSA, although offering homomorphic properties, cannot be used for REV systems. A non-deterministic cryptosystem can provide the security notion of IND-CPA, i.e.,

IND-CPA combines the security goal of indistinguishable ciphertexts in combination with the chosen-plaintext attacker-model. IND-CPA is the most important security notion in applied cryptography and is referred to as *semantic security*. IND-CPA captures the notion that ciphertexts should not leak any information about plaintexts as long as the key is kept secret. Hence, one way to achieve IND-CPA is to use *randomized encryption*, which implies that encryption is *probabilistic*, while decryption remains deterministic [5].

While HE enables the confidentiality of the ballots, the validity of the ballots needs to be ensured. In other words: how can voters prove that the ballot is a valid ballot and not an invalid ciphertext? A suitable tool is *Zero-Knowledge Proofs (*ZKP*)*, which allow a prover P to demonstrate knowledge about a secret to a verifier V, without leaking anything about the secret itself [23, 35]. A ZKP must satisfy the following three properties [35]:

1. Completeness: if the statement is true, the honest V will accept it.
2. Soundness: if the statement is false, it is not possible to convince an honest V that the statement is true.
3. Zero-knowledge: no malicious V can learn anything about the secret from the proof, except for its correctness.

ZKPs provide probabilistic soundness guarantees, i.e., the probability that a malicious prover can craft a valid proof for a false statement that an honest verifier will accept is negligible but non-zero. In REV systems, ZKPs can be used to prove that a ballot is valid and well-formed without leaking anything about its contents and also to prove the correctness of decryption without revealing the private key used.

Interactive ZKP are formulated as a challenge-response game in which P first commits to a statement, and secondly, the V sends a challenge to the P, to which P finally responds. These three-move protocols are referred to as Σ-protocols [23, 35]. The communication overhead in these protocols is not suitable for many use-cases. The so-called *Non-Interactive* ZKP (NIZKP) do not require interaction and can be generated by the P alone, while the same goes for the verification by the V, which does not require the P.

The so-called Fiat-Shamir Heuristic (FS) is the most common technique to render interactive Σ-protocols *non-interactive* by replacing the random challenge generated by the V with a hash of the algorithm's transcript, which then serves as the random oracle, i.e., the source of uniform randomness. This allows NIZKP to be easily generated and published, allowing anyone to verify a statement without information leakage.

Different protocols have made use of the FS heuristic to reduce communication overhead in REV schemes. Bernhard et al. [8] identified that two distinct applications of such heuristics exist in the literature: a strong variant (sFS) and a weak one (wFS). While sFS hashes the commitments and the statement, wFS *only* hashes the commitments. This can lead to unsound proofs in the case of malicious provers: i.e., proofs that do not actually prove the statement that they are supposed to prove.

NIZKPs are a central tool widely used in REV protocols in order to reduce trust assumptions and provide verifiable proof without leaking any information about the

data at hand. For instance, in HE-based schemes, ballot validity NIZKPs are used to prove that a ballot is formed correctly and does indeed contain a valid ballot (for instance, a 0 or 1 vote). Also, NIZKPs are used to prove the correct decryption of shared tallies, for instance, when multiple entities are in possession of a private part of the election key, which can be implemented using Secret Sharing techniques.

Secure Multi-Party Computations (MPC) allows the removal of any TTP from a distributed computation, enabling participants to compute the function on their private inputs with a cryptographic protocol. One example of an MPC is built on the idea of Secret Sharing [10]. With Distributed Key Generation (DKG), trust can be distributed, and the risk of one party decrypting individual ballots is mitigated. DKG can be applied with threshold cryptosystems, where a threshold of n out of k total amount of trustees is defined, which then collaborate to generate a new public key (i.e., the election key). In order to decrypt a ciphertext, collaborative decryption must be performed by a minimum of n participating trustees, which decrypt ciphertexts with their shared private key share, where the cipher decrypts to plaintext again with the last share submitted. With this approach, no single centralized trustee protects the secrecy of the voting, but it is split among multiple trustees.

Commitment Schemes (CS) are another essential building block for many cryptographic protocols. A CS is used to commit to a value v without revealing it at the time of commitment, but possibly revealing it later, i.e., a CS is useful when P must be prohibited from changing a value they have committed to, but the value should also stay private at the time of commitment [78]. For instance, in proof transcripts for NIZKPs, it is often necessary that P commits to a certain statement and later reveals that commitment (e.g., the order of candidates) [44].

As an alternative to using HE-based voting protocols, Mix Nets (MN) build the foundation of many REV systems. There are two types of Mix Nets: Decryption Mix Nets (DMN), which were first proposed by [16], and Re-Encryption Mix Nets (RMN), first proposed by [65]. Decryption Mix Nets (DMN) decrypt votes that are encrypted in as many layers as there are nodes in the DMN. Therefore, each layer corresponds to a key owned by a DMN node. Each node removes one layer of the encryption and permutes the input, repeating the process until the last encryption layer is removed and the plaintext result is revealed. Re-encryption Mix Nets (RMN) use multiple rounds of re-encryptions (or re-randomizations) of the ciphertexts, while only one decryption step is sufficient for decryption to the plaintext result. In RMNs, each mix node permutes the input and re-encrypts or re-randomizes the inputs to make them look completely different from the input. The last mix node performs a decryption step to recover the plain texts.

REV schemes make use of a wide range of cryptographic tools to construct voting protocols. From a practical point of view, security mechanisms based on well-established cryptographic methods, such as TLS (Transport Layer Security)-encrypted communication channels, and key management approaches are also required to be assessed closely. However, these mechanisms are not specific to REV and are in use in other areas as well, and the expenditure ultimately depends on the deployment use-case and budget of the REV system at hand. One thing that many of the security mechanisms do have in common with the REV specifics is the

underlying computationally hard problems that are discussed in the next subsection, after discussing the ElGamal cryptosystem in detail.

3.1 ElGamal Public-Key Cryptosystem

The ElGamal public-key cryptosystem was formulated in 1985 [28]. ElGamal is often used in cryptographic voting schemes because of the multiplicative homomorphism, which enables the encoding of votes in ciphertexts (cf. Sect. 3.2). Further, ElGamal offers semantic security (cf. Sect. 3), because ciphertexts are not deterministic but probabilistic. The computational hardness assumptions for ElGamal are based on the Decisional Diffie-Hellman (DDH) problem (cf. Sect. 3.2). The DDH assumption is commonly considered to hold in the subgroup of quadratic residues modulo a safe prime: $p = 2q + 1$ [11].

3.1.1 ElGamal Key Generation, Encryption, and Decryption

ElGamal is defined within the cyclic group $\mathbb{G}$. The cyclic group, $\mathbb{G}_q$ with prime order q, is defined over a finite field $\mathbb{Z}_p$, defined over positive integers modulo p, where p is an odd prime. The cryptographic operations are performed in a subgroup $\mathbb{G}_q \subset \mathbb{Z}_p^*$, a multiplicative cyclic group $\mathbb{G}_q$ defined over the finite field $\mathbb{Z}_p$. Alternatively, the cyclic group $\mathbb{G}$ can also be defined over an elliptic curve that is itself defined over a finite field [28].

In ElGamal, the public elements are $\mathbb{G}$, p, q and two independent generators g and h, where $g, h \in \mathbb{G}_q \setminus 1$. The private key sk is a random value: $\mathsf{sk} \in_r \mathbb{Z}_q^*$, where $q = \frac{p-1}{2}$. The public key pk is defined as follows:

$$(\mathsf{sk}, \mathsf{pk}) = (\mathsf{sk}, g^{\mathsf{sk}} \mod p)$$

The encryption of a plaintext message m is performed by using a random value $r \in_r \mathbb{Z}_q^*$.

$$\begin{aligned} E(m, \mathsf{pk}) = c &= (a, b) \\ &= (g^r \mod p, \mathsf{pk}^r \cdot m \mod p) \end{aligned}$$

The decryption of a ciphertext $c = (a, b)$ is performed by computing the multiplicative inverse of a^{sk} multiplied with b.

$$\begin{aligned} D(c, \mathsf{sk}) = m &= (a^{\mathsf{sk}})^{-1} \cdot b \mod p \\ &= (g^{r^{\mathsf{sk}}})^{-1} \cdot \mathsf{pk}^r \cdot m \mod p \\ &= (g^{r^{\mathsf{sk}}})^{-1} \cdot g^{\mathsf{sk}^r} \cdot m \mod p \\ &= (g^{r \cdot \mathsf{sk}})^{-1} \cdot g^{r \cdot \mathsf{sk}} \cdot m \mod p \end{aligned}$$

3.2 Message Encoding in ElGamal

The correct message encoding is crucial for voting systems because votes (i.e., messages) are encoded and then encrypted. In detail, messages in the ElGamal cryptosystem must be encoded, i.e., mapped with a reversible function to a group element, such that the DDH assumption holds [21, 27]. Otherwise, the ElGamal cryptosystem is not secure against chosen plaintext attacks (i.e., does not achieve `IND-CPA`). Hence, two major variants of message encodings are possible [26].

The first variant to perform a message encoding had been introduced in [21]. The core premise is to encode any message $m \in \mathbb{Z}_q$ as g^m. The corresponding ciphertext is given by $E(m, \mathsf{pk}) = c = (a, b) = (g^r, \mathsf{pk}^r \cdot g^m)$. For instance, the Provotum 2.0 system uses this message encoding format [49]. The encoding enables additively homomorphic operations making the ElGamal cryptosystem *additive*, instead of *multiplicative* homomorphic. A downside of this encoding is that the plaintext, resulting from the decryption, must be decoded by computing the discrete logarithm of g^m (i.e., $g^? = g^m$), although computing discrete logarithms is hard. It is only possible for a restricted message space $m \in 0, 1$ where brute-forcing or using the `baby-step-giant-step` algorithm is possible [27]. In the listing below, two ElGamal ciphertexts $E(m_1)$ and $E(m_2)$ are homomorphically summed. To achieve that, the components are multiplied, i.e., its exponents are added up, resulting in the equivalent sum as if the plaintexts m_1 and m_2 had been added together.

$$E(m_1 + m_2, \mathsf{pk}) = E(m_1, \mathsf{pk}) \cdot E(m_2, \mathsf{pk}) = (g^{r_1+r_2}, \mathsf{pk}^{r_1+r_2} \cdot g^{m_1+m_2})$$
$$m_1 + m_2 = \text{decode}(D(E(m_1 + m_2, \mathsf{pk}), \mathsf{sk}))$$

The second encoding variation ensures all messages are restricted to quadratic residues in a safe prime group $\mathbb{G}$ of order q. Using this approach, it is required to check that the equality: $m^q \mod p \equiv 1$ holds *before* a message is encrypted. If the equality does not hold, the message must not be encrypted [27]. A downside of the second variant is that the equality check needs to be performed *before* encrypting, thus increasing the computational overhead due to the modulo exponentiation. However, in real-world practical systems, the increase is negligible for a single operation.

3.3 Hard Problems and Post-Quantum Safety

Computationally hard problems (e.g., factorization of prime numbers as defined in RSA) are the backbone of public-key cryptography. These problems are easy to formulate but extremely hard to solve. For computationally hard problems, no algorithm will run in a reasonable amount of time [5], and they are often also referred to as *intractable* problems. Most REV systems use cryptographic primitives, which are in turn based on hard problems. BCs and DLs are also based on

modern cryptography and, thus, are also based on such intractability assumptions. It is crucial to assess these problems against current and future adversaries, especially with the advent of quantum computing, which may break certain hard problems. Considering BCs and DLs, data is persisted on publicly readable storage, which means that future adversaries could store that data now and break the underlying schemes in the future (e.g., store election data now and break ballot secrecy in the future). Therefore, these assumptions directly impact the long-term privacy and verifiability properties because they are often assuming hard problems. Hence, it is important to define the assumptions and hard problems that are considered by these primitives and the direct implications on the security notions.

Hard Problems Most voting schemes are based on public-key encryption methods and are thus based on assumptions of computational complexity [44]. Since every major public-key cryptosystem deployed today relies on either factoring (using the RSA algorithm) or the Discrete Logarithm Problem (DLP), systems using these schemes inherit the underlying assumptions. One of the key hard problems that modern cryptography is built on is the *Factoring Problem*, which is defined as finding the prime numbers p and q, given a large number $N = p \times q$. The widely known and used RSA algorithms are based on the factoring problem. The security of the factoring problem is based on the fact that factorization of prime numbers is hard, i.e., it is hard to decompose N as a product of prime numbers.

The *Discrete Logarithm Problem (DLP)* is similar to the factoring problem. DLP describes a *discrete* problem, which means it deals with integers, as opposed to real numbers (which would be called continuous). Further, it deals with a *logarithm*, i.e., the inverse function of exponentiation. Also, it deals with mathematical groups of prime order modulo p, which are a set of elements (typically numbers) that are related to each other according to well-defined rules written as Z_p^*. The DLP is defined as finding the y for which $g^y = x$, given a base number g within some group Z_p^*, where p is a prime number, and given a group element x.

The last group of hard problems is the Diffie-Hellman problems, which are a special variation of the DLP. The *Computational Diffie-Hellman (CDH)* problem is a variation of the DLP. The CDH describes computing the shared secret g^{ab} given only the public values of g^a and g^b, and not any of the secret values a or b. If an eavesdropper was to obtain g^a and g^b, it should not be possible for her to determine the shared secret g^{ab}. Lastly, the *Decisional Diffie-Hellman (DDH)* problem is stronger than the CDH. The DDH assumes that the attacker is not able to compute g^{ab} partially, given the values of g^a and g^b, i.e., the attacker cannot learn *anything* about the shared secret. To ensure that, the value of g^{ab} needs to be indistinguishable from a random group element.

The importance of these problems may not be obvious at first glance, but most of the cryptographic methods used in everyday communications and the Internet are based on using these hard problems. Therefore, the problems are *assumed* to be hard to solve. These assumptions build the foundation of all major infrastructure used in modern computer networks and in many voting protocols and REV systems and BC and DL networks. The implications of that are clear:

REV systems, BCs, and DLs alike base their security on public-key cryptography and hash functions. These primitives are used to authenticate transactions (or ballots) with signatures and are also used in communication channels and consensus algorithms to identify authorities (e.g., PoA). Collision-resistant hash functions are used to link together blocks of transactions, effectively forming the chain or ledger. Advances in quantum computing directly threaten these building blocks, since it is estimated that in the next 20 years, a powerful Quantum Computer (QC) will be functional enough to break current strong public-key cryptosystems by breaking the underlying hardness assumptions [30]. The research community is tackling these issues with many post-quantum cryptography efforts and various novel approaches for post-quantum encryption and signatures. For example, the NIST is conducting standardization processes to establish solid post-quantum public-key cryptography mechanisms [62].

4 Overview of REV Approaches

Over the years, many REV approaches have been developed, evaluated critically, and refined. Thus, it is essential to describe these approaches, outlining their core ideas and adding necessary elements on selected cryptographic elements which may not have been covered in Sect. 3, providing the required background to follow the principle approaches outlined hereafter. Inspired by [9, 13], the following figures' organization serves as an essential reference and role model to introduce REV schemes and their specific challenges. The following approaches highlight the inherent problems of creating an REV system in practice. Privacy, verifiability, and trusted parties are given access to overview major features, characteristics, and concerns that are still partially present. While the PBB is an essential part of any of these approaches (and REV systems in general), the following subsections focus on the approaches, regardless of the practical deployment methods used (e.g., be it a centralized server, a BC, or a DL). For a dedicated discussion of practically decentralizing any of these approaches, Sect. 5 outlines, discusses, and evaluates the potential directions and deployments.

4.1 Encrypt and Sign Approach

A basic approach toward an EV scheme combines an encryption scheme with a digital signature scheme, as outlined in Fig. 4. The encryption scheme provides Ballot Privacy, while the signature scheme ensures the data integrity of the sender's transmitted message and authentication. During ongoing voting, the secrecy of the voter's ballots is ensured. However, since the voting service has the private election key (i.e., the counterpart of the public election key used to encrypt the ballots), the digital signatures are directly linking the encrypted ballots to the voters [13].

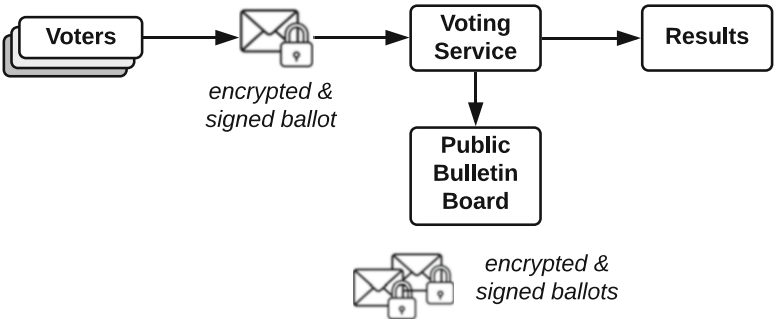


Fig. 4 Basic remote E-voting architecture based on a simple encrypt and sign

Thus, the voting service acts as a single trusted entity and can break ballot secrecy individually (e.g., decrypt individual ballots without anyone realizing). Also, a breach of the voting service's security could potentially leak the private key of the election and allow for complete decryption of the PBB, breaking ballot secrecy in full. By publishing a list of eligible public voter credentials, anyone can verify the signatures on the ballots. However, the major issue is that the final results cannot be verified without any additional mechanisms because the voting service would break ballot secrecy by decrypting ballots and publishing their plaintexts. While individual voters can verify whether their ballot was correctly stored on the PBB, they cannot verify the final results and fully trust in the voting service. In summary, the encrypt & sign approach is feasible for scenarios where voters choose to trust the voting service fully.

4.2 The Two Agencies Approach

Another popular voting scheme was first proposed in 1992 by [32], often referred to as the *two agencies model* [67], FOO92, or just FOO [9]. It is one of the first and most influential REV schemes designed for practical use. A simplified version of this voting scheme is shown in Fig. 5. The Validation Service authenticates the voter (checks eligibility) and then provides an anonymous token to the voter. The voter sends the encrypted ballot with the anonymous token to the Voting Service, which only appends the ballots if the Validation Service issued the anonymous token. The introduction of the validation service as a new TTP removes the link between the encrypted ballots and the voters' identities.

Voting schemes based on the two agencies model usually use the cryptographic primitive of Blind Signatures, which allows a signer to cryptographically sign a message without viewing its content in plaintext. Blind Signatures were first introduced in [16] as a means to establish untraceable and anonymous electronic payments. In the context of Fig. 5, the Validation Service can digitally sign the ballot

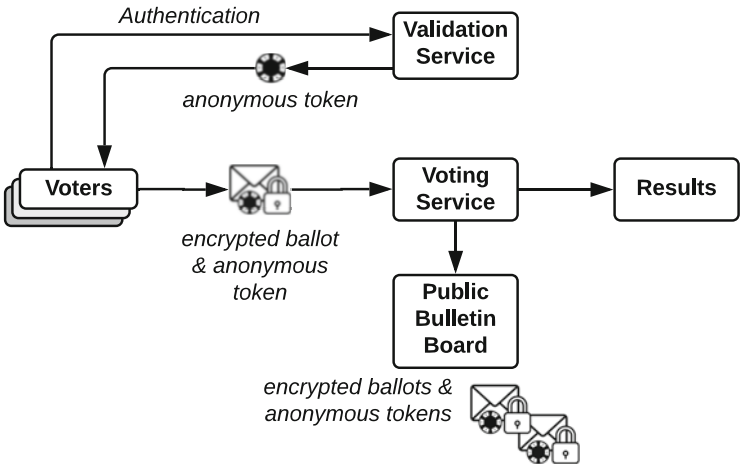


Fig. 5 Two agencies model with anonymous tokens (e.g., Blind Signatures)

without viewing its content (i.e., signing the blinded ballot). The voter receives the signed ballot, removes the blinding factor, and then casts the unblinded but signed ballot to the Voting Service, which validates whether the signature originates from the Validation Service. Even if the Validation Service colludes with the Voting Service, the token cannot be traced back to the voter since it was signed in its blinded form. Ballot Secrecy is assured in the final tally along the same line of arguments because the link between the vote and the token is removed. However, collusion between the two trusted agencies would make it rather easy to identify voters and break Ballot Privacy by correlating the information. Another major threat is *ballot stuffing* because the Validation Service could create valid tokens for non-eligible voters, which the Voting Service would accept. With this scheme, the voting service is still able to decrypt individual ballots since it owns the private election key.

4.3 Homomorphic Encryption Approach

Another major approach to achieve a private and verifiable REV is by deploying HE. In Fig. 6, an exemplary homomorphic tallying system is depicted. The ballot must be encrypted according to the format defined by the voting service. For instance, in this case, three ciphertexts of binary values, either 0 or 1, are encrypted three times, once for each voting option A, B, and C. Additionally, the ciphertexts may be cryptographically signed (for eligibility verification), and usually, the cryptographic proof is generated (e.g., a Non-Interactive Zero-Knowledge (NIZK) proof, more on that in Sect. 3). Because not a single ciphertext is decrypted during tallying (only the result), the proofs are crucial to ensure the ballot's validity and mitigate the risk that malicious voters could encrypt invalid values. Therefore, during tallying, the

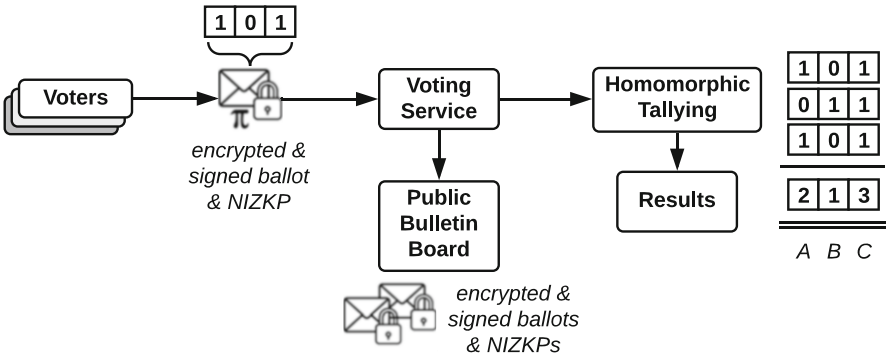


Fig. 6 Homomorphic Encryption-based REV system (e.g., ElGamal)

underlying cryptosystem's homomorphic properties are used to add the ciphertexts together and then only decrypt the final result. The major drawback of HE-based systems is that the number of possible voting options is limited (e.g., write-ins cannot be supported). A precise representation of the ballot is required.

Despite these limitations, an interesting advantage of HE-based systems is the straightforward distribution of decryption keys by using threshold cryptography. In such a case, each trustee holds shares of the private key, which is required to decrypt the final tally. In 1997, the seminal work [21] proposed the use of threshold cryptography in an REV system considering a multi-authority setting that requires each key share holder to submit partial decryptions. Then, the final decryption can only be performed if the pre-defined threshold of authorities collaborates.

An example voting system as depicted in Fig. 6 usually requires trustees to perform a Distributed Key Generation (DKG) algorithm, which results in a public election key pk_e, while each trustee holds a secret key share sk_e. Then, voters form a valid ballot and encrypt their ballot by using pk_e. Further, voters generate a NIZK proof proving that the ballot is well-formed and valid before sending both to the voting service. Upon receiving the ballots, the voting service authenticates voters and validates the proofs, ultimately storing them in the PBB. After the voting period ends, the homomorphic tallying is performed among all trustees over the set of all stored and valid ballots. For that, the trustees use their secret key shares to homomorphically tally the ballots together (without decrypting individual ballots), only decrypting the final tally and publishing it to the public.

4.4 Mix Network-Based Approach

Besides HE, Mix Networks (MN) are the primary building blocks for deployed REV systems used in real-world elections in Switzerland, Australia, Estonia, and Norway [40]. MNs are versatile cryptographic protocols that can be tailored to

other privacy-sensitive use-cases, such as anonymous communication protocols for messaging or routing.

In principle, MN-based approaches are similar to paper-ballot voting schemes where the paper ballots are shuffled after they are separated from any voter-identifying information. As seen in Fig. 7, once the ballot is no longer traceable through the MN, any correlation of the ballot before and after the MN is impossible. For the final tally, each ballot is decrypted individually in contrast to HE, where only the ciphertext of the final tally is decrypted. Further, MN-based schemes can distribute trust by leveraging threshold cryptography (similar to HE-based schemes cf. Sect. 4.3), for instance, by using multiple trustees to perform the MN re-encryption and final decryption ceremony.

4.5 Discussion of Approaches

The majority of current REV systems are based on either a MN- or a HE-based approach. In Table 2, an overview of various REV systems is provided, including academic implementations, enterprise and commercially oriented products, and country-specific implementations. The table is based on [6, 79], but focused only on REV schemes, with additional properties added, and it assesses the REV systems with regard to their properties based on the publication references listed alongside them. Most of these voting systems are evolving and being researched actively. Thus, novel improvements may be performed, and the table may not present a static view of these properties and systems.

The majority of REV systems provide privacy, although the details and assumptions vary depending on the voting protocol and implementation. Some systems also provide mechanisms enabling RF and CR. Often, these mechanisms require additional procedures or additional TTPs and services that perform specific actions, such as in [14] where one additional assumption is the existence of a re-randomization service that provides a re-encryption of ciphertexts so the voter does not possess a receipt of the encrypted ballot, but is still assured that the ballot is valid. Systems that provide CR, such as Civitas [19], allow voters to create fake credentials (which are not excluded in the final tallying) that a coercer cannot distinguish from real

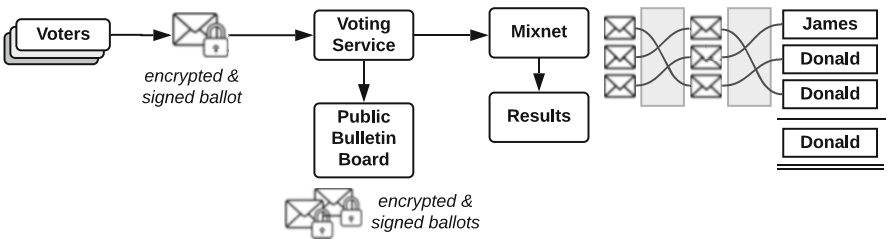


Fig. 7 Mix network-based REV system

Table 2 Comparison of related work on REV systems ● = Provides Property, ○ = Does Not Provide Property ◐ = Partially Provides Property. A Full Overview is Available in [8]

	Ballot Privacy	Receipt Freeness	Coercion-Resistance	Everlasting Privacy	Unconditional Privacy	Software Independence	Individual Verifiability	Universal Verifiability	Eligibility Verifiability	Cast-as-Intended	Recorded-as-Cast	Tallied-as-Recorded	Transparency	Decentralized PBB	Voting Protocol
Academic Implementations															
Helios [2]	◐[1]	○	○	○	○	●	●	●	○	●[2]	●	●	●	○	HE
BeleniosRF [14]	●	●	○	○	○	●	●	●	●[4]	○	●	●	●	○	HE
Civitas [19]	●	●	●	○	○	○	●	●	●[4]	○	●	●	●	○	RMN
Selene [71]	●	●	○	○	○	●	●	●	○	●[5]	●	●	◐[6]	○	RMN
Verify-Your-Vote [15]	●	●	●	○	○	●	●	●	●[4]	○	●	●	◐[6]	●	HE
OVN McCorry [57]	●	○	○	○	○	●	●	●	◐[7]	○	●	●	●	●	HE
Provotum 2.0 [48]	●	○	○	○	○	●	●	●	●	○	●	●	●	●	HE
Æternum [46]	●	◐[3]	○	●	●	●	●	●	●	○	●	●	●	●	CS
Enterprise Implementations															
nVotes [61]	●	○	○	○	○	●	●	●	○	●[2]	●	●	○[8]	○	RMN
Voatz [77]	○	○	○	○	○	○	○	○	○	○	○	○	○[8]	◐[12]	n/a
Country-specific Implementations															
CHVote [35]	●	○	○	○	○	○	●	●	●[4]	●[9]	●	●	●	○	RMN
Estonia [41]	●	○	○	○	○	○	●	●	○	●	●	○	◐[10]	○	RMN
Norway [32]	●	○	○	○	○	○	●	○	○	●	○	◐[11]	●	○	HE

[1] Vote copying possible
[2] Cast or challenge
[3] Extension available in [54]
[4] Uses trusted registrar
[5] Used e2e tracking number
[6] No source code available
[7] Needs trusted contract creator
[8] No source code, no documentation available
[9] Mailed verification code
[10] Partial source code available
[11] Needs trusted auditor
[12] BC deployed in private cloud

credentials. Besides coercion, long-term privacy, such as EP, is a major challenge for REV systems because most schemes are based on public-key cryptosystems, and privacy assurance relies on computational hardness assumptions (cf. Sect. 3.3). Also, achieving UP, i.e., not assuming any trusted authority for Ballot Privacy, is arguably impossible in practical HE or RMN systems since the decryption keys are split among trusted authorities. Also, protocols that provide UP rely on the existence of an anonymous channel to cast votes [48, 56] while still relying on intractability assumptions for verifiability properties (i.e., during the vote casting period).

Verifiability is a core property of REV systems, and its sub-properties are fundamental to such systems. As outlined in Sect. 2.3.2, if a system achieves IV and UV, that does not automatically imply that the system provides E2E-V. Achieving IV is done by enabling voters to check the correct inclusion of the ballot on the PBB. This creates tension with RF. UV is usually achieved by providing publicly verifiable proof that the decryption was performed correctly. Hence, ballots from the PBB were included and tallied correctly. The main challenge in achieving E2E-V is providing the ability for voters to check whether their vote was CaI.

One popular way to achieve this is the *Challenge-or-Cast* method, where a voting device's ballot encryption can be challenged by sending the ballot's parameters to a second trustworthy device that is used to re-perform the encryption steps. This verification method is used in [63]. Another option to achieve E2E-V includes *Code-Voting*, which requires a trustworthy printing office, as used in the CHVote voting system [37].

Another key aspect of REV systems is Eligibility Verifiability, which is tightly coupled to the IdM of voters, i.e., whether or not it is possible to verify whether eligible voters only vote once. Usually, REV systems assume external ballot authentication systems, i.e., a TTP that authenticates ballots by checking eligibility or distributing the credentials before the voting occurs. The variety in approaches also stems from IdM systems' heterogeneity across industries, use-cases, and geographical regions. Only a few Internet voting schemes have explicitly incorporated Eligibility Verifiability and provided defenses against ballot stuffing, e.g., JCJ [46] and Civitas [19], as well as Cobra [29].

The Transparency assessment in Table 2 mainly concerns the publication of the system's source code and additional documents that describe the voting protocol (e.g., security proofs). Transparency is crucial for trust in any REV system. In Switzerland, the Swiss Federal Government published an Ordinance [25] that demands an REV system publish its source code if it is used within a national context, with differing tiers of electorate percentages that would use said REV system.

Further, the deployment of decentralized PBBs is not widely adopted. Most of the REV systems included in Table 2 rely on centralized PBBs deployed as databases and, thus, a strong trust assumption in whoever is in charge of the database operation. Decentralized PBBs are, for instance, deployed on the public permissionless Ethereum mainnet [59] or deployed in a public permissioned PoA Ethereum network [15]. Also, dedicated DL and BC developments are proposed as decentralized PBBs [15, 49], which significantly improve the trust assumptions, i.e., instead of trusting one operator, trust is decentralized among multiple.

Voting protocols are the basis of the voting systems, and the majority of systems either rely on homomorphic or MN-based tallying. There are multiple key differences between HE- and MN-based approaches. First, with HE, the ballots' public aggregation is performed into the final outcome, whereas with an RMN, trustees (i.e., trusted mixers) anonymize the ballots by performing re-encryption. Also, with HE, only the outcome is decrypted, which means that little information is revealed and little computation is needed, whereas in MN-based tallying, more info is revealed, and the computation grows with the number of voters. Another key difference is the ballot validity checks. In HE-based schemes, the ballot validity can be checked at submission time because the voters perform heavy computations to generate the proof for ballot validity before casting. The proof needs to be adapted according to the election rules. In MN-based schemes, the validity check is performed *after* decryption, which means that the voters do not need to provide a ballot validity proof, and a universal ballot format can be used because ballots will be decrypted after mixing. In conclusion, the major burden in HE-based schemes

lies with the voters who need to prove the validity and ensure the ballot is cast correctly to the PBB. In MN-based schemes, the major burden is on the voting authorities who will perform the mixing. Another major responsibility is with the trustees that need to perform the mixing and decryption [12].

5 Research Directions Toward a Decentralized REV System

The primary goal of REV systems is to achieve a privacy-preserving and end-to-end verifiable voting process. Those approaches, as outlined in Sect. 4, show various directions to reach an overall secure process. While the primary responsibility of the PBB is to persist verifiable information immutably, the architectural model is not always considered carefully since the existence of such a PBB is either explicitly or implicitly stated as an assumption. Also, depending on the voting protocol chosen, the trust model is not influenced substantially either. For instance, if a single TTP stores private keys for a given election, the integrity depends solely on that TTP, whether the PBB is deployed in a distributed setting or not.

It is necessary to differentiate between a public permissionless BC or a public permissioned DL operating as a PBB since the underlying consensus algorithm's assumptions differ and impact key assumptions on trust (cf. Sect. 2.2). For example, a BC (e.g., Bitcoin) follows a permissionless, or generally spoken "trustless," and fully decentralized trust model, where it is not necessary to trust entities, only (i) their actions, which are publicly persisted on the BC and (ii) those cryptographic mechanisms in use. This implies a paradigm shift from placing trust directly in a centralized entity instead of trusting a decentralized platform's cryptographic mechanisms, which enables transparency and public verifiability. However, it is crucial to consider privacy aspects ensuring that Ballot Privacy (BP) is maintained under all circumstances, which means that, e.g., Bitcoin's fully public and permissionless approach becomes inadequate for guaranteeing scalable confidentiality.

Therefore, a generic centralized voting system's architecture is essential to take these considerations into account. Based on such a proposal as of below, its trust assumptions are analyzed, and a broader risk assessment is performed. This leads to a new, fully decentralized architecture proposal and a discussion of its trade-offs.

5.1 Centralized REV System

The architecture of a centralized REV system (cf. Fig. 8) is decoupled from a specific voting protocol. Major stakeholders include voters and one Voting Authority (VA). Voters are provided with the VA's voting credentials and are equipped with technical means to cast their ballots in their voting client. These credentials can be provided electronically (e.g., via electronic mail) or physically (e.g., via postal mail) by a VA for either mixing or decrypting the final tally of votes. In most cases,

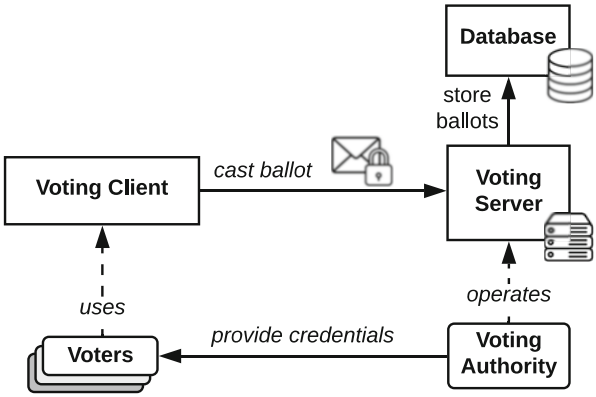


Fig. 8 Centralized REV system's architecture

a voting client is a software executed on voting clients, which can be considered personal devices owned by voters, e.g., typically Web browsers fetching the voting client software from a voting server. This allows for casting a remote vote in which the ballot is sent to the voting server to be persisted on a database that acts as a PBB.

For any properties to be evaluated, all trust assumptions need to be formulated beforehand. In essence, the centralized REV system requires a fully trusted VA that operates, stores, and manages cryptographic election keys to calculate the final tally. Further, the VA needs to operate a trusted server that persists all ballots cast and ensures the integrity and immutability of these ballots. While the server can be queried from the outside, the server's data is the single source of truth.

Assuming that a centralized REV system is based on a HE-based (Homomorphic Encryption) voting protocol, the homomorphic tallying process does not decrypt single ballots but homomorphically adds the tally and only decrypts the resulting tally. If the system is employing an MN (Mix Net) voting protocol, the mixing ensures BP. However, without any additional mechanism, the system does not provide Receipt Freeness (RF), because the voter can produce a receipt of the encryption procedure and its randomization parameters. Subsequently, Coercion-Resistance (CR) can also not be guaranteed since a coercer could force a voter to leak his/her credentials. Also, since the centralized VA generates the public election key, single ballots could be decrypted, thus, breaking BP altogether. While this threat can be mitigated by using threshold cryptography and splitting up the key to different trustees, the voting server's storage would still present a Single Point of Failure. For the decryption, the trustees would be required to interact with said voting server securely.

The ElGamal cryptosystem is the most popular choice for REV systems, used in both HE- and MN-based schemes. It bases its security on the DDH (Decisional Diffie-Hellman) problem (cf. Sect. 3.3). Therefore, the centralized approach does not provide Everlasting BP (EP), because an adversary can query the PBB, break the DDH problem in the future, and restore the private election key to decrypt all ballots to plaintext. Lastly, Unconditional Privacy (UP) is also not given since BP

relies on a trusted VA, i.e., if the VA is compromised or malicious, BP cannot be guaranteed. Thus, the *condition* for BP is trust in the VA.

All verifiability properties in any centralized approach rely heavily on trust assumptions, i.e., these properties *rely* on trusted VAs to persist and safeguard data integrity. For instance, to achieve Individual Verifiability (IV), voters can query the central database and check whether their vote is included in the set of ballots. Achieving Universal Verifiability (UV) is significantly more complex in a HE- or an MN-based scheme. For instance, the system can use a NIZKP to prove the correct decryption of the final tally or the final mix step, which the electorate can verify. Any centralized approach does not provide eligibility verifiability as such since it would require someone to verify that only eligible voters voted only once. Since the VA is trusted to provide credentials to all voters directly, the VA could create fake voters or cast votes with unused credentials. Without an Eligibility Verifiability, the final tally is not trustworthy. The first property of End-to-End Verifiability (E2E-V) is Cast-as-Intended (CaI) verifiability, which requires a method for voters to verify whether the vote had been cast as it was intended, i.e., whether the encryption was performed correctly by the voting device. The centralized approach does not provide any method to check whether the voting client's operations were correct and, hence, does not provide CaI verifiability at all. Nevertheless, the centralized approach can provide Recorded-as-Cast (RaC) verifiability since it can be checked by querying the central database and checking whether the ballot had been persisted correctly. Tallied-as-Recorded (TaR) (also referred to as Counted-as-Recorded, CaR) verifiability is given when the final proof of correct decryption verifies successfully.

5.2 Decentralized REV System

A major drawback of any centralized architecture is the dependency on trust on a single VA, i.e., the privacy and verifiability properties only hold under the assumption of a single trusted VA. Therefore, an approach for a fully decentralized and end-to-end verifiable architecture based on a public permissioned DL is provided here. The generic decentralized architecture for an REV system (cf. Fig. 8) leads to the observation that the significant difference of a centralized architecture is the decentralized deployment of the PBB, assumed to be operated as a public permissioned DL. On the voter's side, the process is similar to the centralized variant, but the voter *directly* interacts with the DL by running their own DL node or by communicating with a DL node operated by a TTP. Also, all VAs should operate DL nodes. These nodes form the public permissioned DL network, which the public can query, but only trusted VAs can sign and append new entries to.

The architecture (cf. Fig. 9) is agnostic toward any voting protocol selected. However, the voting protocol needs to include an approach that allows for the distribution of trust. By deploying a DKG scheme [33], the private election key is not held by a single authority. Finally, the authorities need to submit their decrypted

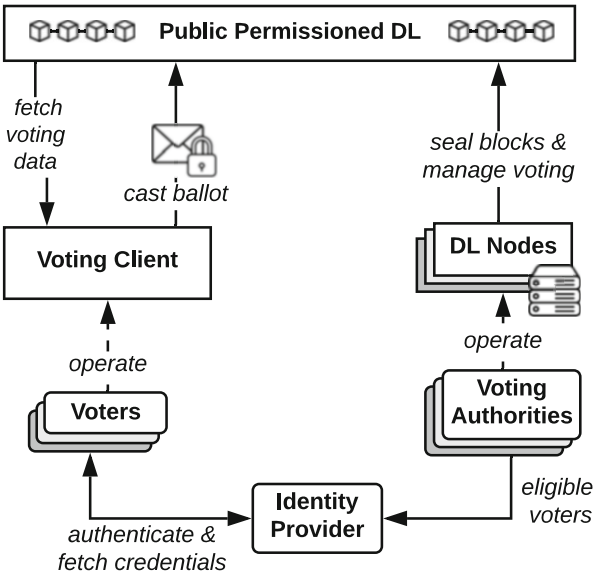


Fig. 9 A fully decentralized and generic high-level REV system's architecture

share of the final tally to decrypt the result collaboratively [50]. In that sense, the public permissioned DL serves as a transparent, tamper-proof platform, allowing the voting protocol to execute publicly shared information, which the electorate and auditors verify. Given a voting protocol based on a public key cryptosystem, voters encrypt their ballots with the public election key and send the encryption to the DL network directly, where anyone can verify the inclusion of the respective ballots [50].

The major benefit of using a decentralized architecture is the use of a suitable permissioned consensus algorithm to decentralize trust in the PBB. Many REV systems assume a centralized trusted PBB that provides a consistent view of the election's evidence trail. The implementation of cryptographic schemes into practical systems deployed on centralized PBBs does not hold under real-world conditions and breaks verifiability claims (e.g., integrity and availability guarantees). Thus, by decentralizing the PBB, the risk of a single entity breaking the integrity and consistency of the PBB is mitigated completely. The selection of an appropriate consensus algorithm is fundamental to the network's security characteristics [85]. Considering permissioned networks, where a set of initially trusted authorities is available (e.g., VAs, Identity Providers, and other TTPs), the Proof-of-Authority (PoA) consensus algorithm is suitable. Killer et al. [49, 50] proposes such a PoA-based PBB, allowing for the decentralization of trust. With a PoA consensus mechanism, the maximum number of tolerated malicious nodes is $\frac{N}{2}$ and $\frac{N}{3}$ for PBFT (Practical Byzantine Fault Tolerance), which is fully suitable for trusted VAs in operation. The selection of PoA is aligned to those REV system's general

requirements (cf. Sect. 2.4) since a set of trusted authorities is tasked with operating the voting or election. Thus, authorities allowed to propose and sign blocks (also called *minting*) are defined *a priori*, enabling a bootstrapping stage, where all authorities are already defined.

In summary, decentralizing the REV architecture allows for the distribution of trust on a public permissioned DL. The key benefit of a decentralized deployment is that trust assumptions are congruent with (i) trust assumptions of the voting protocol, i.e., threshold cryptography allows for distributing trust among VAs, which generates a public election key, as well as decrypting the final tally collaboratively. Secondly, benefit (ii) is that the trust is decentralized on a DL-consensus layer, since the PoA consensus only accepts blocks from pre-defined authorities. Thus, the decentralized architecture directly maps trust assumptions to a practically deployed system. Although these aspects present clear advantages, the translation of theoretical voting schemes to practical implementations always bears risks that require further research, e.g., there exist dedicated security threats on every layer of the system such that they need to be investigated in full detail before these systems reach fully practical feasibility.

5.3 Open Issues and Challenges

The previous section outlined a path toward a decentralized REV architecture. However, even with these major and measurable benefits of a decentralized PBB as identified, a couple of dedicated open issues remain:

- **Long-Term Privacy:** If the voting protocol's encryption scheme is based on a public-key cryptosystem (e.g., the ElGamal cryptosystem [28]), long-term privacy (e.g., EP and UP, cf. Sect. 2.3) issues need to be considered. For instance, the ElGamal cryptosystem bases its security on the DDH problem (cf. Sect. 3.3), which could be broken in the future [30]. Also, privacy is not provided unconditionally, because trusted authorities that hold a private key-share could conspire (or be compromised) and break BP.
- **Verifiability:** Achieving E2E-V in a decentralized architecture first requires a mechanism to achieve CaI verifiability (cf. Sect. 2.3), which allows voters to verify, whether the voting client correctly encrypted the ballot. The CaI verifier should be provided by DL nodes as well, decentralizing trust as well. Secondly, RaC verifiability can be checked by querying the PBB directly, without any TTPs. Similarly, TaR/CaR is achieved by storing the decryption proofs (i.e., including the necessary evidence trail) of the final tally on the public DL. Thus, anyone, including the voters, can verify that the steps were computed correctly. Attempts at manipulation would be immediately noticeable to any observer.
- **Voter Authentication:** Another crucial aspect of any REV system is ensuring that only eligible voters vote and that they only vote once. For instance, in Fig. 9, the Identity Provider (IdP) is responsible for verifying voter's eligibility. DL

nodes verify that each transaction containing a ballot originates from a public key through a blind signature provided by an IdP. Due to this mechanism, the IdP cannot trace an identity to a wallet address and a ballot. The IdP could also be decentralized in a way that is similar to the public permissioned DL by using threshold cryptography.

Further, real-world REV system deployments as partially discussed above made clear that open issues require future research to tackle interdisciplinary advances and challenges [8]. Future research requires the combination of theory and practice, combining both pillars of practical and theoretical research to formulate voting *protocols* in working *systems*. The *theory* pillar is formed by design techniques (e.g., Public-Key Cryptography [78]) and methodologies for verification (e.g., Applied pi calculus [72]) of voting protocols. The *practice* pillar combines implementation techniques (e.g., X.509 PKI [81]) with analysis methods (e.g., Attack Graphs [77]) toward practical systems [36] (Fig. 10).

On theoretical and practical pillars, open issues must be addressed before real-world REV becomes practically feasible. The following list of issues follows the structure of [8, 66], where many crucial and general security considerations for REV are stated. These aspects apply to all REV systems, whether votes are stored on either public permissionless BCs or permissioned DLs, since, either way, the PBB is only one aspect of the voting system as a whole, i.e., these security aspects are independent of the fact, whether the PBB is decentralized or not.

Generally speaking, all *stakeholders and adversaries* need to be clearly defined, including their capabilities, and responsibilities need to be assigned in full. The major formulation of trust assumptions builds the foundation for the system's security guarantees if any of the stakeholders is malicious. Hence, the fundamental questions include: *Who are the stakeholders and adversaries, and what are the trust assumptions, capabilities, and responsibilities within the voting process?*

Another high-level aspect is the *security model*, which needs to be carefully formulated in the context of the given voting system. A security model consists of security goals and a threat model, and these aspects are highly dependent on the relative importance of the election. Arguably, political elections should always

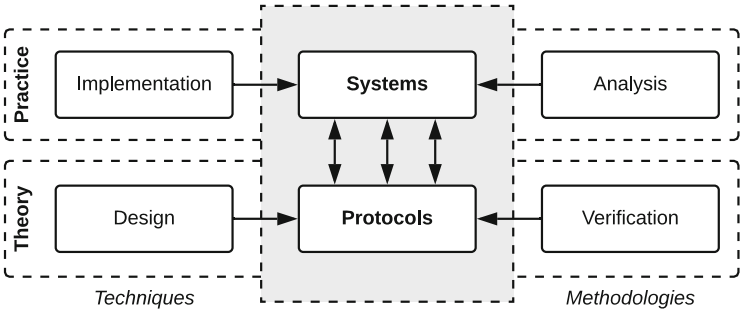


Fig. 10 Overview of research of REV, based on [36]

consider the mitigation of a state-sponsored and powerful attacker, while a risk assessment should always include attacks on all system areas. Hence, another crucial question is: *What are security goals and threat models the system needs to address, and under which trust assumptions do these goals hold?*

Furthermore, important questions regarding the key management deployment infrastructure need to be considered since they form the foundation of an REV system's operation that will be executed via the Internet. Additionally, key challenges in REV systems based on a fully decentralized PBB based on BC or DL must be considered carefully. First of all, (i) the decentralization of all single trusted entities should be enforced by the voting protocol and system. Second, (ii) inherent threats and risks of deploying a decentralized PBB (e.g., Distributed Denial-of-Service) requires specific mitigation techniques without introducing censorship. Third, (iii) the verification of the correct execution of the voting protocol and system should be provided on a decentralized PBB as a publicly verifiable audit trail. Possibly, a public and private environment is necessary to provide a suitable basis for [52]. Lastly, (iv) long-term privacy aspects (as well as vote-buying and coercion-resistance) are crucial and need to be addressed depending on the threat model and the election at hand because future adversaries may break computational intractability assumptions [48].

6 Overall Considerations

In an ideal election scenario, voters would express their voting choices remotely over the Internet and check each election stage transparently and independently. However, there are many challenges, ranging from societal to technological, which hinder the possibility of a remote election. Within the technical scope, the biggest challenge is to provide a solution that strikes a balance between privacy and verifiability embedded within the governmental and legal structure of a given democratic country. In this sense, this chapter explicitly outlined the current progress and the open challenges toward remote and decentralized elections.

Distributed Ledgers (DL), in their different flavors of permissioned and permissionless systems (including Blockchains – BC), stand out as a platform with the potential to provide fully decentralized elections, satisfying requirements in terms of privacy and verifiability based on existing cryptographic protocols. Public Permissioned DLs address a fundamental issue of REV systems' architectures: the strong assumption of a trusted PBB. REV systems require an immutable, publicly accessible, and verifiable audit trail. Hence, a permissioned DL can serve as a PBB for an REV system. A public permissioned DL's trust assumptions show a substantial improvement over a centralized PBB, where a single stakeholder needs to be trusted with the key management and operations. In contrast to a centralized deployment, a permissioned network can serve as a fully decentralized PBB, allowing for the distribution of trust on every layer of the voting system.

Transparency is a key aspect of public trust in the electoral process, and thus, a publicly readable DL can provide such an evidence trail.

> The peaceful transfer of power depends on confidence in the electoral process. That confidence should not automatically be given to any outcome that seems plausible – it must be earned by producing evidence that the election result is what the people chose. (Bernhard et al. [6])

However, there are major open issues in the interdisciplinary research of Electronic Voting in general. In particular, the constant threat of cyberattacks and security issues has created widespread awareness of cybersecurity issues. Even if all the crucial security considerations needed for high-stakes elections and voting systems are deployed, REV systems are still highly complex software/hardware systems that are practically impossible to deploy without any implementation errors. Therefore, it is fundamental that the REV system provides forms of dispute resolution and error detection. The software-independence property describes these idealized notions where implementation errors should be recognized effectively and do not go unnoticed. In essence, the cryptographic voting protocol should allow for implementation errors to be detected and optimally rolled back to recover the system's state. Also, more robust forms of verifiability, such as accountability, should incentivize stakeholders to act honestly. Hence, the critical challenges in REV remain the same: to strike a balance of privacy and verifiability while ensuring the integrity of the vote. Thus, future research on REV requires interdisciplinary approaches, combining theoretical voting protocols into practical and working systems, which decentralize the voting process, inherently reducing the potential abuse of power and trust. In an ideal election scenario, voters would express their voting choices remotely through the Internet and check each election stage transparently and independently. However, there are many challenges, ranging from societal to technological, which hinder the possibility of a remote election. Within the technical scope, the biggest challenge is to provide a solution that strikes a balance between privacy and verifiability embedded within the governmental and legal structure of a given democratic country. In this sense, this chapter aimed to explicitly outline the current progress and the open challenges toward remote and decentralized elections.

Acknowledgments This work was supported partially by (a) the University of Zürich UZH, Switzerland, and (b) the European Union's Horizon 2020 Research and Innovation Program under grant agreement No. 830927, the Concordia Project.

References

1. B. Adida, Advances in cryptographic voting systems. Ph.D. thesis, Massachusetts Institute of Technology, 2006
2. B. Adida, Helios: web-based open-audit voting, in *USENIX 17th Security Symposium (SS08)*, San Jose, CA, USA (2008), pp. 335–348

3. S.T. Ali, J. Murray, An overview of end-to-end verifiable voting systems. Computing Research Repository (CoRR) in arXiv (2016). https://arxiv.org/abs/1605.08554. Accessed on 11 Oct 2021
4. F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C.A. Reuter, M. Strand, A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192 (2015). https://eprint.iacr.org/2015/1192. Accessed on 11 Oct 2021
5. J.P. Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*. (No Starch Press, 2017)
6. J. Benaloh, M. Bernhard, J.A. Halderman, R.L. Rivest, P.Y.A. Ryan, P.B. Stark, V. Teague, P.L. Vora, D.S. Wallach, Public evidence from secret ballots. Computing Research Repository (CoRR) in arXiv (2017). https://arxiv.org/abs/1707.08619. Accessed on 11 Oct 2021
7. J. Benaloh, D. Tuinstra, Receipt-free secret-ballot elections, in *26th Annual ACM Symposium on Theory of Computing (STOC '94)*, Montreal, Quebec, Canada (1994), pp. 544–553
8. D. Bernhard, O. Pereira, B. Warinschi, How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to helios, in *Advances in Cryptology (ASIACRYPT 2012)*, LNCS, vol. 7658, eds. by X. Wang, K. Sako (Springer, Beijing, China, 2012), pp. 626–643.
9. D. Bernhard, B. Warinschi, Cryptographic voting – a gentle introduction, in *International School on Foundations of Security Analysis and Design VII (FOSAD 2012/2013 Tutorial Lectures)*, LNCS, vol. 8604 (Springer, 2014), pp. 167–211
10. G.R. Blakley, G. Kabatiansky, Shamir's threshold scheme, in *Encyclopedia of Cryptography and Security*, eds. by H.C.A. van Tilborg, S. Jajodia (Springer US, Boston, MA, 2011), pp. 1193–1194
11. D. Boneh, The decision Diffie-Hellman problem, in *3rd International Symposiun on Algorithmic Number Theory (ANTS-III)*, LNCS, vol. 1423, Portland, Oregon, USA (Springer, 1998), pp. 48–63. https://doi.org/10.1007/BFb0054851. Accessed on 11 Oct 2021
12. P. Bulens, D. Giry, O. Pereira, Running mixnet-based elections with helios, in *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE'11)*, San Francisco, CA, USA (2011)
13. S.G. Castelló, Individual verifiability in electronic voting. Ph.D. thesis, Universitat Politècnica de Catalunya, 2016. https://upcommons.upc.edu/handle/2117/96245. Accessed on 11 Oct 2021
14. P. Chaidos, V. Cortier, G. Fuchsbauer, D. Galindo, BeleniosRF: a non-interactive receipt-free electronic voting scheme, in *ACM Conference on Computer and Communications Security (CCS'16)*, Vienna, Austria (2016), pp. 1614–1625
15. M. Chaieb, S. Yousfi, P. Lafourcade, R. Robbana, Verify-your-vote: a verifiable blockchain-based online voting protocol, in *15th European Mediterranean and Middle Eastern Conference on Information Systems (EMCIS 2018)*, LNBIP, vol. 341, Limassol, Cyprus (Springer, 2018), pp. 1–14
16. D. Chaum, Blind signatures for untraceable payments, in *Advances in Cryptology (CRYPTO '82)*, eds. by D. Chaum, R.L. Rivest, A.T. Sherman, Santa Barbara, California, USA (1983), pp. 199–203
17. D.L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2) (1981)
18. B. Chevallier-Mames, P.A. Fouque, D. Pointcheval, J. Stern, J. Traoré, On some incompatible properties of voting schemes, in *Towards Trustworthy Elections: New Directions in Electronic Voting*, LNCS, Vol. 6000, eds. by D. Chaum, M. Jakobsson, R.L. Rivest, P.Y.A. Ryan, J. Benaloh, M. Kutylowski, B. Adida (Springer, Berlin/Heidelberg, 2010), pp. 191–199
19. M.R. Clarkson, S. Chong, A.C. Myers, Civitas: toward a secure voting system, in *2008 IEEE Symposium on Security and Privacy (SP'08)*, Oakland, CA, USA (2008), pp. 354–368
20. J.D. Cohen, M.J. Fischer, A robust and verifiable cryptographically secure election scheme, in *26th Annual Symposium on Foundations of Computer Science (SFCS'85)* (IEEE Computer Society, Portland, OR, USA, 1985), pp. 372–382
21. R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, in *17th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'97)*, LNCS, Vol. 1233, Konstanz, Germany (Springer, 1997), pp. 103–118

22. R.J. Dalton, The social transformation of trust in government. Int. Rev. Sociol. **15**(1), 133–154 (2005)
23. A. De Santis, S. Micali, G. Persiano, Non-interactive zero-knowledge proof systems, in *Advances in Cryptology: Proceedings of CRYPTO'87*, LNCS, vol. 293 ed. by C. Pomerance, Santa Barbara, CA, USA (Springer, 1988), pp. 52–72.
24. S. Delaune, S. Kremer, M. Ryan, Verifying privacy-type properties of electronic voting protocols: a taster, in *Towards Trustworthy Elections: New Directions in Electronic Voting*, eds. by D. Chaum, M. Jakobsson, R.L. Rivest, P.Y.A. Ryan, J. Benaloh, M. Kutylowski, B. Adida, LNCS, vol. 6000 (Springer, Berlin/Heidelberg, 2010), pp. 289–309
25. Die schweizerische Bundeskanzlei (BK): Verordnung der BK über die elektronische Stimmabgabe (VEleS) (vom 13. Dezember 2013 (Stand am 1. Juli 2018)) (2013). https://www.admin.ch/opc/de/classified-compilation/20132343/index.html. Accessed on 11 Oct 2021
26. M. Eck, Mixnets in a distributed ledger remote electronic voting system. Master's thesis, 2021
27. M. El Laz, B. Grégoire, T. Rezk, Security analysis of ElGamal implementations (2020), pp. 310–321. https://doi.org/10.5220/0009817103100321. Accessed on 11 Oct 2021
28. T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
29. A. Essex, J. Clark, U. Hengartner, Cobra: toward concurrent ballot authorization for internet voting, in *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/-WOTE 2012)*, Bellevue, WA, USA (2012), pp. 1–13
30. T.M. Fernandez-Carames, P. Fraga-Lamas, Towards post-quantum blockchain: a review on blockchain cryptography resistant to quantum computing attacks. IEEE Access **8**, 21091–21116 (2020)
31. C. Fontaine, F. Galand, A survey of homomorphic encryption for nonspecialists. EURASIP J. Inf. Secur. **013801**, 1–10 (2007)
32. A. Fujioka, T. Okamoto, K. Ohta, A practical secret voting scheme for large scale elections, in *Advances in Cryptology (AUSCRYPT'92)*, LNCS, vol. 718, Gold Goast, QLD, Australia (Springer, 1992), pp. 244–251
33. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)
34. K. Gjøsteen, The Norwegian internet voting protocol, in *Third International Conference on E-Voting and Identity (VoteID '11)*, LNCS, vol. 7187, Talinn, Estonia (Springer, 2012), pp. 1–18
35. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems, in *17th Annual ACM Symposium on Theory of Computing (STOC'85)*, Providence, Rhode Island, USA (1985), pp. 291–304
36. R. Haenni, E. Dubuis, U. Ultes-Nitsche, Research on e-voting technologies: a survey (2008)
37. R. Haenni, R.E. Koenig, P. Locher, E. Dubuis, CHVote protocol specification (Version 3.2). Cryptology ePrint Archive, Report 2017/325 (2017). https://eprint.iacr.org/2017/325. Accessed on 11 Oct 2021
38. T. Haines, M. Johannes, How not to VoteAgAin: pitfalls of scalable coercion-resistant e-voting. Cryptology ePrint Archive, Report 2020/1406 (2020). https://eprint.iacr.org/2020/1406. Accessed on 11 Oct 2021
39. T. Haines, S.J. Lewis, O. Pereira, V. Teague, How not to prove your election outcome, in *2020 IEEE Symposium on Security and Privacy (SP'20)*, San Francisco, CA, USA (2020), pp. 644–660
40. T. Haines, J. Muller, SoK: techniques for verifiable mix nets, in *33rd IEEE Computer Security Foundations Symposium (CSF'20)*. Virtual Conference (2020), pp. 49–64
41. J.A. Halderman, V. Teague, The new South Wales iVote system: security failures and verification flaws in a live online election, in *5th International Conference on Electronic Voting and Identity (VoteID'15)*, vol. 9269 (2015), pp. 35–53
42. F. Hao, P.Y.A. Ryan, *Real-World Electronic Voting: Design, Analysis and Deployment* (2016). CRC Press. https://www.amazon.com/Real-World-Electronic-Voting-Analysis-Deployment/dp/1498714692

43. S. Heiberg, T. Martens, P. Vinkel, J. Willemson, Improving the verifiability of the estonian internet voting scheme, in *International Joint Conference on Electronic Voting (E-Vote-ID 2016)*, LNCS, vol. 10141 eds. by R. Krimmer, M. Volkamer, J. Barrat, J. Benaloh, N. Goodman, P.Y.A. Ryan, V. Teague (Springer, 2017), pp. 92–107
44. H. Jonker, S. Mauw, J. Pang, Privacy and verifiability in voting systems: methods, developments and trends. Comput. Sci. Rev. **10**, 1–30 (2013)
45. H. Jonker, J. Pang, Bulletin boards in voting systems: modelling and measuring privacy, in *6th International Conference on Availability, Reliability and Security (ARES 2011)* (2011), pp. 294–300
46. A. Juels, D. Catalano, M. Jakobsson, Coercion-resistant electronic elections, in *ACM Workshop on Privacy in the Electronic Society (WPES'05)*, Alexandria, VA, USA (2005), pp. 61–70
47. A. Kiayias, M. Yung, Self-tallying elections and perfect ballot secrecy, in *International Workshop on Public Key Cryptography (PKC 2002)*, LNCS, vol. 2274 (Springer, 2002), pp. 141–158
48. C. Killer, M. Knecht, C. Müller, B. Rodrigues, E.J. Scheid, M. Franco, B. Stiller, Æternum: a decentralized voting system with unconditional privacy, in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC 2021)*. (IEEE, Darlinghurst, Australia, 2021), pp. 1–8
49. C. Killer, B. Rodrigues, R. Matile, E. Scheid, B. Stiller, Design and implementation of cast-as-intended verifiability for a blockchain-based voting system, in *35th Annual ACM Symposium on Applied Computing (SAC'20)*, Brno, Czech Republic (2020), pp. 286–293
50. C. Killer, B. Rodrigues, E.J. Scheid, M. Franco, M. Eck, N. Zaugg, A. Scheitlin, B. Stiller, Provotum: a blockchain-based and end-to-end verifiable remote electronic voting system, in *IEEE 45th Conference on Local Computer Networks (LCN)* (IEEE, Sydney, Australia, 2020)
51. C. Killer, B. Stiller, The Swiss postal voting process and its system and security analysis, in *International Joint Conference on Electronic Voting (E-Vote-ID 2019)*. LNCS, vol. 11759, Bregenz, Austria (Springer, 2019), pp. 134–149
52. C. Killer, L. Thorbecke, B. Rodrigues, E. Scheid, M. Franco, B. Stiller, Proverum: a hybrid public verifiability and decentralized identity management. Computing Research Repository (CoRR) in arXiv (2020). https://arxiv.org/abs/2008.09841. Accessed on 11 Oct 2021
53. S. Kremer, M. Ryan, B. Smyth, Election verifiability in electronic voting protocols, in *ESORICS. Lecture Notes in Computer Science*, vol. 6345 (Springer, 2010), pp. 389–404
54. R. Krimmer, A structure for new voting technologies: what they are, how they are used and why, in *The Art of Structuring* (2019), pp. 421–426. https://doi.org/10.1007/978-3-030-06234-7_39. Accessed on 11 Oct 2021
55. R. Küsters, J. Müller, Cryptographic security analysis of e-voting systems: achievements, misconceptions, and limitations, in *International Joint Conference on Electronic Voting (E-Vote-ID 2017)*, LNCS, vol. 10615 (Springer, 2017), pp. 21–41
56. P. Locher, Unconditional privacy in remote electronic voting theory and practice. Ph.D. thesis, University of Fribourg, 2016
57. P. Locher, R. Haenni, Verifiable internet elections with everlasting privacy and minimal trust, in *International Conference on E-Voting and Identity (Vote-ID 2015)*, LNCS, vol. 9269 (Springer, 2015), pp. 74–91
58. S.Z. Lowry, P.L. Vora, Desirable properties of voting systems. NIST E2E Workshop (2009). https://www.nist.gov/publications/desirable-properties-voting-systems. Accessed on 11 Oct 2021
59. P. McCorry, S.F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in *International Conference on Financial Cryptography and Data Security (FC 2017)*, LNCS, vol. 10322 (Springer, 2017), pp. 357–375
60. T. Moran, M. Naor, Receipt-free Universally-verifiable Voting with everlasting privacy, in *Advances in Cryptology (CRYPTO'06)*, LNCS, vol. 4117 (Springer, 2006), pp. 373–392
61. T. Moran, M. Naor, Split-ballot voting: everlasting privacy with distributed trust. ACM Trans. Privacy Secur. **13**(2), 246–255 (2007)
62. National Institute of Standards and Technology (NIST): Post-Quantum Cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography. Accessed on 11 Oct 2021

63. nVotes: nVotes – Secure Online Voting Software. https://nvotes.com. Accessed on 11 Oct 2021
64. P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *Advances in Cryptology (EUROCRYPT'99)*, LNCS, vol. 1592, ed. by J. Stern, Prague, Czech Repbulic (Springer, 1999), pp. 223–238
65. C. Park, K. Itoh, K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, in *Advances in Cryptology – (EUROCRYPT '93)*, LNCS, vol. 765, Lofthus, Norway (Springer, 1993), pp. 248–259
66. S. Park, M. Specter, N. Narula, R.L. Rivest, Going from bad to worse: from internet voting to blockchain voting. J. Cybersecur. **7**(1), (2021)
67. J. Puigalli, S. Guasch, Privacy and anonymity management in electronic voting. Identity Priv. Manag. **XI**(1), 59–95 (2010)
68. M. Richards, N. Ford, *Fundamentals of Software Architecture: An Engineering Approach* (O'Reilly Media, Inc., 2020)
69. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
70. R.L. Rivest, On the notion of 'software independence' in voting systems. Philos. Trans. R. Soc. Math. Phys. Eng. Sci. **366**(1881), 3759–3767 (2008)
71. P. Rogaway, The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162 (2015). https://eprint.iacr.org/2015/1162.pdf. Accessed on 11 Oct 2021
72. M.D. Ryan, B. Smyth, Applied Pi Calculus. Cryptol. Inf. Secur. Ser. **5**(2011), 112–142 (2011)
73. P.Y. Ryan, P.B. Rønne, V. Iovino, *Selene: Voting with Transparent Verifiability and Coercion-Mitigation*. LNCS, vol. 9604, Christ Church, Barbados (Springer, 2016), pp. 176–192
74. K. Sako, J. Kilian, Receipt-free mix-type voting scheme, in *Advances in Cryptology (EUROCRYPT'95)*. LNCS, vol. 921, Saint-Malo, France (Springer, 1995), pp. 393–403
75. F.B. Schneider, Open source in security: visiting the bizarre, in *IEEE Symposium on Security and Privacy (SP)*. Berkeley, CA, USA (2000), pp. 126–127
76. U. Serdült, M. Germann, F. Mendez, A. Portenier, C. Wellig, Fifteen years of internet voting in Switzerland: history, governance and use, in *2015 Second International Conference on eDemocracy & eGovernment (ICEDEG'15)*, Quito, Ecuador (2015), pp. 126–132
77. O.M. Sheyner, Scenario graphs and attack graphs. Ph.D. thesis, Carnegie Mellon University, 2004
78. N.P. Smart, *Cryptography Made Simple* (Springer Publishing Company, 2015)
79. R. Staubli, Secure deployment and configuration management for a decentralized remote electronic voting system. Master's thesis, 2021
80. C.J. Tolbert, K. Mossberger, The effects of e-government on trust and confidence in government. Publ. Adm. Rev. **66**(3), 354–369 (2006)
81. S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, Internet X. 509 Public Key Infrastructure (PKI) Proxy Certificate Profile. Technical report, RFC 3820 (Proposed Standard) (2004)
82. United Nations (UN) General Assembly: Universal Declaration of Human Rights (1948)
83. P. Vinkel, Internet voting in Estonia, in *Information Security Technology for Applications (NordSec 2011)* (2012), pp. 4–12
84. B. Witten, C. Landwehr, M. Caloyannides, Does open source improve system security? IEEE Softw. **18**(5), 57–61 (2001)
85. Y. Xiao, N. Zhang, W. Lou, Y.T. Hou, A survey of distributed consensus protocols for blockchain networks. Computing Research Repository (CoRR) in arXiv (2019). http://arxiv.org/abs/1904.04098. Accessed on 11 Oct 2021

Blockchain Technology Accelerating Industry 4.0

Jan Pennekamp, Lennart Bader, Eric Wagner, Jens Hiller, Roman Matzutt, and Klaus Wehrle

1 Introduction

The advances of the Internet of Things (IoT) and Industrial IoT (IIoT) fostered the development of cheap, long-living, potentially mobile sensors. These sensors lay the foundation for a pervasive gathering of data on workpieces, monitoring of production processes, and tracking of supply chains [107]. More specifically, the application of IIoT functionality, to make sensor data and aggregated information available to stakeholders in the Industry 4.0, promises improved product qualities [97], reduced overall costs [54], sustainable production [99], adjusted maintenance periods [95], optimized production planning [39], or more efficient resource usage [19].

However, making full use of sensor data often requires various stakeholders to combine their data and form an *Internet of Production* (IoP) [18, 94]. Thereby, companies may have to exchange data with yet unknown – and thus untrusted – collaborators, customers, or even direct competitors. Traditional IoT and IIoT

Jan Pennekamp, Lennart Bader, and Eric Wagner contributed equally to the work.

J. Pennekamp (✉) · J. Hiller · R. Matzutt · K. Wehrle
RWTH Aachen University, Aachen, Germany
e-mail: pennekamp@comsys.rwth-aachen.de; hiller@comsys.rwth-aachen.de; matzutt@comsys.rwth-aachen.de; wehrle@comsys.rwth-aachen.de

L. Bader
Fraunhofer FKIE, Wachtberg, Germany
e-mail: lennart.bader@fkie.fraunhofer.de

E. Wagner
Fraunhofer FKIE & RWTH Aachen University, Wachtberg, Germany
e-mail: eric.wagner@fkie.fraunhofer.de

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_17

security mostly focus on encryption and authentication to protect data in transit [57, 58, 60, 61, 120] or realize secure offloading to cloud resources [22, 34, 109]. When sharing data among mutually distrusting stakeholders, additional demands such as *verifiable correctness* and *immutability* of stored data emerge. Blockchain technology promises to provide corresponding security features for IIoT data handling in various industry domains and, thus, Industry 4.0. Specifically, the immutability properties of blockchains offer verifiability and accountability even in the face of mutually distrustful stakeholders. Furthermore, blockchain technology can build and represent a commonly agreed consensus among these otherwise mutually distrustful and conservative stakeholders.

Moreover, blockchain technology can be used in various configurations to adapt to the different industrial use cases. A common expectation is that the sharing of raw data or already analyzed and aggregated information will foster the move to more *dynamic* (business) relationships – even in competitive environments [93] – as trustworthy data can replace or bootstrap not yet existing trust in new collaborators. The accompanying benefits, such as better utilization of machine capacity, reduced costs, and flexible re-assignment in case of machine failures or transportation delays, mainly materialize *along* a supply chain (cf. Supply Chain Management Using Blockchain). Further improvements can be achieved if information is also shared *across* supply chains [95]. For example, information on machine configurations, machine failure rates, or influence of environmental settings on production quality can greatly improve productivity for all users of a machine type globally. However, especially this scenario requires sharing of information between companies without any existing (trust) relationships [99]. This data sharing can have severe consequences if business secrets are leaked or misbehaving entities share falsified information, e.g., machine settings that reduce productivity or even cause damage to production lines or humans [54].

Blockchain technology provides various options to tailor its features to specific industry needs. For example, the uniqueness of a product can be modeled with blockchain technology to prevent double-spending. Similarly, blockchains can also support the use of pseudonyms for participants to hide their identities. Furthermore, different blockchain types provide a varying level of openness (i.e., public vs. private blockchain), trust (i.e., permissionless vs. permissioned blockchain), scalability (from massive amounts of raw data up to highly compressed aggregated information), and additional functionality (e.g., smart contracts). Consequently, blockchain technology can dynamically adapt to various industrial scenarios and their needs.

However, while the foundations of blockchain technology are well understood, research to integrate it into different industrial use cases is still in its infancy as novel – both challenging and interesting – use cases are still being proposed, researched, and implemented. Some initial approaches for industrial collaboration and data sharing surfaced recently but are mostly realized without blockchain technology, e.g., a mechanism for the exchange of production process parameters [98] or a remote knowledge system for a privacy-preserving data exchange [25]. These approaches lack the industry-demanded accountability and verifiability features or fail to incorporate different stakeholders altogether. Nevertheless, these initial

approaches pave the way for more sophisticated realizations by underlining the unrealized potential.

We expect that the (full) utilization of blockchain benefits in industry, and especially in settings with multiple stakeholders, will follow a three-step pattern [95, 99] with an additional preparative step: **(0)** As a preliminary and preparatory step, expertise concerning blockchain technology must be established in the respective (industrial) domain, and decision-makers, engineers, and technologists need to be educated in the potential benefits of blockchain technology as well as in the handling of blockchain-supported technology and machines [2, 20]. **(1)** Afterward, existing use cases need to be remodeled and overhauled, improving existing (imperfect) approaches by integrating blockchain technology. **(2)** Following, these use cases have to be extended to also integrate market players without established trust relationships, superseding traditional trust with verifiable, trustworthy data. This step demands that the foundations of the novel technology are well understood and field-tested. **(3)** Finally, various new use cases will emerge and cover both trusted and untrusted entities by linking and connecting them using blockchain technology to unlock new potentials. Along this line, we present envisioned benefits achieved by the application of blockchain technology to Industry 4.0. However, we also shed light on the challenges when integrating this technology into the industrial domain and provide an overview of available configuration and design options to tackle these challenges. Furthermore, we outline the use of blockchains in early-adopting industrial domains.

Organization This chapter is structured as follows. In Sect. 2, we give a broad overview of the potential of blockchain technology for Industry 4.0 by highlighting its envisioned benefits for industry in general. Afterward, in Sect. 3, we examine challenges and corresponding design options for integrating blockchain technology in Industry 4.0 use cases. Then, in Sect. 4, we identify domain-specific needs and potentials by surveying approaches across a variety of industrial sectors. Based on these analyses, we discuss our findings in Sect. 5 and conclude the chapter in Sect. 6.

2 Envisioned Benefits of Blockchain Technology in the Industry

The unique properties of blockchains, especially providing transparency, verifiability, and accountability in decentralized settings, i.e., where no central entity can take on a leading role, render their use attractive for various business and industrial use cases. Stakeholders hope for simplified processes and relationships [117], cost reductions [117], improved customer satisfaction and reputation [76], and greater independence from third parties [117]. In this section, we provide an overview of the benefits of blockchain technology for industries. As we will detail in Sects. 4 and 5, some of these benefits are of primary interest for specific industries, whereas others are relevant for many industrial use cases.

2.1 Increased Transparency (⊞I)

Blockchain technology promises to provide transparency for currently opaque processes. Bitcoin [85] – the first blockchain application – promotes the vision to make all information publicly available and enables access by everyone. This paradigm seems to stand in stark contrast to typical goals of the industrial domain, where full transparency is often infeasible. Specifically, businesses risk their competitive advantage when providing insight into business secrets, processes, and collaborations; public access to such data could further hinder competition and even stall innovation. Consequently, stakeholders often lean towards keeping as much information secret as possible [43]. Nevertheless, as we explore next, industries can benefit from increased transparency by integrating blockchain technology into their processes.

2.1.1 Business-to-Business Transparency

Sharing information about (i) the origin of products, (ii) production details, and (iii) product destinations along the supply chain can yield benefits for businesses [97]. Specifically, businesses with access to public and immutable assurances of production steps along the supply chain can guarantee product properties without becoming liable for false claims by predecessors in the supply chain. In case of disputes, the liable entity can be unambiguously identified by all involved entities.

Despite these benefits, businesses often cannot afford to share the required detailed information about their production steps. Nevertheless, already the sharing of partial information, e.g., the product's origin, can result in advantages for the involved entities. More and more businesses desire to track the origin and trade intermediates of base products for legal, ethical, or marketing reasons. This information enables businesses to ensure the sustainable and legal sourcing of raw material [33, 65] and to unveil criminal associations or corrupt entities among their suppliers [112]. Likewise, the immutability and transparency properties of blockchains provide up-to-date tracing data and information on past actions, which can increase the credibility of businesses [65]. For example, access to trustworthy immutable data enables businesses to detect conflicting information and tempered claims early on [15, 74].

Considering legal frameworks, public access to business processes has the potential to realize a fair market access for each competitor by ensuring that every entity is abiding by the law. To this end, the ability to follow processes makes it harder to evade detection when breaking laws. Specifically, public records enable law enforcement, regulatory authorities, and competitors to check the compliance of business actions with the laws. Notably, digitalization would also allow businesses themselves to review and audit their actions to prevent misbehavior in the first place.

2.1.2 Business-to-Customer Transparency

Consumers become concerned about the ethics and sustainability of products, and consequently, they demand assurance that what they buy does not negatively impact others or the environment [28, 86, 121]. Research even repeatedly showed that the consumers' likelihood of purchasing "greener" products depends on how confident they are that a product is really green [48, 92]. Similarly, customers and companies are concerned about buying products that support terrorists, criminal associations, or corrupt entities [112]. Using blockchains to establish transparency on how the material is sourced, processed, sold, and maintained helps customers to select suitable products. For example, transparent sourcing promises to prevent blood diamond trades [36] or illegal tuna fishing [118].

Furthermore, customers benefit from access to historical price information, (business) behavior, sustainability, and contract conditions. They can then base purchase decisions on, e.g., worker conditions or fair trading agreements. Businesses that can reliably and credibly provide corresponding data through blockchain technology become more attractive to customers, which boosts the businesses' revenue and reputation [65]. Thereby, blockchains enable customers to choose based on unbiased data of long historical periods and without relying on a trusted third party.

Transparent processes also enhance business relationships that comprise a direct feedback loop between customers and company, e.g., continuous maintenance or insurances, leading to faster and cheaper interactions for both entities [10]. Moreover, this transparency can also lead to quality-improved or longer-lasting products, as the enhanced knowledge of products and their usage enables, e.g., to schedule maintenance intervals on time or to adapt products to specific needs [95].

2.1.3 Customer-to-Customer Transparency

The discussed transparency benefits are not limited to business relationships but also apply to interaction between customers. For example, direct reselling between customers is a market that significantly profits from information on the product's manufacturing and maintenance history, enabling buyers to infer more about the product's state before purchase. Access to this data could be realized via blockchain-based data markets, which also provide best practices for usage or settings considering productivity, low operating costs, or sustainability of products. As we focus on industrial use cases, we do not further discuss such customer-related benefits.

Concluding, establishing transparency with blockchain technology has several benefits. We note that, to realize these benefits, all involved entities (businesses and customers alike) must be accountable for the available data.

2.2 *Decentralized Accountability (⊞II)*

Orthogonal to transparency (⊞I), blockchain technology increases accountability (⊞II) without requiring a central entity. By transparently recording information on an immutable distributed ledger, businesses obtain a reliable proof of business actions, relationships, and contracts. As discussed before, stakeholders are able to set up control mechanisms and identify misbehaving stakeholders. To this end, cryptographic signatures publicly link information on the blockchain to the providing party. Consequently, blockchain technology increases the trust between businesses.

To not jeopardize this trust, a key requirement is the resistance against collusion attacks. Especially in comparison to centralized platforms, where the deletion or even manipulation of information might only require a collusion of a few stakeholders, the decentralized approach of blockchains achieves a significantly higher degree of collusion resistance. In particular, already a simple majority of honest and well-behaving blockchain nodes [75] prevents data loss, manipulation, or deletion.

An open challenge, however, is to realize the insertion of correct information in the blockchain [96, 124]. Here, technical means are needed to ensure that the processed data is reliable and has not been tampered with before. Otherwise, all beneficial features of blockchain technology are rendered useless.

In the following, we specifically discuss the advantages of accountability for product data, business contracts, and behavior policies that promise to foster trust-based relationships between multiple businesses or towards their customers.

2.2.1 Data Accountability

Customers and businesses can profit from increased transparency and open, accessible information (⊞I). However, the value of the shared information and the achieved transparency heavily depend on the originality, correctness, and completeness of the provided data. Consequently, only immutable and cryptographically proved information as, e.g., provided by blockchain technology ensures the desired accountability. These properties thus strengthen the benefits of information transparency and corresponding enhanced business opportunities collaboration (⊞III) [75, 97].

Furthermore, the decentralized handling of on-chain information obviates the need for a trusted third party. In combination with cryptographic proofs of origin, which link data to the providing party, businesses can be more confident about the correctness of data, enabling conflict resolution in case of disputes or misbehavior.

Moreover, holding businesses legally responsible for all submitted information or maintaining a publicly available reputation system [76] reduces financial risks while increasing the trust between involved businesses. A digital and reliable source of information further paves the way for more advanced approaches and their benefits, including sophisticated automation (⊞IV) and a fostering of digitalization (⊞V).

2.2.2 Legal Compliance

Another aspect that is closely related to accountability is legal compliance. Especially in competitive environments, making sure that all participants follow enacted laws is crucial to ensure a fair market. However, also from a governance (e.g., taxes), customer (e.g., warranty), and environmental perspective (e.g., sustainability), legal compliance is an important aspect. Here, blockchain technology offers different features to document and ease checking for legal compliance.

Most legislation requires businesses to document their actions and transactions. Blockchain technology offers a simple, digital way to securely conform with these requirements. This digital documentation can entail additional benefits, such as a complete and correct, but also simpler and more transparent, taxation [23]. Likewise, transparently recorded money flows could help uncover corruption [42] or identify illegal practices, such as sourcing material from sanctioned countries [23].

Another benefit for the legal domain is an increased digitalization (⊞V) which evolves from the pre-requirement to use digitized data to utilize blockchain technology. In the legal domain, this digitalization reduces barriers for the establishment of new business relationships. Specifically, maintaining a history on chain eases the negotiation of contracts as all relevant information is readily available. Thereby, lengthy checks for legal compliance are significantly shortened such that collaborations can be established faster and more easily. Thus, blockchains could simplify audits, reduce administrative costs, and speed up processes, e.g., by automating large parts of otherwise manual checks of paperwork [23]. In case of conflicts, contracts on chain represent a single undisputed view of the current situation. Consequently, the situation in legal proceedings is transparently documented and cannot be easily disputed, i.e., finding loopholes is more challenging due to publicness. These aspects not only hold for contracts between businesses but also towards customers, e.g., blockchain-based insurances increase policy transparency, enable automation, and ensure publicly verifiable legal compliance in case of claims (cf. Sect. 4.8).

2.2.3 Building Reputation

With accountability, companies can easily build up a well-documented reputation. This reputation can help to acquire more business partners or customers. Furthermore, new business relationships which yet lack trust can draw from publicly available information on past company behavior.

For e-commerce, blockchain technology could further realize a trustworthy reputation system. Similar ideas are known from the context of supply chains, and initial approaches realize such functionality using smart contracts [76, 122]. Likewise, the government can utilize blockchain technology to encourage the participation of both companies and citizens by publicly reporting the planning, execution, and reporting of processes in cities [37]. Some specific applications would be, e.g., to report and track waste or potholes. Overall, blockchain-based

reputation can support all stakeholders in a domain and thus allows for well-founded decisions and actions.

2.3 Collaboration (⊞III)

In addition to transparency (⊞I) and accountability (⊞II), blockchain technology can further facilitate the collaboration between stakeholders by providing a shared and distributed medium for the required exchange. In this section, we highlight the benefits for already established collaborations and ad hoc collaborations.

2.3.1 Benefits for Already Established Collaborations

The blockchain provides a unified interface for the provision of and access to trustworthy data shared between collaborators. Together with established trust levels in long-lasting collaborative relationships and a fine-granular access control realized with selective accessibility (details in Sect. 3.2), this unified data access enables novel applications. For instance, the unified and trustworthy interface facilitates the tracking and tracing of goods and products even over multiple hops of a supply chain [97]. Furthermore, a fine-grained exchange of detailed product quality information enables collaborators to adapt their machine settings when processing these products further, thereby avoiding low-quality results [97]. Finally, blockchain-based collaborations can improve Enterprise Resource Planning (ERP) across businesses [77].

2.3.2 Benefits for New or Ad hoc Collaborations

Some industries face the challenge of relying on relatively short-lived or spontaneous collaborations. Especially small businesses and startups need to remain flexible regarding new collaborations as they often depend on frequently acquiring new collaborators [35, 45, 100]. Furthermore, also more established industries with traditionally long-lived collaborations, such as manufacturing, are in the process of becoming more dynamic, e.g., through on-demand manufacturing in low quantities [31]. Both scenarios require stakeholders to spontaneously seek a collaboration partner and frequently engage in only short-lived collaborations based on their short-term needs. Such an environment makes it hard for stakeholders to establish trust, and they thus require other means to assess a potential collaborator's assets.

Industry-wide blockchain-based platforms can provide the data source for an assessment of the trustworthiness of potential new collaborators [100]. By making use case-relevant data accessible for potential collaborators, businesses can reduce reservations of potential collaborators to establish a new, not yet trustworthy,

business relationship. Thereby, the blockchain prevents those businesses from retrospectively altering past records, i.e., the transparency (⊞I) ensures a full view of past collaborations of the potential new partner. Furthermore, special blockchain designs such as *consortium* blockchains and mechanisms for fine-granular access control to blockchain data (which we detail in Sects. 3.1 and 3.2) can help to establish technical trust throughout the new or short-term business relationship.

2.4 Trustworthy Automation (⊞IV)

One key benefit of blockchain technology is the potential for higher degrees of automation. The concept of smart contracts fosters this association as they provide stakeholders with a framework for setting up mutually agreed rule sets that are automatically invoked based on new events being recorded on the blockchain. Especially using blockchain-extrinsic data sources, e.g., (I)IoT sensors [96], GPS data [41], or weather data [41], promises an increased potential for automating processes tied to the real world, between not fully trusting stakeholders. These potentials unfold over all industry-related interaction models, i.e., bilateral negotiation between individuals [119], customer-related processes such as processing insurance claims [10], business-to-business (B2B) interaction [49], and even fine-granular machine-to-machine (M2M) communication [53, 96].

In these scenarios, the blockchain provides a logically centralized place for the automated lookup of process-related data, providing an alternative to today's heterogeneous platforms for inter-business exchanges [74]. Hence, smart contracts could replace or augment traditional contracts and thereby simplify the processes between businesses due to a simplified drafting and interpretation of contracts [23]. Furthermore, the increased automation can ultimately increase both customer and employee satisfaction as well as the competitiveness, profits, and overall performance of businesses. For example, increased automation can result in *easier payment processing* as well as *efficient M2M communication*, which we briefly discuss in the following.

2.4.1 Easing Payment Processing

Cryptocurrencies established an effective way for private users to realize low-cost and near-instantaneous international money transfers [113]. Likewise, collaborating businesses from different countries could agree on relying on cryptocurrency B2B payments to seize the same advantages [74]. In this case, B2B money transfers can further be automated as discussed earlier and thereby enable new and more dynamic forms of collaboration, such as manufacturing-as-a-service [95] and, in general, properly billed cross-business M2M communication [53].

Since blockchain transactions are typically visible for the participating stakeholders, smaller suppliers, such as gemstone miners with lower extraction capaci-

ties [112], can further profit from increased price transparency [112]. Finally, tiny stakeholders, such as household energy prosumers, can also benefit from reliable micropayments and transparency provided by blockchain technology [62, 82, 104].

2.4.2 Machine-to-Machine Communication

Referencing real-world data on the blockchain opens up new potentials for efficient M2M communication. By sensing data with trusted hardware, machines owned by different stakeholders can exchange information while mitigating the risks of data manipulation by a (partially) distrusted collaborator [96]. Furthermore, machines can use automated micropayments and potentially provide revenue in real time (cf. Sect. 2.4.1). Customer-oriented M2M micropayments are already used, e.g., to automate payments at gas stations [53]. Again, these payments build on technically enforced rules and can thus occur on the fly without further negotiation. In the future, other industries can seize these benefits, e.g., to fully realize the aforementioned manufacturing-as-a-service paradigm where smart production lines can directly bill any short-term customer based on a fine-grained tracking of used machine hours.

2.5 Blockchains Foster Digitalization (⊞ V)

The benefits provided by blockchain technology can only be achieved with a thorough digitalization of data and information in corresponding uses cases such that it can be provided to the blockchain. Many forms of digitalization would have been possible without blockchain technology, e.g., using simple digital signatures. However, we posit that the additional benefits of blockchain technology provide the required strong incentives to finally make digitalization happen in many domains. For example, as briefly discussed in Sect. 2.2.2, the use of blockchain-based smart contracts fosters digitalization in the legal domain. In contrast to traditional paperwork, digitalized data on blockchains is virtually always accessible, is properly indexed, and supports fast automatic (keyword) searches. In the following, we showcase the potential of blockchain-backed data management to *supersede intermediates*; discuss how *digital certifications* can provide fast and trustworthy attestations about products, businesses, and even employees; and show to which extent blockchains can facilitate the negotiation and enforcement of *contracts* while reducing the required paperwork.

2.5.1 Disintermediation Through Blockchain-Backed Data Management

Digital paperwork allows for timely and automated access to relevant documents due to instant availability and fast searches based on digital indexes. As we discussed for blockchain-based collaboration (⊞ III), maintaining such a document

index on the blockchain also allows for a fast and automated exchange of documents between businesses. Furthermore, the blockchain's immutability ensures the authenticity of requested documents. Using a blockchain with selective accessibility (cf. Sect. 3.2) additionally enables businesses to use corresponding benefits in a non-public setting which further facilitates document-dependent business processes, such as reporting to collaborators or the exchange between businesses' legal departments, by reducing human interaction to managing access control. This vast reduction of required interaction along with a smaller chance for errors enables a form of *disintermediation*, i.e., avoiding tasking trusted third parties with organizing required document exchanges [88]. Corresponding benefits have been seized in multiple proposals already, e.g., in the contexts of shipping containers [52], diamonds [36], or insurances [10].

2.5.2 Expressiveness of Digital Certification

Digital certification can help businesses to establish a reputation, e.g., by getting certificates for products, product properties, or production processes [65] while reducing the organizational overhead and paperwork associated with the certification processes [102]. Furthermore, employees can generate electronic portfolios based on digital (micro) accreditation, e.g., in the form of Open Badges [3], which attest their skills and experiences. Blockchain-based platforms have naturally been identified as a candidate for hosting such digital certifications [47]. Linking such a certificate immutably to a blockchain underpins the certificates' trustworthiness as multiple independent parties can acknowledge its authenticity [46]. In cases of a fully digitized certification process, e.g., in the context of e-learning, a complete automation of the process, including an immutably stored certificate on the blockchain, is conceivable. Furthermore, if businesses also link all data required to become certified to the blockchain in a selectively accessible way, corresponding audit processes can be digitized as well and thereby reduce costs and resources blocked by ongoing audits. Finally, blockchain-based certifications can be extended further without risks of fraud to, e.g., certifying the sustainable processing goods along a whole supply chain.

Thereby, digital certification can be seized in different industrial applications. On the one hand, in addition to the origin of goods, e.g., gemstones and fish, environmentally responsible businesses can receive an attestation of their sustainable sourcing of goods and other materials [33]. On the other hand, unified and immutable certification via industry-widely used blockchain-based platforms helps to disambiguate conflicting claims. For instance, recording transactions of assets, such as land ownership, on a blockchain makes it easier to decide conflicts, e.g., stemming from conflicting ownership records, and prevent further fraud [15, 74].

2.5.3 Blockchain-Backed Contracts

Beyond the automation of inter-business processes through smart contracts (⊞IV), blockchain technology can also improve paper-based contracts. By disclosing their expertise on a blockchain, e.g., in the form of an audited digital certification, businesses can easily identify potential collaborators (⊞III) using the blockchain as a centralized repository of business capabilities and offered services. This information can serve as a foundation for new contracts, as responsibilities can be expressed in reference to the (claimed or attested) capabilities.

Furthermore, a blockchain can also hold terms and conditions specified by one business partner and immutably record the collaborators' dedication to abide to these terms. This approach is especially beneficial when handing over intellectual properties. For instance, creators of educational material may sell their content over a blockchain-based market that records license terms [80]. Comparable approaches can be transferred to other, especially data-centric, industrial applications.

3 Challenges and Design Spaces for Industrial Blockchains

Initial use cases for blockchains focused on public services such as digital currencies [85], general-purpose frameworks for smart contracts [123], or digital notary services for existence proofs of data [14]. Contrary to these public services, realizing the blockchain benefits for industrial applications and the Industry 4.0 discussed in Sect. 2 requires addressing complex challenges regarding storage and accessibility of data. More specifically, industrial use cases often have tighter demands on (i) who controls the blockchain's infrastructure, (ii) access control for data stored on the blockchain, (iii) scalability considering growing infrastructures, numbers of users, and data volumes, and (iv) the efficiency of the blockchain, i.e., preventing the blockchain from becoming a performance bottleneck. In the following, we discuss design options that address these demands on blockchain-based systems and thus realize the envisioned benefits for Industry 4.0.

3.1 Control over Blockchain Infrastructure

Public blockchains – as used by initial deployments – provide relatively small control over who can insert which data and require expensive consensus mechanisms to protect against malicious actors. These restrictions are, however, only necessary if not all contributors are known beforehand [124], which is rarely the case in industrial scenarios. Still, industrial applications can seize the benefits of established public blockchains (i) as a decentralized trust anchor, (ii) for access to a widespread digital currency, and (iii) for a resource-less integration. In contrast to public and trustless systems, industrial stakeholders pushed *private blockchains*, which are

controlled by a single entity. While the applications of such blockchains are limited since all users must trust a single stakeholder, they are sufficient, e.g., if a parent company sets up a blockchain for its subsidiaries [56]. In exchange, they can rely on a much more efficient consensus mechanism for increased data throughput and reduced maintenance costs [83]. *Consortium blockchains* present a hybrid approach that distributes control among a group of known stakeholders to obviate the trust in a single entity while still supporting efficient consensus protocols. As consortium blockchains only require partial trust among entities, they are attractive, e.g., for supply chain management.

3.2 Access to Blockchain Data

Beyond the control over the blockchain infrastructure, industrial applications also often require tight access control for data stored on the blockchain. We distinguish three basic access levels: *public*, *stakeholder*, and *selective* accessibility. Public accessibility is, e.g., suitable to achieve high transparency regarding the ethical, sustainable, and regulation-abiding handling of goods [24, 50, 66, 87]. However, publicly disclosing production or other business-related data risks that data is exploited to infer sensitive information about customers, business partners, or production processes [124]. Thus, public data has to be carefully selected, and more restrictive data access models need to be applied for sensitive data. Stakeholder accessibility presents the contrary extreme where only the blockchain-maintaining nodes can access blockchain data. This model is suitable for internal accounting and tracking of events without (automated) access for outsiders. It furthermore often matches the demands of use cases with private or consortium blockchains, which typically serve internal processes. Finally, selective access to blockchain data can be achieved via fine-granular access control systems [97] that enable operators of private or consortium blockchains to selectively disclose blockchain data to outsiders, e.g., known customers or subcontractors, while keeping internal data secret. However, this flexible model burdens blockchain nodes with key management overheads.

3.3 Scalable Blockchain for Massive Amounts of Industrial Data

Industrial use cases likely challenge the scalability of blockchains with respect to the amount of stored or referenced data on the blockchain. For instance, industrial use cases that monitor production processes can produce massive amounts of data through large numbers of heterogeneous sensors with potentially high sampling frequencies. This data volume remains challenging concerning the throughputs of today's blockchain systems. Private or consortium blockchains can help to

meet scalability demands as they limit the number of participants and achieve larger throughputs (cf. Sect. 3.1). Still, data-intensive use cases such as monitored production need further options. Using *data aggregation* to only extract and store the relevant information from generated data on the blockchain can, e.g., reduce throughput requirements [96, 126]. Another option is to only store small identifiers of large data and cryptographically link them to a scalable *off-chain storage*, e.g., distributed hash tables [110, 130], the Interplanetary File System (IFPS) [89], or cloud storage [127]. Furthermore, obsolete data, e.g., information about drugs that exceeded their date of expiry [59], can be *pruned* from the blockchain to free valuable storage [79, 81].

3.4 Efficiency of Blockchain Operation

Finally, some industrial use cases depend on low latencies of blockchain writing processes. For instance, future power grids use a latency-dependent coordination (cf. Sect. 4.4), whereas traditional blockchains incur delays of several minutes. To overcome these delays, industrial use cases can realize low-latency data insertion via mainly two options: *efficient consensus mechanisms* or *off-chain computations*. Private or consortium blockchains can achieve a quick use of blockchain data through efficient consensus mechanisms, which can be further enhanced if participants trust the maintainers and do not wait until immutability is achieved through technical means. Furthermore, critical operations can be shifted off-chain, e.g., financial transactions during energy trading (cf. Sect. 4.4) can be processed through micropayments channels [21, 29], where the sender and receiver of digital currency repeatedly exchange valid transactions without sending them to the blockchain. Similarly, complex smart contracts can be executed off-chain if latencies or costs become prohibitive.

4 Use of Blockchain Technology in Industrial Domains

In this section, we provide an overview of blockchain integration by different and heterogeneous industries and the achieved benefits. Thereby, we consider the general benefits of the technology (cf. Sect. 2) and map them to real-world benefits for industries from different domains. To identify both common and industry-specific obstacles for seizing the benefits of blockchain technology, we cover industries from the primary, secondary, and tertiary sectors in our case studies.

4.1 Engineering and Manufacturing

One prominent industry that benefits from blockchain technology is the engineering and manufacturing industry. The reliability and quality of products are especially crucial when they constitute integral parts of larger systems with corresponding catastrophic consequences in case of failures. Sharing information along the supply chain can, e.g., increase product quality and reliability, reduce redundant work, establish more trustworthy product data, strengthen auditing processes, detect potential issues in an early production stage, and ease product recalls [11, 70, 75, 97, 117].

Notably, sharing this information with potentially unknown and untrusted businesses entails the risk of exposing critical business secrets, e.g., production volumes and quality [124] or collaborations [11, 97], and collaborators may intentionally share imprecise or wrong information [95]. The use of consortium blockchains, however, can mitigate these risks by realizing selective accessibility with permission control and accountability. Thus, blockchain technology can play a crucial role in enabling the sharing of authentic data along the supply chain [40]. More specifically, it realizes a unified view among participants via data consistency, increasing trust in the shared data, and preventing counterfeit products from entering supply chains or end customer markets [52]. Thereby, use cases still can use fine-grained access control and benefit from transparency, consistency, and immutability. Even more, the integration of data sharing into manufacturing processes facilitates auditors' and certifiers' work, enabling continuous certification processes instead of only periodic audits [117]. These processes can even completely forgo a trusted third party and instead rely on automatic and verifiable processes based on a blockchain [40].

Besides collaborating with respect to individual products, collaboration extends to sharing of domain knowledge [98], manufacturing equipment [117], and common long-term goals [117]. Digital machine data marketplaces aim to collaboratively optimize production processes by sharing data on unforeseen machine failures or production issues with business partners or even competitors [117]. Without blockchain, the reliability and availability of such information are limited as a verification of origin, integrity, and validity of the data is missing [117]. A blockchain can improve the verifiability of these aspects by transparently and immutably recording data acquisition metadata and providing the foundation for a decentralized data marketplace where information itself becomes a monetary good [99, 117]. Ultimately, sharing domain knowledge on such a decentralized data marketplace paves the way for establishing digital factories [99] and reduces costs for process optimization in the companies themselves [98, 117]. Thinking even further, sharing equipment and machines in a pay-per-use model using a decentralized blockchain-based application for payments and machine management promises both a better utilization and a highly flexible use of machine pool capacities [117]. Finally, blockchain technology can facilitate addressing collaborative goals such as optimizing a consortium's environmental footprint [117]. Transparently and collaboratively managing aspects such as recycling or carbon emissions with the help of on-chain

contracts in combination with financial benefits for "green" companies provides an additional motivation to reduce their environmental impact as they can prove their achievements to end-customers.

4.2 Lumber Industry

Besides the manufacturing industry, also industries concerned with raw materials production can benefit from blockchain technology. We focus on the lumber industry, where illegal timber harvesting causes estimated annual damages between USD 51 billion and USD 152 billion [125]. Preventing illegal logging protects biodiversity and forests, reduces carbon emissions, and enables sustainable forest management [125]. Also, customers increasingly care about *trustworthy* sustainability information [87]. However, current protection mechanisms against illegal logging, e.g., the *EU Timber Regulation*, lack effective auditing, and detected anomalies are not properly prosecuted: Paper-based certification approaches are opaque, inefficient, and easily tampered with [125]. Blockchain technology has been envisioned as a key building block to tackle corresponding challenges [33, 87].

While both customers and the overall industry share the desire for sustainable products [8], these demands typically affect only the customer-sided end of a supply chain and not the raw materials producers [87]. Consequently, even honest end-producers have to trust the paper trail of their suppliers and thus cannot abundantly guarantee the usage of sustainable and conflict-free raw materials. Blockchain technology promises more efficient, trustworthy, and holistic tracking of a product's origin and sustainability by making the data records immutable and publicly accessible. In contrast to paper-based certification, the division, merging, and replication of certificates of origin can be easily prevented through blockchain properties. Realizing a trustworthy certification process would also enable the import of timber from regions that are currently not considered for trade by many companies due to an elevated risk of breaching the law in western countries [33].

While the transparency provided by a blockchain eases fraud detection and thus makes it hard for individual stakeholders to systematically cheat within their supply chain, ensuring authentic data requires additional efforts. Here, proteomics can provide physical evidence of the correctness of blockchain data [33]. Furthermore, IoT sensors, GPS location trackers, RFID chips, bar codes, and point-of-sales data provide additional information to detect inconsistencies of blockchain data [87]. However, finding an optimal inconsistency detection method is still an open research problem. Furthermore, the underlying blockchain has to support multiple hundred parties, producing over 100,000 transactions per minute [33]. Nevertheless, even today, the transparency and data consistency offered by blockchain technology could make the lumber industry fairer, more sustainable, and more competitive, which makes its use attractive for end-consumers, producers, and even regulators.

4.3 Petroleum Industry

The petroleum industry primarily deals with the detection, extraction, refining, and transport of crude oil. This resource is an important component of many products, including fuel, plastic, and pharmaceuticals, and will thus remain important for the next decades. This industry is controlled by a few big and established players, mainly because crude oil extraction is profitable only at large scales and limited to a few regions of the world. Despite the industry's known resistance to change, established big players and new startups alike explore the integration of blockchain technology into their processes, showing its potential to benefit the industry [74].

Blockchain technology can even help before any crude oil extraction takes place. Companies spend significant amounts of money for managing land claims to extract oil. Here, a significant amount of conflicting claims frequently leads to disputes [74], and fraud in land transactions is rather common, especially in less developed regions [15]. In these regions, land registrations are often still managed on paper, and records are kept in a single location [15]. This situation leads to inefficient processes, inconsistent information, the potential of data loss, and manipulation due to the lack of redundancy and backups. Managing land claims via a blockchain-backed system would consequentially tackle these problems and provide additional benefits over a decentralized database: transparency and tamperproofness ensure a globally consistent state of land claims, digitized contracts enable automation to reduce costs and processing time, and public trades prevent corruption [15]. Yet, corresponding changes are hindered by a lack of expertise and political resistance to changes [15].

After extraction, crude oil usually crosses several borders and is traded multiple times on the path to its destination, involving numerous middlemen [106]. Blockchain technology can increase automation for these processes and establish transparency, which speeds up tax audits, reduces administrative costs, and tackles delays due to paperwork validation, e.g., by intermediate traders [23]. Blockchain-based processes could, e.g., tokenize individual barrels of oil or replace traditional with smart contracts that, e.g., automatically execute buy orders upon governmental approval [23, 74]. Overall, these improved processes would lead to fewer disputes (a significant problem in the industry [23]) and misunderstandings by eliminating ambiguities and reducing the amount of work to draft and interpret contracts while simultaneously reducing the necessary efforts to prepare for eventual conflicts [23, 74].

Another aspect of the petroleum industry that can be significantly improved upon by blockchain technology is payment processing, either by handling payments on chain or by sharing provably consistent information with all involved parties. Here, the petroleum industry can especially benefit as multiple parties are involved in a single trade, trades involve large sums of money, and trades happen while the ware is in transit [106]. First, post-transaction management, which currently occupies banks for several hours per trade due to employing special scrutiny, can be sped up by sharing information on a blockchain [106]. Second, the trusted

environment offered by a blockchain enables the entities in the petroleum industry to cut out intermediaries [74]. Finally, the transparency of blockchains eases fair and transparent taxation even when the traded oil is traveling across the world [23]. This fair taxation benefits governments and businesses alike, as it ensures that all trades are taxed correctly [23]. Furthermore, blockchain technology is beneficial in legal aspects. In particular, the transparency of blockchains enables easier and faster verification of abidance to ever-changing global sanctions and environmental laws [23], as well as helps in preventing and uncovering corruption [42, 74].

4.4 Energy Industry

The energy industry, i.e., businesses that manage continental, national, or regional power grids and feed energy into them, experiences extensive changes regarding the power grid's structure and the applied business models. Traditionally, power is produced in the "core," e.g., by large power plants, and traverses long distances via the transmission grid to local distribution grids, e.g., serving a smaller region consisting of several small towns. The increase in renewable energy plants, e.g., solar power plants and wind turbines, results in a distributed power production and more complex grid management, especially as local energy production might exceed or fall short of local demands, depending, e.g., on the weather. Further, individual households might be producers and consumers (prosumers) at the same time, e.g., houses equipped with photovoltaic arrays can flexibly import and export energy. Energy storage systems in private households further enable a fine-granular control *when* to buy or sell energy, further increasing the energy market's complexity [5]. *Local Energy Markets (LEMs)* even enable energy trading between prosumers [82] and local consumers to achieve a better cost-efficiency as if buying energy from the grid providers. Trading between multiple LEMs can further contribute to this cost-efficiency, but requires even more advanced synchronization and coordination [82].

Blockchain technology offers multiple benefits for such peer-to-peer environments by realizing a decentralized, yet accountable trading [108]. With the help of blockchain, LEMs can independently trade excessive energy in a decentralized manner [82]. Meanwhile, blockchain technology can also help coordinate energy consumption to stabilize the power grid [5]. Depending on the specific requirements, both consortium blockchains and public ledgers are applicable. Existing approaches already consider several aspects of this trade, such as digital currencies [82], on-chain contracts [84], privacy enhancements [38], and transparency of local trading [38], whereby price transparency enables fairer competition and pricing policies [51].

Another application of blockchain technology in energy-related industries deals with the internationally regulated trading of carbon emission licenses [4]. In contrast to LEMs, blockchain-based carbon emission applications aim to enable on-chain business-to-business contracts and trading. Companies that cannot meet their carbon emission targets can buy *emission credits* from those companies that have excess

credit [4]. Similar to LEMs, carbon emission trades benefit from decentralization [68] and wide-ranged (pricing) transparency to increase competitiveness and prevent extortion and monopolies. Besides decentralization, existing approaches such as the Bitcoin-based carbon emissions trading platform D-CETI [4] consider further aspects, covering privacy, dedicated currencies, and efficiency measures.

4.5 Food Supply

A field that is even more critical to humanity than the energy industry is that of food production and supply, which represents a further beneficiary of blockchain technology. There is an increasing demand of consumers and business partners to reliably determine the origin, ingredients, and quality of food products as well as to obtain knowledge about animal and worker treatment as well as the fulfillment of food label requirements [64]. As food supply chains comprise various actors with different incentives, ranging from farmers over producers to retailers, blockchain technology is seen as a major innovation driver for the industry's future [64, 69, 72, 111].

As a first benefit, information on production and transportation parameters can be reliably recorded on blockchains. Information, such as temperature readings in warehouses and trucks, aids the detection of interruptions in cold chains [96, 124]. While an accountable monitoring of products can increase customer satisfaction by transparently communicating this data, it can also reduce costs for producers, e.g., by reducing food waste [72, 128]. Two general concepts to share such production and transportation information are applicable. Firstly, the history of each product can be individually monitored with the help of QR codes, RFID chips [115], and sensors [124]. Alternatively, accountable information sharing for all or a random sample of products and involved businesses enables companies to provide reputation-based proofs of their reliability to partners and customers. In particular, a customer learns how involved businesses fulfill certain requirements based on past and recent data elicitations. While customers benefit from fine-grained information on individual products, such granularity is more challenging and costly to achieve.

Secondly, tracking individual products and their ingredients along the supply chain from the initial producer, e.g., the farmer, to the final customer, enables advanced quality assurance to, e.g., identify contaminated food [64, 111]. While blockchain technology would not add benefits in an ideal world where all involved businesses collaborate to identify affected products, contaminations lead to financial and reputational damage for the responsible businesses, such that they have an incentive to obfuscate incidents and hinder investigations. Hence, the accountability and transparency of blockchain technology are valuable to acquire reliable information regarding the production of individual products. Thus, the digital trace of products can aid in the reliable and fast identification of other contaminated product batches.

Thirdly, blockchains can aid to achieve accountability and transparency for food labels, e.g., organic and *Designation of Origin Collected and Guaranteed* (DOCG) food [6], kosher [32], halal [12], and vegan [32] food. Foremost, blockchain data prevents an inappropriate relabeling of food, e.g., labeling horse meat as beef [105]. Similarly, food certifications and seals such as fair trade-related labels [103] or the MSC fishing label [91] can benefit from blockchain-based transparency as customers get a more detailed insight into the production process. However, maintaining accountability along multiple hops of the supply chain remains challenging.

4.6 Logistics

Beyond the transportation of food, the whole logistics industry, which delivers wares to business partners, is a natural candidate for blockchain applications. The industry currently spends roughly USD 360 billion annually on the administration and documentation for moving goods across borders [106]. Blockchain technology offers one building block to reduce these costs by enabling the distributed tracking of shipping documents, e.g., customs forms and bills [106]. This distributed tracking reduces delays and costs compared to manual information sharing among all entities while preventing inconsistencies. Such a blockchain-backed process even enables the automatic initiation of follow-up steps, e.g., bill payments [30]. Additionally, such a process prevents the alteration, theft, or forgery of documents [52], decreasing legal and administrative costs for settling disputes.

Beyond easing paperwork, blockchain technology can also improve the planning of shipments. Blockchains offer a trusted platform for data sharing, such that all stakeholders view the same data. Therefore, businesses can collaborate without arbitrators, enabling a more efficient, automated, and error-free managing of transportations [30]. These benefits are not only available to established partners but also to short-term cooperations, since positive historical data enables a company to build up a good reputation for its service. Thus, blockchain technology enables the efficient use of crowdshipping, where one shipping company can use up the remaining capacities from a (previously unknown) competitor [65]. This benefit is not only profitable for both companies but is also efficient for the environment due to reduced greenhouse gas emissions. Specifically for new and short-term relationships between companies, other benefits, e.g., insight into partner's reputation, faster and cheaper cross-border payments, certification of green vehicles, or management without a trusted intermediary, make the use of blockchain technology attractive [65].

Similar to other industries, having trustworthy information available on the responsible and sustainable handling of products during its transportation is also attractive to end-customers of goods [65]. Sustainability information, in the form of the carbon footprint of a product's transportation, benefits especially, since the current disclosure of this information is, due to different reporting methodologies by various subcontractors, not done homogeneously across the logistics industry [55]. Finally, if issues arise during a product's transportation, readily available, detailed, and transparent tracking information allows for fast issue identification and

potential fault recovery as well as the identification of the responsible entity [116]. Unfortunately, such issues are not handled satisfactorily in many cases today, since necessary information is hard to retrieve, missing, or simply not available [116].

4.7 Waste Management

Besides improving sustainability in many previously discussed use cases, blockchain technology can also have a major impact on waste management, one of the most important industries in realizing a more sustainable and less wasteful lifestyle. Waste management encompasses three major aspects: extending a product's lifetime, recycling disposed products, and responsibly handling unavoidable waste. All these three aspects are handled suboptimally today, mainly because of various involved actors with (partially) opposing incentives and goals. Blockchain technology can improve the current situation by generating trust among these distrustful parties, providing transparency, ensuring accountability, and enabling short-term collaboration.

To extend a product's lifetime, a product must be maintained properly and easily resellable when it is no longer needed by the original owner. The former can be improved through blockchain technology by enabling active maintenance management, i.e., a product, e.g., some machinery, is equipped with sensors that collect data to proactively schedule maintenances when, e.g., production cycles or sensor reading passes a threshold [73]. Here, blockchains could manage the relation between product owner and maintainer to ensure adequate, but not excessive maintenance, and to provide the maintainer with a tamper-proof history of a product's usage [73]. This history can also be used to make educated decisions on when to update or replace existing hardware [73]. Having access to a reliable history of a product is also important when it comes to reselling, e.g., a car checkbook as trusted history significantly influences a car's resale value and hints at the state of the vehicle. Such benefits can extend to other products, benefiting resellers and buyers alike. If reselling is also managed transparently on the blockchain, this information could provide buyers and sellers with price estimates for products based on statistics of former trades of products in similar conditions [73]. Blockchain technology can also pave the way to ensure secondhand customers that a product was sustainably and environmentally friendly produced [65], which can further increase resale values [48, 92].

Blockchain technology can also improve product recycling at the end of a product's life. Here, it facilitates finding short-term business partners that may have a use for disassembled products, e.g., rubble from a demolished building can be used by a construction company, saving costs for both parties [65]. However, as recycling is not always lucrative, laws and regulations are often ignored. Here, blockchain technology can improve coordination between various actors and establish transparency to properly enforce laws [50]. Additionally, blockchains can realize a fine-grained incentive system that encourages recycling through micro-

payments [66]. Finally, blockchain technology can be used to certify new products that are built from recycled goods. Providing customers with this information in a trustworthy way also benefits the product sales, since the likelihood of people buying green products increases with the trustworthiness of the certification [48, 92].

Lastly, unavoidable waste is often not appropriately disposed of and, thus, harmful to the environment. The United States alone spends multiple million dollars annually on the cleanup of poorly managed landfills [65]. While transparency on *claimed* behavior is not directly revealing the entities responsible for illegal waste disposal, it allows potential collaborators or customers to verify which parties are honest and reputable, thus facilitating prosecutions by narrowing the number of potential malicious actors. Additionally, blockchains can arrange the waste disposal process when no entity is trusted or willing to manage the process, e.g., for cross-border waste disposal [90]. Such a change could enable more efficient waste flows and faster administration with less overhead. Finally, blockchain technology offers the foundation for systems where citizens can interact and contribute to public bodies [37]. Such systems can be used to publicly report and track, e.g., waste or potholes in cities and their remediation [114].

4.8 Insurance Industry

A second service industry that sees major research into the use of blockchain technology is the insurance industry [10, 13, 16, 26, 41]. Insurances rely on contracts between the insurer and the customer to define the conditions that must be met to legally force payments to the customer. However, the business model of insurances suffers from contradicting interests of insurers and customers, i.e., in the case of damages or losses, customers demand fast and unbureaucratic payments based on comprehensive rules. Contrarily, insurers aim to minimize their expenses to achieve higher profits and better competitiveness. As a consequence, the corresponding insurance contracts tend to exhibit excessive complexity leading to unsatisfied customers. Furthermore, the insurer usually conducts advanced investigations in case of significant claims to counter fraud by malicious customers that try to trick the insurer by providing unjustified claims [7]. To improve the handling of claims in this complex field, the insurance industry investigates blockchain technology as a way to improve insurance policies by simplifying associated processes, speeding up claim processing, and providing transparency for decisions. Thus, blockchain technology can achieve higher customer satisfaction and, at the same time, increase the business performance of insurers, e.g., by attracting new customers through their reputation or a reduction of work-intensive (and thus costly) investigations.

A common use of blockchain technology in the context of insurance is to model the insurance policy as a smart contract and to automate behavior, e.g., by linking sensors [10, 26, 41]. Representing the insurance policy as a smart contract increases the transparency towards the customer as the policy becomes publicly verifiable [41] and – considering the complexity of traditional representations [10] – potentially

improves the comprehensibility. Further, the conditions for successful claims are pre-defined, which increases the trust of the customers towards the insurer [10]. As a result, claim processing can be automated, simplified, and sped up.

Beyond specifying policies as on-chain smart contracts, the linking of external information promises additional benefits for insurances. Sensor data, e.g., for home or car insurances [10, 26], can speed up the processing of insurance claims and optimally even issue automated payments. Similarly, fine-granular pricing models, e.g., pay-per-use policies where the costs of a car insurance depend on the driving behavior of the customer [41], can be realized when linking sensors to an on-chain contract. However, sensor recordings can easily interfere with privacy demands or regulations, such that respective approaches require sophisticated privacy preservation measures. Applications that do not require private customer information, e.g., anonymized travel insurances with automated payback in case of canceled flights or trains [16] or automated payments for crop damage reported by external weather data [41], exhibit lower hurdles for the deployment of blockchain-backed solutions.

Finally, while traditional insurance relationships require the insurer to be a superior and centralized entity, blockchain technology enables peer-to-peer insurances without the need for a centralized and trusted third party [41]. To this end, the combination of the distributed ledger and one or multiple smart contracts takes the role of the previously centralized insurer and manages capital, policies, and the handling of claims. While initial platforms have been proposed [27, 101], ensuring appropriate financial reserves as well as fraud-resistant, reliable, and legally binding claim handling remains an active challenge for peer-to-peer insurances.

4.9 Gemstone Mining

The increasing desire for sustainable products [48, 92] also extends to luxury goods [28, 86]. Especially responsibly sourced gemstones are in high demand as the industry has a bad record for being exploitative [121]. Consequently, gemstone certificates, such as *GIA Diamond Origin Reports* and *AGTA Certificates*, are an important part of high-end gem and jewelry markets [24]. Various certification processes offer different levels of protection against fraud by relying on legislature, deploying third-party observers, or analyzing the chemical and physical properties of each stone in a lab. However, these certifications are still susceptible to fraud and entail significant costs [112]. Blockchain-backed certification efforts can be more secure, informative, and reliable as they provide immutable records of origin, allow for continuous tracking of the gemstones' ownership, and provide this information transparently to the public, thus giving businesses a competitive edge [112].

Still, the correctness of uploaded certification information has to be ensured, for which several preemptive and reactive measures have been proposed. Preemptive measures comprise, e.g., the soaking of gemstones in mine-specific nanoparticles that fill small fissures and present a unique property that can be linked to the blockchain [24]. However, this approach requires advanced technology and

logistics. It is further inapplicable for pure elements, e.g., diamonds [24]. Other preemptive approaches range from data entry validation by third parties, over the engraving of small QR codes, to the insertion of RFID chips into pearls [24, 112]. Reactive measures comprise physical or chemical analysis to verify the correctness of blockchain data and raise alarms in case of anomalies. Additionally, a global view on supply movement enables the detection of systematic cheating with expensive lab analysis of random samples. Compared to a paper-based certification, blockchain-backed certificates hence offer new mitigation mechanisms against fraud and can additionally store tracking and tracing information over a gemstone's lifetime. Thus, these certificates can, e.g., attest that a gemstone was not involved in illicit practices [112].

Blockchain technology can further benefit businesses in the gemstone industry by, e.g., reducing the need for intermediaries and trusted third parties to verify, audit, and certify the supply chain [112]. Furthermore, blockchains offer easier accounting and better interoperability between businesses due to a uniform interface and transparent information [77]. These benefits also apply to (small-scale) miners, benefiting from price transparency and thus fairer pay if blockchain-backed payment systems are deployed [112]. Finally, supply chain transparency makes it easier to ensure compliance with laws and regulations by governmental agencies [112]. The latter benefit is especially relevant since current schemes to ensure conflict-free gemstone supplies, e.g., the *Kimberley Process*, are increasingly put into question for lack of effectiveness and transparency [44, 63]. Consequently, the council members behind the *Kimberley Process Certification Scheme* actively seek to strengthen their certification process through blockchain technology [67].

5 Blockchain Technology for Industry 4.0: The Big Picture

Previously, we analyzed the benefits of blockchain technology for industrial applications, discussed challenges that emerge when striving to apply it in industrial scenarios, and surveyed mature ideas and deployments for blockchain-supported industrial use cases. In the following, we discuss the gathered insights from a broader perspective. Specifically, we classify the relevance of blockchain benefits for different industries and identify corresponding future research challenges.

5.1 Assessing Blockchain Benefits for Industry 4.0

As presented before, the transparency (⊞I) and accountability (⊞II) available to blockchain-enabled use cases benefit nearly all kinds of industries. The use of blockchains promises to, e.g., foster collaboration, optimize production processes, advance automation and payment processes, ease trustworthy offloading of tasks, enable product life cycle tracking, improve monitoring of legal compliance, and

Table 1 Relevance of blockchain-based benefits for different industries. We provide a relative and qualitative relevance classification, ranging from non-existent ○ to exceptional ●. The relative importance differs for each industry, but each benefit is relevant for almost all industries

	Manufacturing	Lumber Industry	Petroleum Industry	Energy Industry	Food Supply	Logistics	Waste Management	Insurance	Gemstone Industry
Transparency (⊞ I)									
Business-to-Business	●	●	◐	◕	◔	●	◔	◕	●
Business-to-Customer	◐	●	◕	◔	●	●	◔	●	●
Customer-to-Customer	○	○	○	◔	○	○	◐	◐	●
Accountability (⊞ II)									
Data Accountability	●	◔	◕	◐	◔	◔	◔	●	◔
Legal Compliance	◕	◔	◔	○	●	◔	◐	◐	◔
Building Reputation	◐	◕	◕	◐	◐	◔	◐	◔	●
Collaboration (⊞ III)									
Already Established	◔	◐	◔	◐	◐	◔	◐	◕	◕
New and Ad-hoc Collaborations	●	◐	○	◔	◐	●	◐	◔	◐
Automation (⊞ IV)									
Easing Payment Processing	◔	◐	●	●	○	◕	◕	◔	◐
Machine-to-machine Communication	●	◐	◕	◔	◐	◐	○	●	○
Fostering Digitalization (⊞ V)									
Disintermediation	◕	○	◐	◔	○	◕	○	◐	◔
Digital Certification	◔	◔	◕	◕	●	◔	◔	○	●
Blockchain-backed Contracts	◔	◕	◔	◔	◕	◔	◕	●	◕

relevance: ○ none ◕ limited ◐ partial ◔ significant ● exceptional

encourage digitalization. All of this contributes to higher profits and an improved product or service quality under the paradigm of a yet unachieved level of transparency between collaborators, towards customers, or even for the general public. Other businesses increasingly value secondary goals, such as increased customer satisfaction or an improved environmental footprint, due to the potential gain of reputation. Blockchain technology is thus relevant for many industries.

To classify the importance of blockchain-based improvements, we provide a qualitative overview of the relevance of envisioned benefits for the discussed industries in Table 1. Where literature lacks corresponding information, we derive the relevance from industries with similar properties to provide a comprehensive

picture. We posit that our overview can similarly hint at the relevance of benefits in industries not covered in this chapter, e.g., the drug production industry [1, 17].

Transparency Transparency is relevant for all covered industries. Especially transparency between companies or between a business and their customers exhibits a broad relevance, while customer-to-customer transparency is only relevant for a subset of industries. Here, the gemstone industry has been identified as a leading beneficiary, since these high-value goods suffer from criminal intrigues, i.e., the trading of blood diamonds, while both customers and involved businesses show a salient awareness of this issue. Furthermore, industries that qualify for peer-to-peer applications or second-hand markets, i.e., the energy market, insurances, and waste management businesses, can also profit from customer-to-customer transparency.

Accountability As argued before, transparency requires accountability to unfold its full potential, such that industries that demand transparency tend to similarly benefit from data accountability. Beyond that, the blockchain's potential to enable or improve legal compliance especially benefits industries with strong legislative regulations, e.g., the oil, food, and logistics industries, or where malicious or even criminal actors are historically expected, e.g., the lumber and gemstone industries. Furthermore, the possibility to build up reputation shows a significant relevance for those industries where (i) customers directly take part in the industrial applications (e.g., Local Energy Markets (LEMs)), (ii) customers and related businesses are especially aware of issues or care for high quality and specific product properties (food, logistics, gemstones), or (iii) the customer would be directly affected by inappropriate actions of the involved businesses (e.g., with insurances).

Collaboration Businesses benefit from building reputation via blockchain technology when they are able to attract new end customers or establish new business relationships and collaborations. The ability to foster collaborations is especially relevant for industries that observe high volatility in their business relationships, e.g., manufacturing, logistics, and insurances. Contrarily, the petroleum industry exhibits rather rigid business relationships with limited capabilities for establishing new relationships, but still profits from transparency and reputation as they aid to maintain the existing relationships in the face of many legislative regulations.

Automation Automation-related use cases profit from on-chain contracts that offer transparent data handling mechanisms as well as digital cryptocurrencies. Eased payment processes benefit the energy industry with its peer-to-peer applications; the insurance industry, where automated payments promise to reduce administrative expenses and increase customer satisfaction; and the petroleum and manufacturing industries, which depend on international trades or machine-based digital payments.

Fostering Digitalization Finally, blockchain technology removes the necessity for a trusted third party, which is of special relevance for digitalized payment processes in international settings (petroleum and energy), to avoid pricing monopolies or account for low-trust and potentially criminal stakeholders (gemstones). For the latter, digital certifications also show increased relevance, e.g.,, in industries that

target to ensure product quality along the supply chain (manufacturing, food, logistics) or industries that need to provide certificates to the final customer (food or lumber).

Summarizing Table 1, we observe higher-level properties to influence the relevance of specific benefits for different industries. Hence, research regarding a specific industry can also provide valuable insights for related industries or even those industries that only share a single common aspect. However, for every benefit and the features that blockchain technology provides, implications regarding potential challenges always have to be considered as we conclusively discuss hereinafter.

5.2 Open Challenges for a Blockchain-Enabled Industry 4.0

Blockchain technology, and especially its industrial application, is still in its infancy. In the following, we highlight the major challenges towards blockchain-based industrial processes, thereby covering technical, legal, and social aspects.

Education A lack of knowledge and expertise is considered a major barrier for blockchain technology in industrial applications [20]. First, management bodies neither understand the technology nor the potential benefits nor arising challenges, which slows down innovation as the necessary investments are missing. Second, due to the novelty of blockchain technology, the technical departments of companies often do not fully understand the technology themselves and lack exemplary deployments for studies. Educating the various actors in the industrial sectors can thus already lead to more and better blockchain usage in the industry and speed up its deployment.

Legislature Legal uncertainties are also a hurdle for blockchain adoption [20]. First, taxation on trades executed via cryptocurrencies is not properly regulated, which may lead to lawsuits. Second, to which extent digitally attested data, e.g., sensor data or contracts stored on blockchains, will provide evidence in a court of law remains unclear to date. Third, illegal transactions on the blockchain, e.g., those containing illicit data [78], could potentially negatively influence or harm businesses [74]. Therefore, establishing a reliable legal framework for the use of blockchain technology is crucial for the success of blockchain-supported approaches. With such a framework, courts could even directly take the role of an arbitrator for smart contracts, enabling faster and more efficient resolution of disputes [119].

Data Correctness Blockchains ensure the immutability of stored data, but yet lack the means to verify the correctness of data submitted for inclusion. Consequently, we have noticed proposals throughout the industrial sectors that attempt to mitigate the risk of erroneous data on the blockchain. These proposals can be categorized into preemptive and reactive measures. Preemptive measures, e.g., the use of temper-resilient sensors [10, 96], attempt to prevent the upload of erroneous data in the first

place. Reactive measures, e.g., using proteomics to verify the origin of wood [33], allow investigators to verify random samples or suspicious data already uploaded. If incorrect data can only be detected after insertion, businesses potentially also need ways, e.g., lawsuits or on-chain rollbacks, to handle that (automatic) decisions were executed based on wrong data. To this end, linking the provision of incorrect data to a loss of (transparently maintained) business reputation [76] holds businesses accountable for their actions. It enables a careful selection of business partners for future business relationships. Despite these countermeasures, researchers need to further investigate ways to inhibit, detect, and react to erroneous data on blockchains.

Scalability Scalability challenges accompany the immutable, persistent, and distributed nature of blockchains. Industrial applications can generate significant amounts of data, which must be handled [71, 124]. Some applications require low-latency transactions [38], while others are less dependent on transaction latency but require a high throughput of transactions [10] or larger payloads. Correctly choosing the appropriate ledger technology and implementing scalability features (cf. Sect. 3) such as data aggregation [96, 126], off-chain storage [11, 97], block pruning [79, 81], or concepts such as side-chains [9] and sharding [129] heavily impact the applicability of blockchain technology in different industrial sectors. Hence, future research should cover the development of new scalability improvements or optimize existing approaches. Further, increasing awareness for these measures, especially their advantages and drawbacks, is inevitable for a blockchain-backed Industry 4.0.

Ledger Interoperability Different requirements towards privacy, performance, latency, bandwidth, and smart contract complexity may require the combined use of several different blockchains [112]. While complex business relationships cannot be realized with a single blockchain [74], e.g., companies might only want to share some sensor data publicly in contrast to publicly sharing their business relationships, the interoperability of the required ledger technologies remains challenging. Current cross-chain technologies, e.g., atomic swaps [119] and side-chains [9], still can neither transfer significant amounts of data nor link highly specialized blockchains.

Transparency vs. Privacy Trade-Offs Initially, blockchain technology aimed to bring full transparency and decentralization to processes. However, full transparency is not always desired for industrial applications, as business secrets still have to be protected [11, 97]. Consequently, industrial use cases currently often use only permissioned and private blockchains. However, this choice limits the utility of blockchain-supported approaches, as these blockchain types often do not provide the required level of transparency for fully exploiting the benefits of blockchain technology. Modeling business processes with more fine-granular privacy control on blockchains is, therefore, an important research question. Fine-granular access control is not only important for data stored on the blockchain but also for business logic modeled in smart contracts or related information that is stored off-chain.

6 Conclusion

The ongoing digitalization process heavily influences the industrial sector, where Industry 4.0 yields new use cases for inter-business collaborations, monitoring and exchange of sensor-based product and production information, and digitized modeling and mediation of business processes. Blockchain technology provides transparency, immutability, accountability, and a decentralized consensus for digitalized data. It thus offers significant value for the progression towards an Industry 4.0 as well as manifold individual use cases within a variety of industrial sectors. In this chapter, we analyzed the potential benefits of making blockchain technology an integral part of industrial applications, discussed challenges that industries face when adapting the technology, and surveyed case studies for heterogeneous industries from the primary, secondary, and tertiary sectors. Despite the technology's potential benefits for industrial applications, its adoption is still in its infancy, calling for further research, especially on the identified open research challenges.

Acknowledgments Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

1. K. Abbas, M. Afaq, T. Ahmed Khan, W.C. Song, A blockchain and machine learning-based drug supply chain management and recommendation system for smart pharmaceutical industry. Electronics **9**(5) (2020)
2. Accenture, The blockchain benefit. Technical report, 2018
3. J. Ahn, A. Pellicone, B.S. Butler, Open badges for education: what are the implications at the intersection of open systems and badging? RLT **22** (2014)
4. E. Al Kawasmi, E. Arnautovic, D. Svetinovic, Bitcoin-based decentralized carbon emissions trading infrastructure model. Syst. Eng. **18**(2), 115–130 (2015)
5. T. AlSkaif, B. Holthuizen, W. Schram, I. Lampropoulos, W. van Sark, A blockchain-based configuration for balancing the electricity grid with distributed assets. WEVJ **11**(4) (2020)
6. F. Antonucci, S. Figorilli, C. Costa, F. Pallottino, L. Raso, P. Menesatti A review on blockchain applications in the agri-food sector. J. Sci. Food Agric. **99**(14), 6129–6138 (2019)
7. Association of British Insurers The con's not on – Insurers thwart 2,400 fraudulent insurance claims valued at £25 million every week (2017). https://www.abi.org.uk/news/news-articles/2017/07/the-cons-not-on--insurers-thwart-2400-fraudulent-insurance-claims-valued-at-25-million-every-week. Accessed 08 Sep 2021
8. G. Auld, L.H. Gulbrandsen, C.L. McDermott, Certification schemes and the impacts on forests and forestry. Annu. Rev. Environ. Resour. **33**, 187–211 (2008)
9. A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains. Technical report, 2014
10. L. Bader, J.C. Bürger, R. Matzutt, K. Wehrle, Smart contract-based car insurance policies, in *GC Workshops* (2018)
11. L. Bader, J. Pennekamp, R. Matzutt, D. Hedderich, M. Kowalski, V. Lücken, K. Wehrle, Blockchain-based privacy preservation for supply chains supporting lightweight multi-hop information accountability. IP&M **58**(3) (2021)

12. S.S.M. Bahrudin, M.I. Illyas, M.I. Desa, Tracking and tracing technology for halal product integrity over the supply chain, in *ICEEI* (2011)
13. A. Baliga, I. Subhod, P. Kamat, S. Chatterjee, Performance evaluation of the quorum blockchain platform (2018). arXiv:1809.03421
14. M. Bartoletti, L. Pompianu, An analysis of Bitcoin OP_RETURN metadata, in *LNCS* (2017)
15. R. Benbunan-Fich, A. Castellanos, Digitalization of land records: from paper to blockchain, in *ICIS* (2018)
16. T. Bertani, K. Butkute, F. Canessa, Smart Flight Insurance – InsurETH. London fintech week (2017)
17. T. Bocek, B.B. Rodrigues, T. Strasser, B. Stiller, Blockchains everywhere – a use-case of blockchains in the pharma supply-chain, in *IM* (2017)
18. P. Brauner, M. Dalibor, M. Jarke, I. Kunze, I. Koren, G. Lakemeyer, M. Liebenberg, J. Michael, J. Pennekamp, C. Quix, B. Rumpe, W. van der Aalst, K. Wehrle, A. Wortmann, M. Ziefle, A computer science perspective on digital transformation in production. TIOT **3**(2), 32pp. (2022)
19. A.F. Buckhorst, B. Montavon, D. Wolfschläger, M. Buchsbaum, A. Shahidi, H. Petruck, I. Kunze, J. Pennekamp, C. Brecher, M. Hüsing, B. Corves, V. Nitsch, K. Wehrle, R.H. Schmitt, Holarchy for line-less mobile assembly systems operation in the context of the internet of production, in *ICME* 448–453(2020)
20. M. Budman, R. Bhat, S. Borodoloi, Deloitte's 2020 global blockchain survey – from promise to reality. Techniccal report, 2020
21. C. Burchert, C. Decker, R. Wattenhofer, Scalable funding of bitcoin micropayment channel networks. R. Soc. Open Sci. **5**(8) (2018)
22. L. Burkhalter, A. Hithnawi, A. Viand, H. Shafagh, S. Ratnasamy, TimeCrypt: encrypted data stream processing at scale with cryptographic access control, in *NSDI* (2020)
23. G. Cann, E. Catmur, Blockchain: overview of the potential applications for the oil and gas market and the related taxation implications. Technical report, 2017
24. L.E. Cartier, S.H. Ali, M.S. Krzemnicki, Blockchain, chain of custody and trace elements: an overview of tracking and traceability opportunities in the gem industry. J. Gemmol. **36**(3), 212–227 (2018)
25. M. Dahlmanns, C. Dax, R. Matzutt, J. Pennekamp, J. Hiller, K. Wehrle, Privacy-preserving remote knowledge system, in *ICNP* (2019)
26. S. Davies, Bitcoin: possible bane of the diamond thief (2015). https://www.ft.com/content/f2b0b2ee-9012-11e4-a0e5-00144feabdc0. Accessed 08 Sep 2021
27. J. Davis, Peer to peer insurance on an ethereum blockchain. White paper, 2015
28. M. De Angelis, F. Adıgüzel, C. Amatulli, The role of design similarity in consumers' evaluation of new green products: an investigation of luxury fashion brands. JCP **141**, 1515–1527 (2017)
29. C. Decker, R. Wattenhofer, A fast and scalable payment network with bitcoin duplex micropayment channels, in *LNCS* (2015)
30. M. Dobrovnik, D.M. Herold, E. Fürst, S. Kummer, Blockchain for and in logistics: what to adopt and where to start. Logistics **2**(3) (2018)
31. Z. Doubrovski, J.C. Verlinden, J.M.P. Geraedts, Optimal design for additive manufacturing: opportunities and challenges, in *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference* (2011)
32. J. Duan, C. Zhang, Y. Gong, S. Brown, Z. Li, A content-analysis based literature review in blockchain adoption within food supply chain. IJERPH **17**(5) (2020)
33. B. Düdder, O. Ross, Timber tracking – reducing complexity of due diligence by using blockchain technology. CEUR-WS 1898 (2017)
34. M. Eggert, R. Häußling, M. Henze, L. Hermerschmidt, R. Hummen, D. Kerpen, A.N. Pérez, B. Rumpe, D. Thißen, K. Wehrle, *SensorCloud: Towards the Interdisciplinary Development of a Trustworthy Platform for Globally Interconnected Sensors and Actuators* (Springer, Cham 2014)

35. H. Etemad, R.W. Wright, L.P. Dana, Symbiotic international business networks: collaboration between small and large firms. TIBR **43**(4),481–499 (2001)
36. Everledger Limited, Everledger (2017). https://www.everledger.io
37. M. Farnaghi, A. Mansourian, Blockchain, an enabling technology for transparent and accountable decentralized public participatory GIS. Cities **105** (2020)
38. K. Gai, Y. Wu, L. Zhu, M. Qiu, M. Shen, Privacy-preserving energy trading using consortium blockchain in smart grid. TII **15**(6), 3548–3558 (2019)
39. A. Gannouni, V. Samsonov, M. Behery, T. Meisen, L. Gerhard, Neural combinatorial optimization for production scheduling with sequence-dependent setup waste, in *SMC* (2020)
40. C.T.B. Garrocho, M.C. Silva, C.M.S. Ferreira, C.F.M. da Cunha Cavalcanti, R.A.R. Oliveira, Real-time systems implications in the blockchain-based vertical integration of industry 4.0. Computer **53**(9) (2020)
41. V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, V. Santamaría, Blockchain and smart contracts for insurance: is the technology mature enough? Futur. Internet **10**(2) (2018)
42. A.H. Gausdal, K.V. Czachorowski, M.Z. Solesvik, Applying blockchain technology: evidence from Norwegian companies. Sustainability **10**(6) (2018)
43. L. Gleim, J. Pennekamp, M. Liebenberg, M. Buchsbaum, P. Niemietz, S. Knape, A. Epple, S. Storms, D. Trauth, T. Bergs, C. Brecher, S. Decker, G. Lakemeyer, K. Wehrle, FactDAG: formalizing data interoperability in an internet of production. IoTJ **7**(4), 3243–3253 (2020)
44. Global Witness Why we are leaving the Kimberley Process (2011). https://www.globalwitness.org/en/archive/why-we-are-leaving-kimberley-process-message-global-witness-founding-director-charmian-gooch/. Accessed 08 Sep 2021
45. Ó. González-Benito, P.A. Muñoz-Gallego, E. García-Zamora, Role of collaboration in innovation success: differences for large and small businesses. J. Bus. Econ. Manag. **17**(4), 645–662 (2016)
46. W. Gräther, S. Kolvenbach, R. Ruland, J. Schütte, C. Torres, F. Wendland, Blockchain for education: lifelong learning passport, in *ERCIM Blockchain Workshop* (2018)
47. A. Grench, A.F. Camilleri, Blockchain in education. Technical report, 2017
48. C. Groening, J. Sarkis, Q. Zhu, Green marketing consumer-level theory review: A compendium of applied theories and further research directions. JCP **172**, 1848–1866 (2018)
49. P. Grover, A.K. Kar, P. Vigneswara Ilavarasan, Blockchain for businesses: a systematic literature review, in *LNCS* (2018)
50. N. Gupta, P. Bedi, E-waste management using blockchain based smart contracts, in *ICACCI* (2018)
51. M.U. Gurmani, T. Sultana, A. Ghaffar, M. Azeem, Z. Abubaker, H. Farooq, N. Javaid, Energy trading between prosumer and consumer in P2P network using blockchain, in *LNNS* (2019)
52. N. Hackius, M. Petersen, Blockchain in logistics and supply chain: trick or treat?, in *HICL* (2017)
53. Y. Hanada, L. Hsiao, P. Levis, Smart contracts for machine-to-machine communication: possibilities and limitations, in *IOTAIS* (2018)
54. M. Henze, The quest for secure and privacy-preserving cloud-based industrial cooperation, in *SPC* (2020)
55. D.M. Herold, K.H. Lee, The influence of the sustainability logic on carbon disclosure in the global logistics industry: the case of DHL, FDX and UPS. Sustainability **9**(4) (2017)
56. G. Hileman, M. Rauchs, Global blockchain benchmarking study. Technical report, 2017
57. J. Hiller, M. Henze, M. Serror, E. Wagner, J.N. Richter, K. Wehrle, Secure low latency communication for constrained industrial IoT scenarios, in *LCN* (2018)
58. J. Hiller, J. Pennekamp, M. Dahlmanns, M. Henze, A. Panchenko, K. Wehrle, tailoring onion routing to the internet of things: security and privacy in untrusted environments, in *ICNP* (2019)
59. Y. Huang, J. Wu, C. Long, Drugledger: a practical blockchain system for drug traceability and regulation, in *Blockchain* (2018)
60. R. Hummen, H. Wirtz, J.H. Ziegeldorf, J. Hiller, K. Wehrle, Tailoring end-to-end IP security protocols to the Internet of Things, in *ICNP* (2013)

61. R. Hummen, H. Shafagh, S. Raza, T. Voig, K. Wehrle, Delegation-based authentication and authorization for the IP-based Internet of Things, in *SECON* (2014)
62. J. Hwang, M. Choi, T. Lee, S. Jeon, S. Kim, S. Park, S. Park, Energy prosumer business model using blockchain system to ensure transparency and safety. Energy Proc. **141**, 194–198 (2017)
63. IMPACT 2017 Plenary – Statement of IMPACT (2017). https://www.kimberleyprocess.com/en/2017-plenary-statement-impact. Accessed 08 Sep 2021
64. A. Kamilaris, A. Fonts, F.X. Prenafeta-Boldν, The rise of blockchain technology in agriculture and food supply chains. Trends Food. Sci. Technol. **91**, 640–652 (2019)
65. M. Kouhizadeh, J. Sarkis, Blockchain practices, potentials, and perspectives in greening supply chains. Sustainability **10**(10) (2018)
66. M. Kouhizadeh, Q. Zhu, J. Sarkis, Blockchain and the circular economy: potential tensions and critical reflections from practice. Prod. Plan Control **31**(11–12), 950–966 (2020)
67. KP Chair 2016 Kimberley Process. Mid-term report, 2016)
68. J. Kruger, W.E. Oates, W.A. Pizer, Decentralization in the EU emissions trading scheme and lessons for global policy. REEP **1**(1) (2007)
69. S. Latif, Z. Idrees, J. Ahmad, L. Zheng, Z. Zou, A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things. J. Ind. Inf. Integration **21** (2020)
70. J. Leng, S. Ye, M. Zhou, J.L. Zhao, Q. Liu, W. Guo, W. Cao, L. Fu, Blockchain-secured smart manufacturing in industry 4.0: a survey. IEEE TSMC **51**(1), 237–252 (2020)
71. Z. Li, W.M. Wang, G. Liu, L. Liu, J. He, G.Q. Huang, Toward open manufacturing: a cross-enterprises knowledge and services exchange framework based on blockchain and edge computing. Ind. Manag. Data Syst. **118**(1), 303–320 (2018)
72. W. Lin, X. Huang, H. Fang, V. Wang, Y. Hua, J. Wang, H. Yin, D. Yi, L. Yau, Blockchain technology in current agricultural systems: from techniques to applications. IEEE Access **8**, 143920–143937 (2020)
73. X.L. Liu, W.M. Wang, H. Guo, A.V. Barenji, Z. Li, G.Q. Huang, Industrial blockchain based framework for product lifecycle management in industry 4.0. Robot Comput. Integr. Manuf. **63** (2020)
74. H. Lu, K. Huang, M. Azimi, L. Guo, Blockchain technology in the oil and gas industry: a review of applications, opportunities, challenges, and risks. IEEE Access **7**, 41426–41444 (2019)
75. S. Malik, S.S. Kanhere, R. Jurdak, ProductChain: scalable blockchain framework to support provenance in supply chains, in *NCA* (2018)
76. S. Malik, V. Dedeoglu, S.S. Kanhere, R. Jurdak, TrustChain: trust management in blockchain and IoT supported supply chains, in *Blockchain* (2019)
77. S. Mann, V. Potdar, R.S. Gajavilli, A. Chandan, Blockchain technology for supply chain traceability, transparency and data provenance, in *ICBTA* (2018)
78. R. Matzutt, J. Hiller, M. Henze, J.H. Ziegeldorf, D. Müllmann, O. Hohlfeld, K. Wehrle, A quantitative analysis of the impact of arbitrary blockchain content on bitcoin, in *LNCS* (2018)
79. R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, K. Wehrle, How to securely Prune bitcoin's blockchain, in *Networking* (2020)
80. R. Matzutt, J. Pennekamp, K. Wehrle, A secure and practical decentralized ecosystem for shareable education material, in *ICOIN* (2020)
81. R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, K. Wehrle, CoinPrune: shrinking bitcoin's blockchain retrospectively. IEEE TNSM **18**(3), 3064–3078 (2021)
82. E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, C. Weinhardt, A blockchain-based smart grid: towards sustainable local energy markets. Comput. Sci. Res. Dev. **33**(1–2), 207–214 (2018)
83. A. Miller, Y. Xia, K. Croman, E. Shi, D. Song, The honey badger of BFT protocols, in *CCS* (2016)
84. M. Mylrea, S.N.G. Gourisetti, Blockchain for smart grid resilience: exchanging distributed energy at speed, scale and security, in *RWS* (2017)
85. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system. White paper, 2008

86. J. Nash, C. Ginger, L. Cartier, The sustainable luxury contradiction: evidence from a consumer study of marine-cultured pearl jewellery. JCC **63**, 73–95 (2016)
87. W. Nikolakis, L. John, H. Krishnan, How blockchain can shape sustainable global value chains: an evidence, verifiability, and enforceability (EVE) framework. Sustainability **10**(11) (2018)
88. M. Nofer, P. Gomber, O. Hinz, D. Schiereck, Blockchain. Bus Inf. Syst. Eng. **59**(3) (2017)
89. E. Nyaletey, R.M. Parizi, Q. Zhang, K.K.R. Choo, BlockIPFS – blockchain-enabled interplanetary file system for forensic and trusted data traceability, in *Blockchain* (2019)
90. G. Ongena, K. Smit, J. Boksebeld, G. Adams, Y. Roelofs, P. Ravesteyn, Blockchain-based smart contracts in waste management: a silver bullet?, in *BLED* (2018)
91. Pacifical Pacifical MSC sustainable tuna now traceable via ethereum blockchain (2018). https://www.pacifical.com/pacifical-msc-sustainable-tuna-now-traceable-via-ethereum-blockchain/. Accessed 08 Sep 2021
92. K. Peattie, Towards sustainability: the third age of green marketing. Mark. Rev. **2**(2), 129–146 (2001)
93. J. Pennekamp, M. Dahlmanns, L. Gleim, S. Decker, K. Wehrle, Security considerations for collaborations in an industrial IoT-based lab of labs, in *GCIoT* (2019)
94. J. Pennekamp, R. Glebke, M. Henze, T. Meisen, C. Quix, R. Hai, L. Gleim, P. Niemietz, M. Rudack, S. Knape, A. Epple, D. Trauth, U. Vroomen, T. Bergs, C. Brecher, A. Bührig-Polaczek, M. Jarke, K. Wehrle, Towards an infrastructure enabling the internet of production, in *ICPS* (2019)
95. J. Pennekamp, M. Henze, S. Schmidt, P. Niemietz, M. Fey, D. Trauth, T. Bergs, C. Brecher, K. Wehrle, Dataflow challenges in an internet of production: a security & privacy perspective, in *CPS-SPC* (2019)
96. J. Pennekamp, F. Alder, R. Matzutt, J.T. Mühlberg, F. Piessens, K. Wehrle, Secure end-to-end sensing in supply chains, in *CPS-Sec* (2020)
97. J. Pennekamp, L. Bader, R. Matzutt, P. Niemietz, D. Trauth, M. Henze, T. Bergs, K. Wehrle, Private multi-hop accountability for supply chains, in *BIoTCPS* (2020)
98. J. Pennekamp, E. Buchholz, Y. Lockner, M. Dahlmanns, T. Xi, M. Fey, C. Brecher, C. Hopmann, K. Wehrle, Privacy-preserving production process parameter exchange, in *ACSAC* (2020)
99. J. Pennekamp, R. Matzutt, S.S. Kanhere, J. Hiller, K. Wehrle, The road to accountable and dependable manufacturing. Automation **2**(3) (2021)
100. J. Pennekamp, R. Matzutt, C. Klinkmüller, L. Bader, M. Serror, E. Wagner, S. Malik, M. Spiß, J. Rahn, T. Gürpinar, E. Vlad, S.J.J. Leemans, S.S. Kanhere, V. Stich, K. Wehrle, An interdisciplinary survey on information flows in supply chains. ACM Comput. Surv. **56**(2), 32, 1–38 (2024). https://doi.org/10.1145/3606693
101. A. Peperno, V. Kravchuk, E. Porubaev, Teambrella: a peer-to-peer coverage system. White paper, 2016
102. G. Perboli, S. Musso, M. Rosano, Blockchain in logistics and supply chain: a lean approach for designing real-world use cases. IEEE Access **6**, 62018–62028 (2018)
103. M. Pincheira Caro, M.S. Ali, M. Vecchio, R. Giaffreda, Blockchain-based traceability in agri-food supply chain management: a practical implementation, in *IoT Tuscany* (2018)
104. C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, M. Bertoncini, Blockchain based decentralized management of demand response programs in smart energy grids. Sensors **18**(1) (2018)
105. J. Premanandh, Horse meat scandal – a wake-up call for regulatory authorities. Food Control **34**(2), 568–569 (2013)
106. F. Protti-Alvarez, Oil and gas, petchem industries trialing blockchain solutions. Chemical week (2018)
107. W. Rodriguez, J. Zalewski, E. Kirche, Beyond intelligent agents: e-sensors for supporting supply chain collaboration and preventing the bullwhip effect. IJeC **3**(2), 1–15 (2007)
108. S. Seven, G. Yao, A. Soran, A. Onen, S. Muyeen, Peer-to-peer energy trading in virtual power plant based on blockchain smart contracts. IEEE Access **8**, 175713–175726 (2020)

109. H. Shafagh, A. Hithnawi, A. Droescher, S. Duquennoy, W. Hu, Talos: encrypted query processing for the internet of things, in *SenSys* (2015)
110. H. Shafagh, L. Burkhalter, A. Hithnawi, S. Duquennoy, Towards blockchain-based auditable storage and sharing of IoT data, in *CCSW* (2017)
111. A. Singh, V. Kumar, A.K. Ravi, K. Chatterjee, Ensuring food safety through blockchain. LNEE **668** (2020)
112. C.M. Stockton, Blockchain in the mining industry: implications for sustainable development in Africa. J. Gemmol. **37**(1) (2020)
113. H. Subramanian, Decentralized blockchain-based electronic marketplaces. CACM **61**(1), 78–84 (2017)
114. A. Thada, U.K. Kapur, S. Gazali, N. Sachdeva, S. Shridevi, Custom block chain based cyber physical system for solid waste management. Proc. Comput. Sci. **165**, 41–49 (2019)
115. F. Tian, An agri-food supply chain traceability system for China based on RFID & blockchain technology, in *ICSSSM* (2016)
116. E. Tijan, S. Aksentijević, K. Ivanić, M. Jardas, Blockchain technology implementation in logistics. Sustainability **11**(4) (2019)
117. D. Trauth, T. Bergs, W. Prinz, *The Monetization of Technical Data*, 1st edn. (Springer, Heidelberg, 2021)
118. C. Visser, Q.A. Hanich, How blockchain is strengthening tuna traceability to combat illegal fishing. Technical report, 2017
119. E. Wagner, A. Völker, F. Fuhrmann, R. Matzutt, K. Wehrle, Dispute resolution for smart contract-based two party protocols, in *ICBC* (2019)
120. E. Wagner, J. Bauer, M. Henze, Take a bite of the reality sandwich: revisiting the security of progressive message authentication codes, in *WiSec* (2021)
121. S. van der Wal, E. de Haan, Rough cut: sustainability issues in the coloured gemstone industry. Technical report, 2010
122. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in *LNCS* (2016)
123. G. Wood, Ethereum: a secure decentralised generalised transaction ledger. White paper, 2016
124. K. Wüst, A. Gervais, Do you need a blockchain?, in *CVCBT* (2018)
125. WWF, Enforcement review of the EU timber regulation (EUTR). Eu synthesis report, 2019
126. J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, N. Yu, Healthchain: a blockchain-based privacy preserving scheme for large-scale health data. IoTJ **6**(5), 8770–8781 (2019)
127. X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, P. Rimba, A taxonomy of blockchain-based systems for architecture design, in *ICSA* (2017)
128. F. Yiannas, A new era of food transparency powered by blockchain. Innov. Technol. Gov. Glob. **12**(1–2), 46–56 (2018)
129. M. Zamani, M. Movahedi, M. Raykova, RapidChain: scaling blockchain via full sharding, in *CCS* (2018)
130. G. Zyskind, O. Nathan, A.S. Pentland, Decentralizing privacy: using blockchain to protect personal data, in *SPW* (2015)

Blockchain for Health Data Management

Dinh C. Nguyen, Pubudu N. Pathirana, Ming Ding, and Aruna Seneviratne

This chapter presents a blockchain-based solution to build an e-health data management based on a hybrid edge cloud system. The proposed system enables the provision of real-time health service and provides secure user access to clouds, with privacy and security guarantees. Two smart contracts are designed to provide efficient health data storage and trustworthy data exchange among patients and healthcare providers. A case study is presented using an Ethereum blockchain platform with cloud computing to demonstrate the feasibility of the proposed e-health system in terms of data sharing efficiency, system security, and latency.

1 Introduction

The rapid development of cloud computing (CC) and the Internet of Medical Things (IoMT) has led to new opportunities for revolutionizing modern health services. The convergence of such innovative technologies has demonstrated its great potential and efficiency for offering inexpensive and smart healthcare services

D. C. Nguyen · P. N. Pathirana
School of Engineering, Deakin University, Burwood, VIC, Australia
e-mail: cdnguyen@deakin.edu.au; pubudu.pathirana@deakin.edu.au

M. Ding
Data61, CSIRO, Eveleigh, NSW, Australia
e-mail: ming.ding@data61.csiro.au

A. Seneviratne (✉)
School of Electrical Engineering and Telecommunications, University of New South Wales (UNSW), Sydney, NSW, Australia
e-mail: a.seneviratne@unsw.edu.au

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_18

[1–3]. IoMT with its ubiquitous nature provides potentials for remote patient monitoring and automated health status evaluations while allowing the mobility freedom of individuals. Meanwhile, the adoption of CC eliminates the need for expensive computing infrastructure of local servers and effectively overcomes the resource constraints of IoMT devices, thus enabling high-quality healthcare services. With these advantages, the combination of CC and IoMT has the potential to introduce novel healthcare solutions with different features in many areas of healthcare applications. However, the rapid growth of IoMT and CC poses numerous challenges that need to be addressed before they can be fully incorporated in healthcare systems. First, distributed IoMT devices generate high volumes of data which can lead to scalability problems and traffic congestion on the centralized cloud networks [4–6]. Second, due to the physical distance to IoMT devices, compared to the conventional clouds-based health services, they can experience high network latency which makes them unsuitable for time-sensitive medical applications. Finally, the storing of sensitive medical information on public clouds may introduce greater data privacy risks.

These challenges can be addressed by using edge clouds (ECs) which integrate storage and computing services at the edges of networks [7–9]. With the EC being closer to heterogeneous IoMT devices, data analytics, computing, and network services can be made highly available and scalable and also address the latency concerns. In addition, EC brings numerous other benefits such as efficient local distributed computing and, in general, improves the overall quality of services (QoS) and offer superior-user experience [10–12].

Although cloud base systems can be the basis for providing advanced e-health systems, they introduce critical security challenges [13–15]. This is exacerbated in e-health systems by the need for guarantees of privacy and security and the information to be shared among multiple medical providers. For instance, diagnostic results stored in the cloud and accessible by a doctor can make better clinical decisions and devise targeted treatment plans for patients. Thus, unauthorized access to the data in the cloud needs to be prevented to ensure no leakage or modification of sensitive patient information. Further, due to the nature of EMRs, patients need to be able to track and control access to their medical data. Therefore, it is critical to develop mechanisms that ensure the security and privacy and track the usage of patient data.

Recently, blockchains have been proposed as the basis for developing these mechanisms in e-health systems [1, 16, 17]. Blockchain is a distributed ledger technology that was first used as the public digital ledger of bitcoin for economic transactions. The concept of blockchain is based on a peer-to-peer network architecture in which transaction information is distributed among multiple nodes and not controlled by any single centralized entity, thus lending itself to adapt to manage patient data, user authentication, network integrity, and data privacy.

Blockchains utilize public-key cryptography to establish an append-only, immutable chain of blocks which are publicly accessible to all blockchain entities in a verifiable and trustworthy manner. This is achieved using a timestamp, the hash of the previous block, and a list of transactions with transmitted data [18].

In an e-health context, transactions may include updates or changes to medical records, address of patients, or information of medical providers. Another aspect of blockchain technology that could be leveraged is smart contracts. Smart contracts are computer programs controlling the transfer of digital currencies or assets between parties [19, 20]. Essentially, smart contracts are defined as self-executing scripts stored and replicated on the blockchain and visible to all network entities. Smart contracts can provide full [21] turning programmability, allowing independent execution of transactions in a prescribed manner. As blockchain entities can interact with a smart contract by addressing a transaction or message to it, they permit trusted transactions to be transmitted automatically among network nodes without the central controlling entity [22, 23], providing the necessary functionality to provide security and privacy guarantees in distributed systems.

This chapter describes a smart e-health architecture that utilizes blockchains and smart contracts. In particular, this chapter will describe:

1. A blockchain-based solution to build an e-health data management based on hybrid edge cloud system. The system enables the provision of real-time health service and provides secure user access to clouds, with privacy and security guarantees;
2. The use of smart contracts to provide efficient EMRs storage and trustworthy data exchange among patients and healthcare providers. It illustrates the use of one contract *medical record smart contract (MRSC)* for managing EMRs storage on the cloud and another, *data sharing smart contract (DSSC)* for managing data sharing among medical users; and
3. A case study that demonstrates the feasibility of the proposed e-health system and its evaluation of its performance in terms of data sharing efficiency, system security, and latency both analytically and empirically.

The remainder of this chapter is organized as follows. Section 2 introduces a new smart e-health architecture based on hybrid cloud and blockchain and smart contracts. The section also provides the design and analysis of the main components of the system. The smart contract designs are described in Sect. 3. Section 4 presents the use of blockchain and smart contracts for EMR storage and sharing. A case study of the use of a proof-of-concept demonstrator built on an Ethereum blockchain and Android applications is presented in Sect. 5, along with some analytical results. Conclusions are presented in Sect. 6.

2 Smart E-Health Architecture

This section provides an overview of distributed e-health system that uses blockchains. The system is schematically shown in Fig. 1 and provides a brief description of the different components. The system is representative and is one possible configuration that may be adopted, depending on the needs. Its primary purpose is to illustrate the use of ECs, blockchains, and smart contracts.

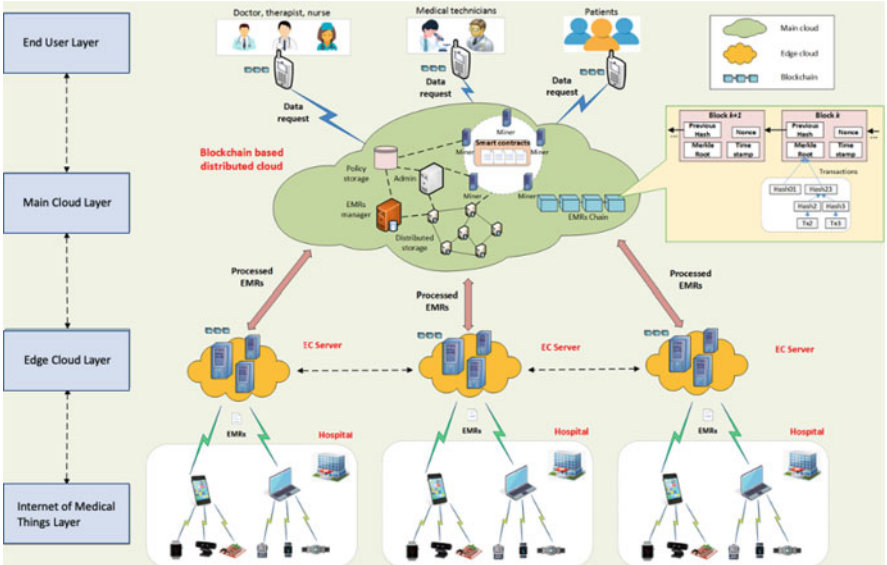


Fig. 1 Illustration of the proposed smart e-health architecture

2.1 Overview

2.1.1 Design Goals

One of the fundamental goals of any e-health system is to securely and efficiently distribute data among geographically distributed users while guaranteeing the privacy and security of data. There have been numerous studies which indicate that blockchain and smart contract technologies can be adapted to facilitate sharing and storage of data. In the case of an e-health system, such a system needs to satisfy the following requirements as a minimum.

- The system should provide functionality for e-health applications by providing access to the data that will enable automated clinical support services to accelerate healthcare processes.
- The system should support real-time services by achieving low network latency while ensuring efficiency and robustness.
- The system should achieve secure data storage and sharing among authorized users.
- The system should minimize risks of leakage of patient information. The proposed scheme should also have the ability to ensure the integrity and confidentiality of the data.

The following subsections present a representative operating environment which is used in the rest of the chapter. The objective is to describe how blockchain and smart contract technologies can be adapted for such environments rather than propose a specific e-health system architecture.

2.2 System

We consider an environment where patients are monitored using numerous devices and the relevant data collected. These devices could be worn by the patient or be in close proximity to them and are referred to as Internet of Medical Things (IoMT). IoMT devices form IoMT networks that are connected to servers via gateways. The data collected from the IoMT devices are stored and processed in the edge cloud (EC).[1]

In this chapter, it is assumed that IoMT networks in a hospital are connected via gateways to an EC. The ECs collect received data via the IoMT network gateways, process and store them, and send them to the central cloud. Blockchain and smart contract technology are used by the cloud system to provide the required functionality. The use of ECs reduces latency and enables the supporting of the real-time services and reducing data transmission cost. This system is schematically shown in Fig. 1.

The EC processes the IoMT data it receives and creates electronic medical records (EMRs) for each of the patients that comprise patient information collected by IoMT and other medical information about the patient such as the treatments the patient has received and the medical test results. In the system described here, they are denoted by *HospID*, *testID* and a patient ID *PID*. With this information, the address of a patient on a blockchain can be as *Addr = (HospID, testID, PID)*.

The use of ECs helps with satisfying the first requirement. The use of a network of distributed edge cloud nodes enables providing e-health services locally and alleviates latency and communication overheads. Further, the use of the distributed cloud storage system facilitates fast data retrieval and enables the lightweight data sharing scheme with simplified smart contracts and an efficient data storage strategy. These help in satisfying the second requirement.

The last two requirements are satisfied with the use of blockchain and smart contract technology with hashing to manage data storage and sharing.

The systems consist of four primary layers as shown, namely, Internet of Medical Things (IoMT), Edge Computing (EC), Main Cloud (MC), and End Users (EU). Brief descriptions of these layers are provided below.

[1] Storage at the edge of the cloud, closer to the data.

2.2.1 IoMT Layer

IoMT layer comprises hospitals that monitor patients using a network of IoMT devices. The IoMT devices collect data from the patients.

2.2.2 EC Layer

The EC layer comprises a cluster of edge cloud nodes and is associated with a set of hospitals and their IoMT networks. The EC will receive the data from the associated hospital and its IoMT networks via a gateway, process them, and combine the processed data with other medical data to create the electronic medical records (EMRs). The EMR created will be stored at the EC associated with the hospital and shared with authorized healthcare providers at the hospital. Finally, the EC will upload the ECs to the MC.

2.2.3 MC Layer

The MC controls all e-health systems operations and facilitates the sharing of information by providing access to end users. It consists of five components:

- *Admin:* manages operations and policies of smart contracts described below in Sect. 3. This is the only entity with the ability to modify and update policies that define access rights of users but does not store transactions.
- *EMRs manager:* is responsible for managing the EMRs on cloud storage. All requests for uploading data from ECs must be verified and accepted by the EMRs manager before being stored on the cloud. The EMRs manager also controls external access from users. This EMRs manager utilizes smart contracts to verify user requests.
- *Miners:* run a consensus algorithm, described below in Sect. 3.1 such as proof of work. They are computing systems with significant computing resources and enable the validation of new blocks that are added to the blockchain, thus ensuring security for blockchain.
- *Distributed storage system:* data generated from EC layers are stored in off-chain cloud storage in the distributed storage system. The distributed storage system is under the control of the EMRs manager. The data is classified and stored as shown in Fig. 2.
- *Storage policy:* stores policies and access rights to the data. This is used to verify user requests and grant permission for EMR uploading, data retrieval, and generating smart contracts.

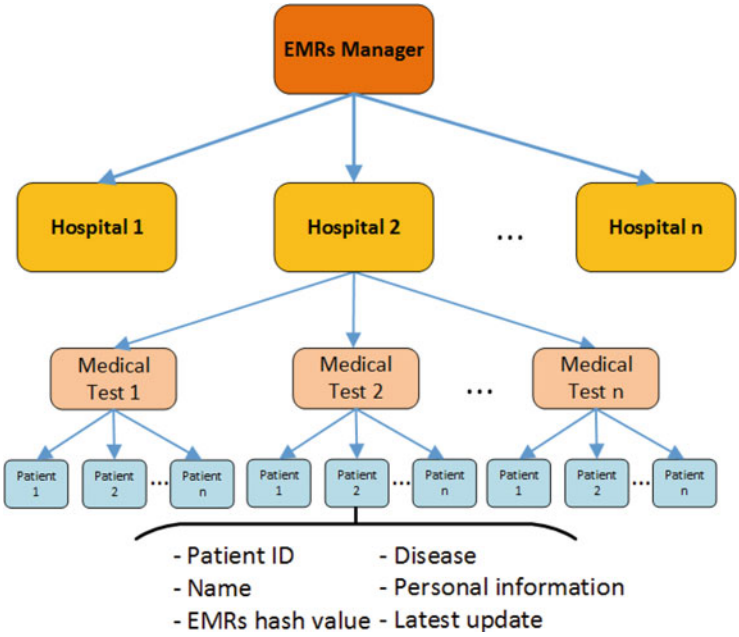


Fig. 2 Flowchart of EMR management system

2.2.4 Data User Layer

Authorized medical entities including patients and healthcare providers can get access to information via the main cloud layer using e-health system app.

3 E-Health Blockchain

In a distributed e-health system, the biggest challenge is providing secure access to data to numerous distributed entities. As mentioned, blockchain technology can be used to provide the necessary security and data provenance in such distributed systems. To illustrate this, this section describes the implementation of the proposed system using an Ethereum blockchain [24].

Ethereum is a second-generation blockchain technology similar to platforms such as Bitcoin, with its unique own features as described in [24]. Ethereum blockchain and smart contracts are used in the proposed system for EMR storage and sharing by storing transaction information and smart contracts.

3.1 Transaction

In the case of data storage, transactions include hash values of data and metadata of users such as patient address. All transactions are added to the blockchain (EMR chain) after being verified by the mining process as described below. Once added, the transactions are accessible by all entities within the end user layer.

Transaction Process on a Main Cloud Layer

Input: Transaction - Tx
Output: Verified block - $block$
1: A user signs off on a transaction Tx and stores it on the associated EC server
2: The EC server in turn transmits the transaction to the EMRs manager, which forwards it to the cloud miners
and stores it in a pool of unconfirmed transactions
3: Miners perform consensus
4: The miner that finds the signature first for the transaction broadcasts that
signature to all the other miners for verification
6: If that solution is found to be valid by the other miners,
the new block, $block$, is appended to the EMR chain
7: The verified $block$ with its signature is published to
other blockchain entities (including users)

Two smart contracts shown in Fig. 3 are used in EMR storage and sharing. The first, Medical Record Smart Contract (MRSC), manages the storage. The second, Data Sharing Smart Contract (DSSC), manages the sharing of data among the users, using the two policy lists described below.

3.2 Medical Record Smart Contract (MRSC)

MRSC comprises the following main components:

- ***Storage request:*** A IoMT network which wants to upload EMRs to an EC for processing sends a storage request to the EC, which in turn sends it to the EMRs manager for verification. Verification requests are identified using a gateway public key described below. The information stored within the MRSC policy is shown in Table 1. Only ECs registered with the MRSC policy are permitted to send data to the EMRs manager.

  MRSC uses the public key *PK*, the role of user *userRole*, and *Addr* as the address of patient in blockchain for the following functions.

  – *AddUser(PK, userRole):* (executed by Admin) This enables the addition of new users to the MRSC contract. User is identified by their public key and is added into the contract with a corresponding role based on their request.

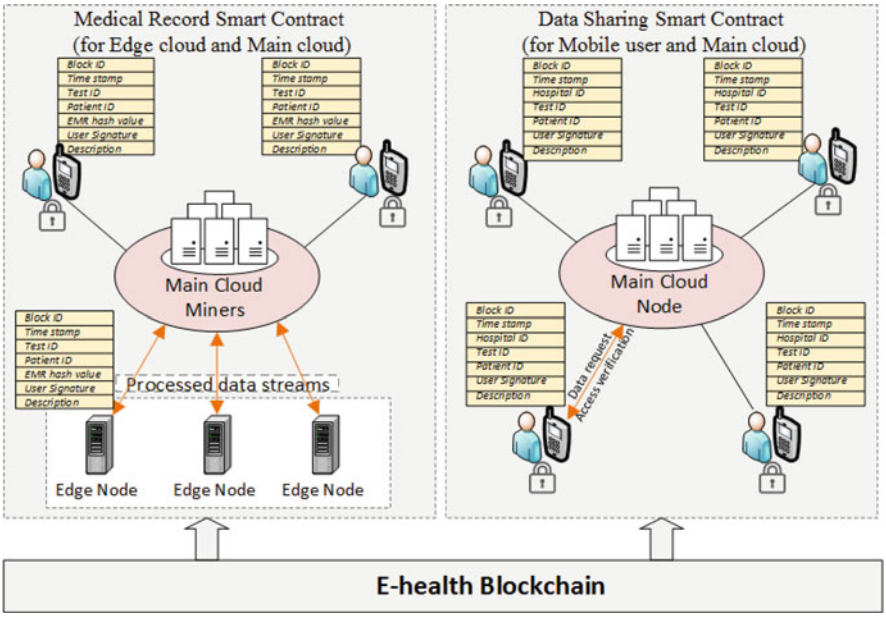


Fig. 3 Smart contracts for e-health blockchain

Table 1 Illustration of MRSC policy

Gateway public key	GatewayID	Action	Timestamp
046ELN134F	G001	Write	2018-10-15
94DJMFKJ88	G002	Deny	2018-10-16
...	...	...	...

- *deleteUser(PK, userRole):* (executed by Admin) It is used to remove users from the MRSC contract.
- *policyList(PK):* (executed by Admin) A pair of health provider-patient can agree on a policy which expresses their preferences in relation to medical services. For example, a patient has a specific doctor and only wishes to provide this doctor access to his/her EMRs. The policy list contains public key of all entities for identification when the smart contract processes new transactions.
- *writeEMRs(Addr):* (executed by EMRs manager) This enables the storing of medical data, such as the treatment the patient underwent, in the cloud storage. Medical data will be inserted into a corresponding storage node using patient address information (including hospital ID, medical test ID, and patient ID).

The type of information that will constitute a MRSC policy is illustrated in Table 1.

- ***Data block:*** Once the storage request is accepted by MRSC, IoMT data can be uploaded to ECs for processing. The processed EMRs are stored and hashed and added to the EMR chain. The EMRs added to the EMR chain can be accessed

by users. Hashing medical data brings two key benefits. First, it reduces the amount of data that needs to be stored on EMR chain, improving latency and communication cost. Second, data privacy is improved, as only the hash values of data are published on the EMR chain.

3.3 Data Sharing Smart Contract (DSSC)

DSSC comprises the following main components.

- ***Request metadata:*** To request data on cloud, users need to generate a request for information which includes the relevant metadata, namely, patient address, digital signature, and user public key. The request is verified by the CSSC using the information contained in CSSC policy. The type of information that will constitute a MRSC policy is illustrated in Table 2. If the request is successfully verified, the cloud storage will return the relevant data to the requester.

  CDSC also offer the following functions:

  – *addUser(), deleteUser(), policyList():* similar to functions of MRSC.
  – *retrieveEMRs(PK, Addr)* (executed by EMRs manager): It allows to retrieve cloud EMRs of patients. A user needs to provide the address information of patient to the smart contract. The contract then verifies and sends a message to EMRs manager to extract and return data to the requester.
  – *Penalty(PK, action)* (executed by Admin): When detecting an unauthorized request to EMRs system, the EMRs manager will inform the smart contract to issue a penalty (action) to the requester. In our work, DSSC gives a warning message as a penalty to the unauthorized entity.

4 EMRs Storage and Data Sharing Frameworks Using Smart Contracts

Based on the smart contract designs described in the above section, we present a EMRs storage scheme using MRSC and a medical data sharing scheme using DSSC in this section.

Table 2 Illustration of CDSC policy

User public key	User role	Action	Timestamp
007AE9704D	Doctor	Retrieve	2018-05-07
98IF46HHEA	Nurse	Write	2018-16-02
76ABw4L721	Patient	Deny	2018-09-03
...	...	...	...

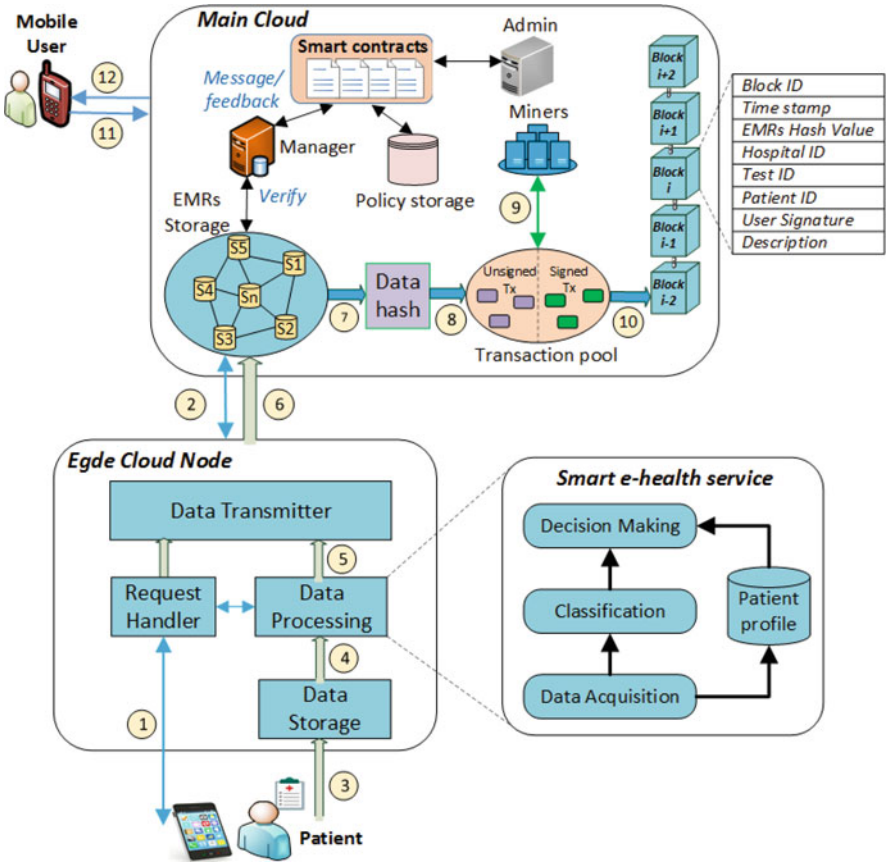


Fig. 4 Secure data storage and sharing schemes on e-health blockchain

The work flow of EMRs storage and data sharing processes is illustrated in Fig. 4.

1. A gateway sends a request to EC.
2. The EC forwards the request to EMRs manager for verification. The verification is done using smart contracts as described earlier. If the request is successfully verified, a response is sent back to the EC requesting the uploading of the data together with a patient reference.
3. The EC collects the data from the IoMT.
4. Raw IoMT data stored and processed by the EC to create or update an EMR for the patient.
5. The EC then transmits the EMRs to the EMRs admin within the cloud which stores the data in a storage node based on patient reference provided in Step 2.
6. Processed data is hashed for data privacy.
7. Data blocks are inserted into the pool for confirmation by miners.

8. Miners gather blocks including user transactions to verify and grant signature for blocks.
9. Verified blocks are appended to EMRs chain and can be accessed by users of the system by sending request to access the data.
10. If the access request is validated, cloud returns the requested data. Otherwise, the smart contract issues a warning message to the requester for illegal access.

4.1 EMR Storage Using MRSC

The EMR storage scheme using MRSC is shown in the following steps.
Step 1: Initialization (executed by the gateway)
The gateway which manages IoMT devices creates an account to join the e-health system. This done by the gateway sending a request, storage transaction *Ts* to the EMRs manager, via the EC. *Ts* will comprise an information index including a (*Gateway ID*), a digital signature, and a timestamp that can be used for verification, as well as a patient index for storage (Step 1 in Fig. 4):

$$Ts = (requestIndex||Sign_{Ts}||patientIndex||timestamp)$$

where

$$requestIndex =< GatewayID >$$
$$Sign_{Ts} = Sign_{SK}(requestIndex)$$
$$patientIndex =< HospID, TestID, PID >$$

Step 2: Verification for storage request (executed by the EMRs manager)
Upon receiving a *Ts* from a gateway, the EMRs manager will issue a signal to MRSC as a notification of a new data upload. By using the policy list, MRSC verifies the validity of *Ts* and, if it is valid, returns a message to gateway that generated the *Ts*, authorizing the uploading of data (Step 2 in Fig. 4).
Step 3: Uploading EMRs (executed by the gateway) The gateway now can upload EMRs to the EC for processing as illustrated in Step 3 in Fig. 4. EC will analyze and process raw medical data that can be in different medical data formats such as text, video, audio, or multimedia records.
Step 4: Processing EMRs for smart e-health services (executed by the EC) Raw EMRs are processed by EC Step 4 in Fig. 4. The processing will extract, analyze data features, classify, and make predictions using advanced artificial intelligent techniques. The process continuously monitors data updates from the gateways. The resulting EMR is then written to main cloud for storage by the *writeEMRs()* function of MRSC contract (Steps 5–6 in Fig. 4). EMRs manager will identify the patient address via *patientIndex* to determine the storage node.
Step 5: Adding the storage transaction to blockchain (executed by miners) Besides adding to cloud storage, data is hashed by a hash function before inserting into the unconfirmed transaction pool. The miners will periodically get transactions from the pool for mining. The storage transaction can be formulated as follows:.

$$Block = (Record||blockHash||Sig_{Record}||timestamp)$$

where

$$Record = (requestIndex||hashvalue||timestamp)$$
$$Sig_{Record} = Sign_{SK}(Record||timestamp)$$

The miner which verifies the data block first will send the signature to other miners for validation. If all miners achieve an agreement, the validated block with its signature is appended to the blockchain in chronological order.

$$dataBlock = (block||Sign_{block}||hash||timestamp)$$

where

$$Sign_{block} = Sign_{SK}(block||timestamp)$$

The work flow for the above process is shown from Step 8 to Step 10 in Fig. 4. This any participating user can monitor and control medical data in a distributed manner. More importantly, due to storing hash values instead of raw data on blockchain, the proposed model avoids risks of data leakage and thus ensures high data privacy. The data storage protocol is presented in Algorithm 6.

4.2 Data Sharing Using DSSC

The data sharing scheme using DSSC is designed to allow users to access data stored on the main cloud layer. The process is explained in the following steps:
Step 1: Generation of request information (executed by user) To get access to the data, users' need to create accounts. Creation of an account requires the use of a private key to sign the transaction and public key for user identification. To request data, the user also needs to provide the address of information. This is done by generating a user transaction *Tu*. This process is presented in Step 11 in Fig. 4. The user transaction can be formulated as below.

$$T_u = (PK||requestIndex||Sign_{Tu}||timestamp)$$

where

$$requestIndex =< HospID, TestID, PID >$$
$$Sign_{Tu} = Sign_{SK}(requestIndex)$$

Step 2: Data access verification (executed by the DSSC)
The *Tu* is sent to the EMRs manager for verification. This is achieved by sending a message to the DSSC. Based on the DSSC policy list which includes user public keys, the DSCC smart contract will confirm the validity of the request based on whether requesting user's public key is registered. If it is not registered, the DSSC will issue a penalty to the requester and deny any further requests.
Step 3: EMRs retrieval (executed by the DSSC)

Algorithm 6 Data storage protocol

Input: $Ts, EMRs$
Output: $data, block$
Require: MRSC policy list $MRSCpolicies$
Initialization: *(by the gateway)*
1: Initialize a storage request
$requestIndex =< GatewayID, DeviceID >$
$patientIndex =< HospID, TestID, PID >$
2: Send a storage request Ts
$Ts = (PK||requestIndex||patientIndex||timestamp)$
Verification: *(by the EMRs manager and MRSC contract)*
3: Receive a new request Ts from a gateway
4: Obtain the public key of the gateway
$PK \leftarrow Ts.getSenderPublicKey()$
5: **if** PK is available in the policy list
$MRSCpolicies[PK] \leftarrow true$
6: **end**
7: Decode the transaction
$decodedTs \leftarrow abiDecoder.decodeMethod(Ts)$
8: Specify request information
$Addr \leftarrow web3.eth.getData(decodedTs([DataIndex])$
9: Specify $requestIndex$
$RID \leftarrow Addr(Index[RequestIndex])$
10: Calculate the severity score using SVM
$EMRsResult \leftarrow SVM(EMRs)$
11: Hash data
$hashValue \leftarrow hashlib.sha1.hexdigest(EMRsResult)$
Writing *data* to storage: *(by the MRSC contract)*
12: $data \leftarrow (EMRsResult, hashValue)$
13: $writeEMRs(PK, patientIndex, data) \leftarrow true$
Adding *Ts* to blockchain: *(by miners, algorithm 1)*
14: Create new block *block* for Ts with $hashValue$ and RID
15: Run mining process to add *block* to blockchain

Once a request right is approved, the EMRs manager will locate the requested data using request index *requestIndex* that includes storage address of patient data. The EMRs manager can locate and extract data using the *retrieveEMRs()* function of the DSSC. If the information of request index is incorrect, the EMRs manager will issue a warning message to the requester and inform to the *penalty* function as a misbehavior.

Step 4: Data feedback (executed by the EMRs manager)
After the EMRs manager acquires the requested data from cloud storage, this data is returned to the requester (Step 12 in Fig. 4) and the user transaction is recorded. The process of adding the user transaction to blockchain is similar to what is described in *Step 5* of the data storage scheme. The data in transactions are patient addresses, which are lightweight and efficient to store on the blockchain. This ensures low network latency and improves data privacy of our e-health system. The data sharing protocol is shown in Algorithm 7.

Algorithm 7 Data sharing protocol

Input: Tu
Output: $Result$
Require: $penaltyCheck \leftarrow false$
DSSC policy list $DSSCpolicies$
Initialization: *(by the EMRs Manager)*
1: Receive a new transaction Tu from a user
2: Get the public key of the requester
$PK \leftarrow Tu.getSenderPublicKey()$
3: Send the public key to Admin
$(msg.sender = EMRsManager)$
Verification *(by the DSSC contract and Admin)*
4: **if** PK is available in the policy list
$DSSCpolicies[PK] \leftarrow true$
5: **end**
6: Decode the transaction
$decodedTu \leftarrow abiDecoder.decodeMethod(Tu)$
7: Specify request information
$Addr \leftarrow web3.eth.getData(decodedTu([DataIndex])$
8: Specify *testID* and *PatientID*
$test_{ID} \leftarrow Addr(Index[test_{ID}])$
$P_{ID} \leftarrow Addr(Index[P_{ID}])$
EMRs Retrieval *(by the DSSC contract)*
9: **while** true **do**
10: **if** $DSSCpolicies[test_{ID}] \rightarrow true$ **then**
11: **if** $DSSCpolicies[P_{ID}] \rightarrow true$ **then**
12: $Result \leftarrow retrieveEMRs(PK, Addr)$
13: break;
14: **else**
15: $penaltyCheck \leftarrow true$
16: $Result \leftarrow warningMessage$
17: break;
18: **else**
19: $penaltyCheck \leftarrow true$
20: $Result \leftarrow warningMessage$
21: break;
22: **end**
23: **end**
24: **return** $Result$;

To conclude, this section described design concepts using blockchain for data storing and sharing medical data in a distributed e-health system. It also demonstrated how smart contracts can be adopted to simplify the processing requirements. Finally, it showed how edge computing can be used to improve latency and efficiency of data retrieval and hashing can be used to augment the privacy of the data that is stored and shared. The next section presents an implementation of the proposed system that demonstrates feasibility of these mechanisms.

5 A Case Study for Health Data Management with Blockchain

We implement a case study for health data management using an Ethereum blockchain platform and cloud computing. An Ethereum blockchain includes smart contracts which contain data-associated code with multiple programmed functions. In our design, users can communicate directly with smart contracts through *Application Binary Interfaces (ABI)* to implement job functionalities such as data access, user verification, or network management.

Meanwhile, Ethereum transactions are first used to transfer ether (Ethereum native token) from an account to another. A typical structure of transaction is as follows *(Account Nonce, address of recipient, Gas price, Gas limit, Ether value, sender's signature, data field)*. In our design, we use Ethereum transactions with the *data field* to send user request information to cloud for EMR storage or data access. Here is an example of a transaction: *Transaction Tx = new Transaction (getFromAddress(), gasPrice, gasLimit, to, value, data)*.

5.1 System Setup

This section presents the proof-of-concept demonstrator that shows the viability of the model shown in Fig. 1 using a private Ethereum blockchain with dimensioned components on the Amazon cloud [25]. The system comprised two virtual machines, AWS EC2 were employed as the miners, a virtual machine Ubuntu 16.04 LTS was used as the EMRs admin and, another instance, AWS S3 as the cloud storage. AWS Lambda edge was used to realize the ECs closer to local IoMT devices. Each EC was configured to manage and process raw data streams of the gateways, and the data processed at the edge was stored in different AWS S3 storage instances. Each storage instance kept the medical data of a given type of medical test. Cloud storage was controlled by an AWS Lambda function, emulating the EMRs manager which listens to new access events or requests from users. The smart contracts are written in *Solidity* and deployed on AWS Lambda functions. Each function interacts with the cloud blockchain via the web3.js API.

Users interact using Android phone where a Geth client (a command line interface implemented in the Go language) is installed to transform each smartphone into an Ethereum node. By adopting the Geth client, a user can establish an Ethereum account to communicate with the blockchain network for accessing data. The web3.js library, a lightweight Java library for working with smart contracts and blockchain, is utilized to develop a mobile application for connecting with the Ethereum blockchain network. To interact with Ethereum smart contracts for

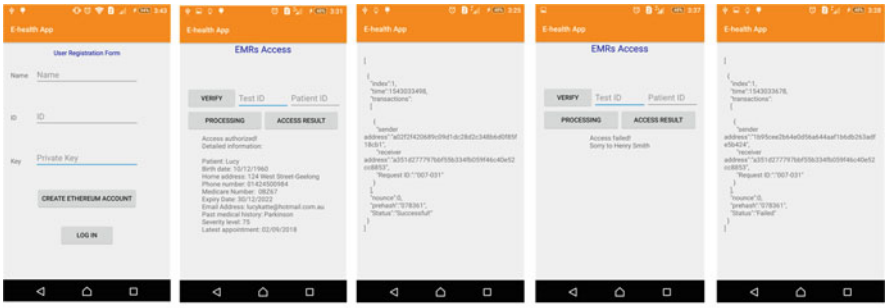
verification and data access, a decode function *abiDecoder.decodeMethod(Tx)* is used on cloud to obtain the request information in the *data field* of transaction.

In the local IoMTs network, we utilize a healthcare platform called *BioKin* [26], which consists of a network of wearable sensors to measure human motion data and a mobile Android application to collect and transmit EMRs to the ECs for processing and storage. In addition, a blockchain application is integrated in Android phones which enables them to act as both gateways and end entities.

In the next subsections, we present our implementation for data storage and data sharing frameworks on the blockchain using design concepts mentioned in Section III. Two smart contracts (MRSC and DSSC) will be employed to formulate our framework.

5.1.1 Data Storage with MRSC

In this implementation, we present a data storage protocol with an automated clinical service for diagnosing real-time human movement disorders. The objective of our scheme is to identify, analyze, and predict the severity of movement disorders, by a designed e-health model at the edge cloud layer. In this system, we propose to use a support vector machine (SVM) regression model to estimate the severity levels of movement disorders for patients. By using the health system with a programmed algorithm, healthcare providers can eliminate manual diagnostic processes and obtain automatically reliable clinical results on their smartphones as shown in Fig. 5. Processed data is kept in cloud storage, while records of data uploading process are added to blockchain for sharing. Its implementation is based on Algorithm 6 as presented in the above section.



(a) (b) (c) (d) (e)

Fig. 5 Illustrations of EMRs access results on Android phones: (**a**) User Registration form with Ethereum account, (**b**) data access results with patient severity score of an authorized user, (**c**) transaction record of authorized EMRs access, (**d**) data access result of an unauthorized user, (**e**) transaction record of unauthorized EMRs access

5.1.2 Data Sharing with DSSC

In this implementation, when a user requests data access to cloud storage on cloud, *Hospital ID* is ignored without loss of generality for simplicity. The protocol is enabled by DSSC contract, and its implementation is based on Algorithm 7 as presented in the above section.

5.2 Results and Evaluation

In this section, we present evaluation of the above implementation in terms of as an illustration of the following features::

- Data sharing: data access results on smartphones with two use cases: authorized and unauthorized access to investigate the feasibility of data sharing algorithms.
- Network latency: network latency in different test scenarios, firstly processing time on main cloud for the data access scheme and secondly network latency of the cloud network with the combination of edge cloud computing and data storage systems.
- Block generation: the time of the block generation process on blockchain for the off-chain and on-chain data storage.

Data Sharing We consider two use cases with authorized and unauthorized access. The results on mobile phones are shown in Fig. 5. In this test scenario, a user such as a physician who wants to access needs to create an account to request access rights through the mobile application (Fig. 5a). Once verified and access is granted, he can start to access e-health services. To obtain medical data of a given patient, the user needs to provide the patient address, including test ID and patient ID as shown in Fig. 5b.[2] If the request is validated, the system will return requested data, and accessing of the data is also updated (Fig. 5b). In our case, the user can obtain the severity degree of movement disorder so proper care services are provided for the patient. Now, as the EMRs access process has been completed, the transaction is added to the blockchain and is broadcast. This allows patients to track access to their EMRs (Fig. 5c).

Unauthorized access requests will be detected using the DSSC contract predefined policies and denied access (Fig. 5d). A penalty with a warning message also returned to the requester, and an unauthorized access transaction will be issued by the smart contract (Fig. 5e).

Network Latency We measure the average time consumption by the cloud system for processing access requests simultaneously (Fig. 6a). The mechanism with trust

[2] It is assumed that there is only a medical institution in our test scenario so hospital ID is not required.

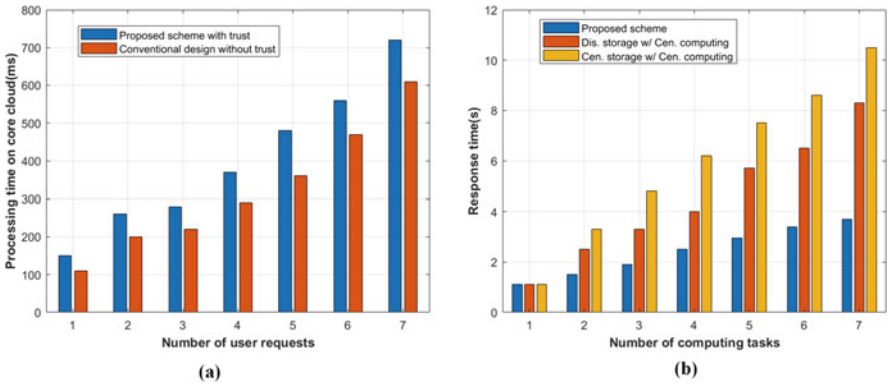


Fig. 6 Time overheads: (**a**) Processing time on core (main) cloud versus user requests, (**b**) response time versus number of computing tasks

based on smart contract consumes more time to process user requests, compared with the non-trust scheme. This overhead comes from time consumed in identifying the user and access verification. However, in the worst case (seven requests), the additional overhead is approximately 100ms. This highlights efficiency of the user authentication and access control.

Response Time is the time taken from when a IoMT network uploads data to an end user getting access to the EMR associated with that data (Fig. 6b). In this test scenario, each EMR is processed by an EC, and processing a EMRs file is represented by a computing task. We measure time cost of many simultaneous computing tasks. Each computing task includes four stages, namely, request validation, EMRs processing, data storage, and data hashing. To evaluate the performance, we compare it with two other conventional approaches: *distributed storage with centralized computing* and *centralized storage with centralized computing* without taking into account communication delay between user devices and cloud. The results indicate that there is a small decrease in performance when the number of computing tasks increases. This decrease in the performance is directly associated with waiting time for further processing tasks on main cloud. In contrast, the conventional schemes show very high time overheads.

Block Generation refers to the time consumed by miners to validate new blocks and append them to blockchain. This is computed using the Kinesis Data Analytics service provided by and is shown in (Fig. 7), using a minimum difficulty level for mining to avoid additional delay in our test evaluation.[3] We hash data using a hash function *SHA-1*, which produces a 20-byte hash value from any data inputs, before adding to blocks on blockchain for sharing. This dramatically reduces the block

[3] The difficulty level should be chosen to ensure the level of security required for blockchain network.

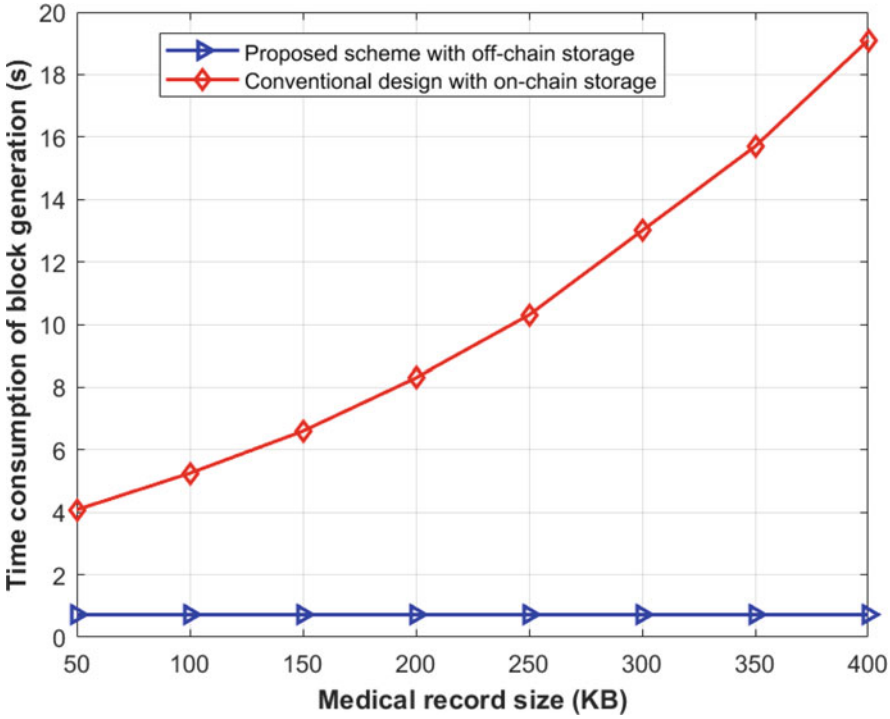


Fig. 7 Time overhead of block generation on cloud blockchain

sizes on the e-health blockchain. From results in the Fig. 4, it can be seen that the overhead for block generation is very small. If the data is stored on the chain, the block generation time will be proportional to the block size, since the larger the file, the more time will be consumed to transmit on blockchain.

6 Conclusion

In this chapter, we provided an overview of how blockchains and smart contracts can be used to provide secure and private data storage and access system. We described how such a system can be used to develop an e-health system where patient data can be securely stored and shared among health professionals. The viability of the proposed system was demonstrated through a proof-of-concept implementation.

While the proof-of-concept implementation was rudimentary, it showed that blockchain and smart contracts can be used to provide secure and efficient data storage and distribution systems.

References

1. E.J. De Aguiar, B.S. Faiçal, B. Krishnamachari, J. Ueyama, A survey of blockchain-based strategies for healthcare. ACM Comput. Surveys **53**(2), 1–27 (2020)
2. T. Hardin, D. Kotz, Blockchain in healthcare data systems: A survey, in *Proceedings of the International Conference on Internet of Things: Systems, Management and Security (IOTSMS)* (IEEE, Piscataway, 2019), pp. 490–497. [Online]. Available: https://www.cs.dartmouth.edu/~kotz/research/hardin-blockchain-survey/index.html
3. D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, integration of blockchain and cloud of things: architecture, applications and challenges. IEEE Commun. Surveys Tutorials **22**(4), 2521–2549 (2020)
4. Y. Ma, Y. Wang, J. Yang, Y. Miao, W. Li, Big health application system based on health internet of things and big data. IEEE Access **5**, 7885–7897 (2017)
5. Y. Gong, P. Rimba, R. Sinnott, A big data architecture for near real-time traffic analytics, in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (2017), pp. 157–162
6. L. Andrioli, R. Righi, C. André da Costa, L. Graebin, Observing network performance and congestion on managing assets with rfid and cloud computing. J. Comput. Commun. **05**, 43–66 (2017)
7. K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research. IEEE Access **PP**, 1–1 (2020)
8. H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, Y. Zhang, Mobile edge cloud system: architectures, challenges, and approaches. IEEE Syst. J. **12**(3), 2495–2508 (2018)
9. D.C. Nguyen, P. Cheng, M. Ding, D. Lopez-Perez, P.N. Pathirana, J. Li, A. Seneviratne, Y. Li, H.V. Poor, Enabling AI in future wireless networks: A data life cycle perspective, in *IEEE Communications Surveys and Tutorials* (2020), pp. 1–1
10. S.R. Chaudhry, A. Palade, A. Kazmi, S. Clarke, Improved QoS at the edge using serverless computing to deploy virtual network functions. IEEE Int. Things J. **7**(10), 10673–10683 (2020)
11. S. Noghabi, L. Cox, S. Agarwal, G. Ananthanarayanan, The emerging landscape of edge-computing, in *ACM SIGMOBILE GetMobile* (2020). [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-emerging-landscape-of-edge-computing/
12. M.A. Rahman, M.S. Hossain, G. Loukas, E. Hassanain, S.S. Rahman, M.F. Alhamid, M. Guizani, Blockchain-based mobile edge computing framework for secure therapy applications. IEEE Access **6**, 72469–72478 (2018)
13. K.R. Choo, O.F. Rana, M. Rajarajan, Cloud security engineering: theory, practice and future research. IEEE Trans. Cloud Comput. **5**(3), 372–374 (2017)
14. J. Li, D. Lin, A.C. Squicciarini, J. Li, C. Jia, Towards privacy-preserving storage and retrieval in multiple clouds. IEEE Trans. Cloud Comput. **5**(3), 499–509 (2017)
15. A.A. Abdellatif, A.Z. Al-Marridi, A. Mohamed, A. Erbad, C.F. Chiasserini, A. Refaey, ssHealth: toward secure, blockchain-enabled healthcare systems. IEEE Netw. **34**, 312–319 (2020)
16. M.A. Rahman, E. Hassanain, M.M. Rashid, S.J. Barnes, M.S. Hossain, Spatial blockchain-based secure mass screening framework for children with dyslexia. IEEE Access **6**, 61876–61885 (2018)
17. D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning. IEEE Trans. Netw. Serv. Manag. **17**(4), 2536–2549 (2020)
18. O. Attia, I. Khoufi, A. Laouiti, C. Adjih, An IoT-blockchain architecture based on hyperledger framework for healthcare monitoring application, in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (2019), pp. 1–5
19. R. Zhang, R. Xue, L. Liu, Security and privacy on blockchain. ACM Comput. Sur. **52**, 1–34 (2019)

20. D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Blockchain for 5G and beyond networks: a state of the art survey. J. Netw. Comput. Appl. **166**, 102693 (2020)
21. S. Rouhani, R. Deters, Security, performance, and applications of smart contracts: a systematic survey. IEEE Access **7**, 50759–50779 (2019)
22. Y. Hu, M. Liyanage, A. Mansoor, K. Thilakarathna, G. Jourjon, A. Seneviratne, Blockchain-based smart contracts-applications and challenges (2018). https://link.springer.com/article/10.1007/s12083-021-01127-0
23. S. Jiang, H. Wu, L. Wang, Patients-controlled secure and privacy-preserving EHRs sharing scheme based on consortium blockchain, in *IEEE Global Communications Conference (GLOBECOM)*, vol. 2019(2019), pp. 1–6
24. D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Blockchain for secure EHRs sharing of mobile cloud based e-health systems. IEEE Access **7**, 66792–66806 (2019)
25. D.C. Nguyen, K.D. Nguyen, P.N. Pathirana, A mobile cloud based IoMT framework for automated health assessment and management, in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2019), pp. 6517–6520. ISSN: 1558-4615
26. S.W. Ekanayake, A.J. Morris, M. Forrester, P.N. Pathirana, Biokin: an ambulatory platform for gait kinematic and feature assessment. Healthcare Technol. Lett. **2**(1), 40–45 (2015)

Supporting Secure Trusted Manufacturing via Blockchain

Ali Dorri, Sabah Suhail, Zahra Jadidi, Rasheed Hussain, Colin Fidge, and Raja Jurdak

1 Introduction

The convergence of Internet of Things (IoTs) technology with the manufacturing industry leads to the introduction of *smart manufacturing*. In the latter, a range of devices in different stages of the manufacturing process communicate and exchange data. This in turn reduces monetary costs and delays as well as facilitates the manufacturing process. However, the device's high connectivity introduces security risks because malicious nodes may access and/or control the devices remotely. Malicious nodes may also tamper with the data generated by the devices or inject fake data into the network that is later used by central controllers to make decisions. This in turn enables the malicious nodes to modify the decisions made by the controller which leads to monetary loss or even interruptions to the required functionality of the manufacturing site. These phenomena make security and data trust, i.e., trust in the data that is generated by IoT devices, highly challenging in smart manufacturing.

A. Dorri (✉) · Z. Jadidi · C. Fidge · R. Jurdak
School of Computer Science, Queensland University of Technology (QUT), Brisbane, QLD, Australia
e-mail: ali.dorri@qut.edu.au; z.jadidi@griffith.edu.au; c.fidge@qut.edu.au; r.jurdak@qut.edu.au

S. Suhail
Research Institute for Cryptoeconomics, Vienna University of Economics and Business, Vienna, Austria
e-mail: sabah.suhail@wu.ac.at

R. Hussain
Smart Internet Lab and Bristol Digital Futures Institute (BDFI), University of Bristol, Bristol, United Kingdom
e-mail: rasheed.hussain@bristol.ac.uk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_19

One of the fundamental methods proposed in the literature to improve the security of smart manufacturing is to detect anomalies by analyzing the log files produced by the devices. Log files contain detailed information about recent events on each device. A device either stores these event records locally or on a remote server. Due to the value of logged information for identifying attacks, malicious nodes may attempt to remove or modify log files after launching a successful attack. Also, storing log files on a remote server introduces new vulnerabilities including log modification or loss during transmission, compromises of the central server to remove log files, and a single point of failure for data storage. The log files may also be removed unintentionally by the organization's staff. These issues highlight the need for decentralized secure log management solutions. In recent years blockchain technology has received notable attention as a solution to address the aforementioned challenges. Blockchain is an immutable database which makes it highly challenging to remove or tamper with data already stored. A blockchain's distributed management also eliminates the single point of failure.

Another challenge is that the data produced by a device cannot always be trusted given that devices may be compromised by an attacker or the devices may capture information with different levels of accuracy. Similarly, devices may malfunction as a result of which erroneous data is produced. This makes data trust highly challenging in smart manufacturing. Therefore, once the data collected from trustworthy sources are recorded in the blockchain, product lifecycle events can be fed into data-driven systems such as Digital Twins (DTs) for anomaly detection, behavior analysis, and optimized control. Thus, integrating blockchain in DTs can support reliable industrial processes where blockchain ensures secure data management and DTs use trustworthy data as input to extract actionable insights for predictive maintenance.

In this chapter, we study these security and data trust challenges in smart manufacturing. First, we discuss distributed anomaly detection methods to secure smart manufacturing by analyzing the logs of the devices. The authenticity of the log information provided by the devices remains a critical challenge as it has direct impact on the security. Thus, we next discuss trust challenges associated with the data and elaborate on Digital Twins (DTs) as possible solution. We describe two fundamental methods for increasing the security of manufacturing. To do so, the methods employ blockchain technology to store data. The data transmitted by the blockchain is then analyzed by particular nodes which use machine learning algorithms to detect anomalies. To reduce the delay in detecting attacks, two levels of learning are used where the data is first analyzed at the site level and then at the organization level.

To maximize industrial revenues in terms of cost management, risk prognosis, and quality assurance, a digital fingerprint (i.e., a DT) of the underlying process can be created where all operations must be analyzed, predicted, and optimized before the process's real-world implementation. DTs run synchronously with physical counterparts to track data anomalies or operational disruptions between the physical space and the virtual space. Consequently, inconsistencies between the two spaces entail the adoption of better calibration and testing strategies that evolve DT models

and physical counterparts to support more accurate optimization and prediction of the processes. The data fed into the DTs accounts for rational decision-making. Therefore, to ensure the trustworthiness of data collated from disparate data sources, a blockchain can be integrated to solve issues related to product lifecycle data management and data security.

2 Background

This section outlines the necessary background information for this chapter. We first discuss anomaly detection based on device logs followed by a discussion on Digital Twins (DTs) and their role in ensuring actionable insights for predictive maintenance in the manufacturing industry.

2.1 Analyzing Logs to Detect Anomalies

Analyzing devices' log files to detect malicious activities, i.e., anomalies, is among the fundamental methods proposed in the literature to secure manufacturing. The process involves two phases, namely, (i) secure log management and (ii) anomaly detection. We discuss these phases below.

Secure Log Management After each significant event they perform, IoT devices generate a record in a log file that contains details of the event including the time, action, variable changes, IP addresses, etc. The device may store the log file locally or on a remote server, e.g., a log server that stores the log files of all devices. Blockchain-based log management has received tremendous attention to address the challenges outlined above due to its salient features, including distributed management and immutability. The distributed nature of a blockchain eliminates centralization and provides a built-in backup feature for the log files. The immutability of the blockchain makes it impossible to modify or remove data previously stored in the blockchain.

Pourmajidi and Miranskyy [10] proposed a *logchain* to securely store and transfer log files using a blockchain. To increase the blockchain's throughput, i.e., the total number of transactions that can be stored in the blockchain per second, a logchain is a *circulated chain*. A group of participants jointly form a circulated chain to store their log files. It has a fixed depth, i.e., the total number of blocks in the chain. The last block in the ledger contains a pointer to the genesis block, i.e., the first block, which creates a circle. This circular blockchain ensures that once the last block is stored, no further blocks can be appended to the ledger, thus preventing malicious nodes from storing fake log files. Once a circle is formed, the corresponding hash is stored in the main chain to ensure the integrity of the data stored in the circulated chain.

Taguchi et al. [14] proposed a blockchain-based log management framework. Their devices send log data to a group of predefined nodes that function as gateways that connect the end devices to the blockchain network. The nodes aggregate the received log files in a particular period of time and form them into a single block. The aggregators then broadcast the block to the blockchain network. Lee et al. [8] proposed a blockchain-based solution to store the logs of removable devices. The removable devices immediately send their log files to the blockchain which ensures the log will be stored on the chain even if the device leaves the network.

Anomaly Detection Particular nodes, referred to as learners, in the network run machine learning algorithms on top of the log files to detect anomalies. Feng et al. [1] proposed combining the contents of the log files transmitted between the nodes and the corresponding time-series to detect anomalies. Inoue et al. [6] compared the performance of deep neural networks (DNNs) and one-class support vector machines (SVMs) for detecting anomalies in a water management system. Their results confirm that although both methods are able to detect anomalies, the false-positive rate was lower in DNN-based algorithms.

Most existing anomaly detection algorithms assume that the log data are trusted and that transmission is secure. However, as outlined earlier, in a real-world scenario, malicious nodes may tamper with the data or the devices. The existing solutions also rely on a centralized model where a central server collects data from the devices and processes them to detect anomalies. However, such centralized methods suffer from being a single point of failure, increasing the computational overhead on the server to process the received logs, having high delays in detecting attacks, and increasing the communication overhead as raw log files need to be transferred to the learner. To address these limitations, in Sect. 3 we present a comprehensive solution to detect anomalies while ensuring the security of the log files.

2.2 Digital Twins

The concept of Digital Twins was informally introduced by Micheal Grieves in 2003 during his course on "Product Lifecycle Management (PLM)." In 2012, the seminal work by the National Aeronautics and Space Administration (NASA) in the field of aviation defined the DT as an integrated multi-physics, multi-scale, probabilistic simulation mirroring the physical system based on models, sensors, history data, etc. [4]. In 2014, Grieves' white paper [5] proposed a preliminary form of DTs that include three key elements: (i) a physical space containing a physical entity; (ii) a virtual space containing a virtual entity; and (iii) the connection between both spaces (as shown in Fig. 1). Later on, the numerous practical applications of DTs beyond the aerospace industry further promoted their development. In parallel, DTs were seen as increasingly attractive, thanks to continuous advances in their enabling technologies. For instance, IoT, big data processing techniques, cloud computing,

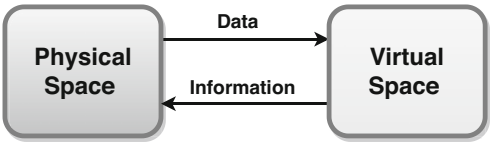


Fig. 1 Adaption of Grieves' DT model [5]

and artificial intelligence (AI) further motivate DTs. According to Gartner's survey report, 13% of enterprises are already using DTs, whereas 62% are in the process of deploying DTs [3].

The industrial applications of data-driven DTs focus on the areas of product design, production, Prognostic Health Management (PHM), etc. For instance, DTs can be used to design and produce products in more responsive and predictable way. PHM plays a significant role in the lifecycle monitoring of equipment conditions where a DT of the equipment along with data is used to depict the behavior of the physical entity, perform diagnosis, and provide design rules for maintenance. Leading companies including Siemens, General Electric (GE), and IBM have already deployed DTs in various industrial applications. For instance, Siemens' plant simulations can optimize production line scheduling and factory layout. Similarly, IBM leverages DTs for automatic vehicles to analyze engine speed, oil pressure, and other critical parameters.

DT data also empowers the use of Augmented Reality (AR) and Virtual Reality (VR). AR and VR provide rapid prototyping and facilitate visualizing the configurations of modules before their actual implementations in industrial systems. For example, AR glasses enable a worker to highlight the location where each part should be mounted in the assembly process and train plant personnel to handle emergencies [2], allowing operators to learn to interact with machines, change parameters, retrieve operational data, and generate insights that can be used to offer a range of new services and to optimize the design. Similarly, other use-cases include predictive maintenance to detect failures and behavior of machines under various circumstances, automated optical inspection to identify quality defects, inventory management for supply chain optimization, etc. The integration of DTs in industrial applications can help to carry out what-if analyses to minimize unplanned downtime, enable remote troubleshooting, lower maintenance costs, and increase a product's lifespan.

3 Enforcing Security and Trust in Manufacturing Using Blockchain

In this section, we present two architectures that address security and trust challenges in the manufacturing industry. We describe secure log management and distributed anomaly detection in Sect. 3.1 and trustworthy DTs through a blockchain-based framework in Sect. 3.2.

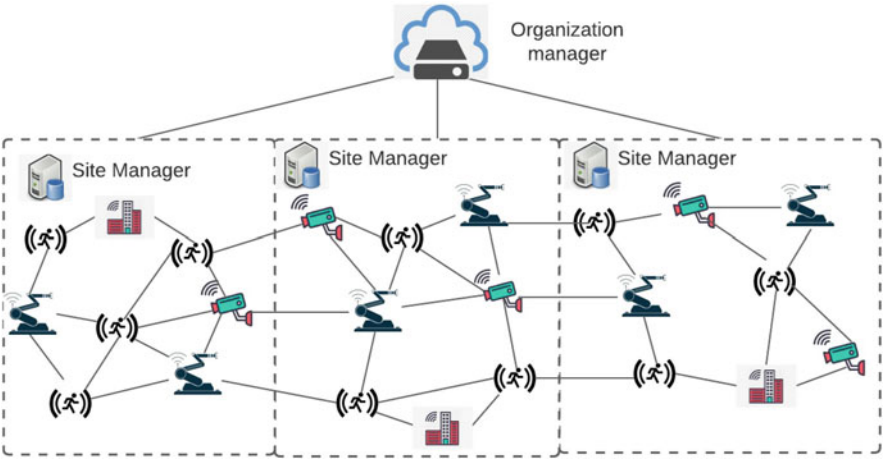


Fig. 2 A high-level overview of the outlined solution

3.1 Secure Manufacturing Using Blockchain

In this section, we outline the details of a comprehensive solution to detect anomalies as shown in Fig. 2 [7]. The introduced architecture comprises two main levels which are site and organization levels. In the site level, the devices generate log information and share in a private blockchain to ensure immutability and protect against log tampering/removing attacks. In the organization level, the logs in different sites are analyzed to detect attacks and anomalies. The manufacturers manage a private blockchain where they authorize nodes to join the blockchain which in turn protects against malicious nodes accessing the log data. Once a device is authorized to join the blockchain, the authorities generate a genesis transaction for the device that is structured as a tuple $\langle T_ID, PK_{\mathrm{a}}, Sign_{\mathrm{a}}, PK_{\mathrm{d}} \rangle$ where *T_ID* is the unique identifier of the transaction which is essentially the hash of the transaction content, PK_{a} and $Sign_{\mathrm{a}}$ are the Public Key (PK) and signature of the authority, and PK_{d} is the PK of the device that will be used in future transactions.

When an event happens, the corresponding device generates a log register transaction (t_{lr}) that is structured as $\langle T_ID, P_T_ID, PK, data, Sign \rangle$ where *P_T_ID* is the identity of the previous transaction generated by the same device, the *PK* and *Sign* are the PK and signature of the device, and *data* is the log information. The data is encrypted with the PK of the authorities to ensure only the relevant parties can read the log information and thus protect the privacy of the company and confidentiality of the data.

To reduce the delay associated with detecting anomalies and reduce the overhead on the central server, the framework introduces two learning levels which are the site and organization levels. In the former, the site manager processes the data

of the devices installed in the associated site to detect any malicious activities locally. Given that the data comes from multiple sources, the framework employs a multi-source deep neural network (MS-DNN) algorithm [9] that uses sequence classification to detect anomalies. The MS-DNN ensures that the input data have the same format by pre-processing the data. The MS-DNN output is the total number of devices whose corresponding logs are analyzed. Training of the deep neural network is conducted by employing a bidirectional Long Short-Term Memory (LSTM) network.

In the case where malicious activity is detected, the site manager informs the rest of the network by generating an *alert transaction* (t_{a}) that is structured as $\langle T_ID, P_T_ID, Data, PK, Sign\rangle$ where *Data* contains the information related to the attack that will be employed by other nodes to identify the attack. This can be the PKs used by the attacker, the attack behavior, the type of the attacked devices, etc. The site level anomaly detection reduces the delay in detecting malicious activities as the data is processed at the edge of the network. However, the data used in the site level is limited to the data produced by the devices at that site. Cyber attacks may be more advanced and complicated and involve devices in various sites which makes them undetectable at the site level. Thus, the framework also introduces organization level anomaly detection.

At the organization level, the organization manager processes the data of devices in all sites to detect anomalies. When it detects an anomaly, the organization manager broadcasts a transaction t_a, as outlined above, to inform other nodes.

In summary, the framework ensures the security of the manufacturing by storing and transferring log files using blockchain and by analyzing the logs in two layers to detect anomalies. As outlined earlier, the anomaly detection algorithm employs the log information to detect anomalies; thus, it is critical to ensure trustworthiness of the log information. In other words, it is fundamental to ensure that the log information is not modified before being stored in the blockchain. In the next section, we discuss digital twins as a potential solution to address this limitation and thus enhance the security of smart manufacturing.

3.2 Trustworthy Digital Twins Using Blockchain

Curating and mining intelligent information from the collected data call for a rational infrastructure that can manifest the trustworthiness of (i) data sources, (ii) data in transit, and (iii) data at rest (storage). For data trustworthiness, *provenance* is considered to be an effective mechanism that facilitates *track* and *trace* solutions during data acquisition from data-generating sources and data transmission [11]. Thus provenance data solves the risk of invalid or misleading data by providing measures to identify device tampering, diagnose data forgery, and revoke a rogue participating entity that may generate or propagate erroneous data [12]. However, provenance management for IoT networks needs to address several challenges such

as low energy, bandwidth consumption, secure transmission, efficient storage, and data collation from disparate sources [11]. To sum up, the overarching questions are:

- How to ensure the trustworthiness of data-generating sources and data collected and collated from disparate data silos?
- How to extract actionable insights for the predictive maintenance in data-driven systems?

Suhail et al. [13] present a framework to address the challenges of data management, data security, and predictive maintenance. Firstly, their framework uses data provenance to ensure the trustworthiness of data as well as data-generating sources. Secondly, considering the challenging requirements of real-time industrial processes due to the connection between the physical and digital world, stringent security solutions, and the growing number of participating entities and to prepare for the quantum apocalypse, the framework focuses on the adoption of a lightweight, scalable, and quantum-immune blockchain-based solution for industrial use-cases. Thirdly, the framework integrates DTs to introduce security by design by addressing issues such as predictive maintenance, anomaly detection, and optimizing processes. The main components of their framework are presented in Fig. 3. The entities including sensors, machines, and humans initially register at the blockchain as authorized participating entities (Step 1). Next, at the *data layer*, asset-specific data items monitor, collect, and process designated parameters in the physical space (Step 2a). The resource-constrained IoT devices monitor and collect sensory data, whereas the gateways and fog-enabled Unmanned Aerial Vehicles (UAVs) perform power-intensive tasks, such as receiving requests from sensors, processing data, or broadcasting transactions in the network. The collected data along with provenance data is sent to the *storage layer* (Step 2b). Provenance data can render detailed information about actors (humans or machines) and actions performed by them and can be chained together to identify the root cause of disruptions and to attenuate risks in the system. The models are generated based on the collected sensor data, domain knowledge, processes and their documents, system history, and state data by the twinned system and are stored at the storage layer (Step 3). The *application layer* keeps on analyzing data to detect any of the manufacturing disturbances or potential security vulnerabilities (Step 4). In case of a malfunction, the respective scheduling services (Step 5a) in the physical space or model calibration services (Step 5b) in the virtual space are called, completing the feedback loop. The storage layer provides secure, distributed, and decentralized data access infrastructure through a blockchain such that the application layer uses it for analysis and decision-making, while DT uses it for generating and uploading virtual models. Additionally, since the repercussions of contingency events require immediate actions, direct continuous monitoring of such critical events through sensory data logged at the control unit is also enabled.

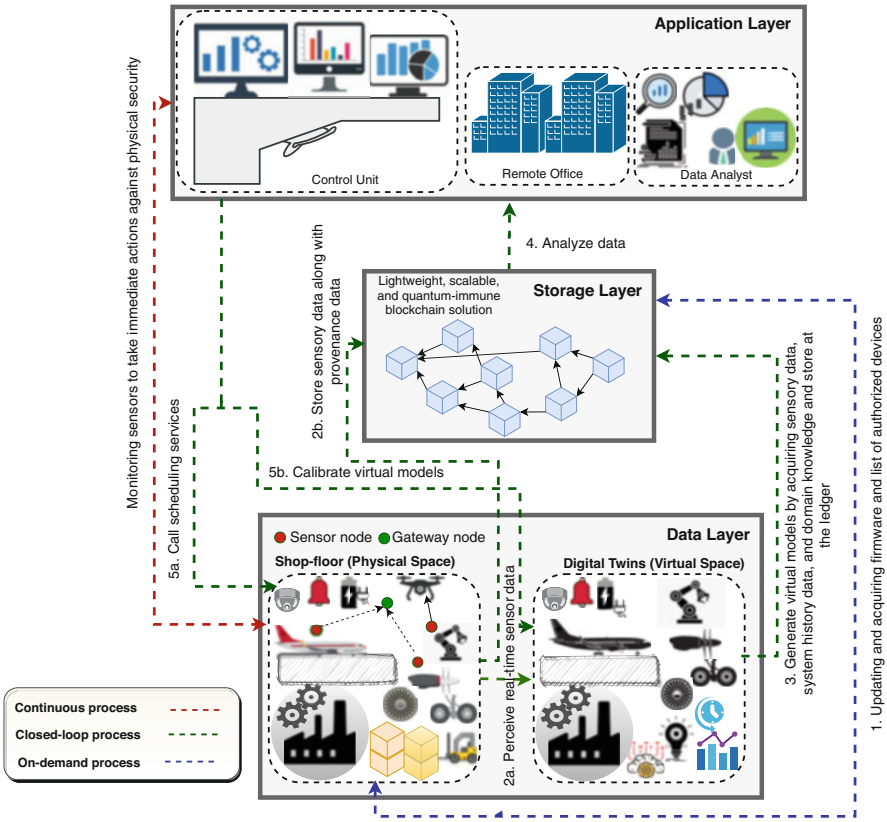


Fig. 3 Leveraging blockchain with digital twins: A framework [13]

Before initiating the production process (Fig. 4), regulatory endorsements can help to ascertain that supply chain processes are abiding by ethical practices and environmentally responsible operations. As suggested by Suhail et al. [12], exercising such Ethical, Sustainable, and Responsible (ESR) operations requires the involvement of regulatory bodies (NGOs, governments, industry self-regulators) to conduct a periodic on-site inspections of the facility and to provide verifiable certificates that are stored on the blockchain (Step 1 and Step 2). To initiate the supply chain trade event, the seller stores supply chain data (that may include transaction data, T_{Data}, and auxiliary data, A_{Data}) on the blockchain (Step 3), while the sensor data, S_{Data} (such as sensor IDs, and time-stamped sensor data), can be attached and updated at the blockchain (Step 4). Sensors are affixed to batches during logistics and transportation to provide information such as temperature, humidity, vibration, shock, location, etc., whereas the consignment information, Con_{info}, can be acquired by the seller or the buyer (Step 5a and 5b) during the trade events. When a product or its parts are received by the buying entity, a transaction receipt is generated to log the completion of the trade event between the seller-buyer

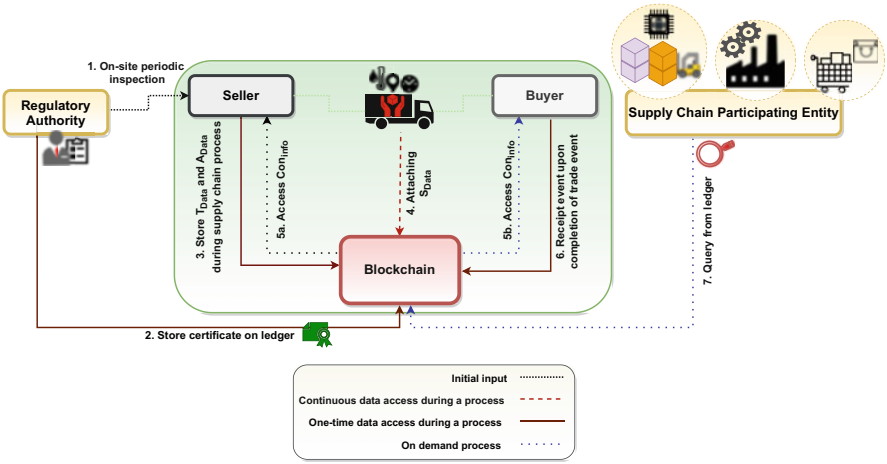


Fig. 4 Before production process: Data flow during a supply chain trade event

pair (Step 6). A twofold purpose of introducing this transaction is to circumvent fake-progressive sub-chains and to indicate loss or damage in an event. Finally, data from the blockchain can be used for ad hoc querying by any of the authorized supply chain participating entities (Step 7).

Thus the necessary details such as supply chain data (e.g., consignment information), order information (e.g., material stock, production quantity, estimated cost), simulation data (equipment historical data, prediction of equipment fault), Enterprise Information System (EIS) data (such as product lifecycle data, process document, market data), etc. are already available in the storage system for usage (Step 1, 1a, and 1b) as shown in Fig. 5. The data fusion method is used to combine them and generate consistent interpretation for a certain object. Based on the acquired data from Step 1, a set of predefined values (or specification data) describing the acceptable ranges of system performance parameters are also stored in the blockchain ledger (Step 2). Such ranges can serve as baseline and training data for normal instances. Also, it may act as an effective countermeasure against cyber attacks and Advanced Persistent Threats (APTs) as it is harnessed on the deep knowledge of system behavior and historical data analysis. Thus during the production process, monitoring the state through time, outliers, and changes can help in detecting anomalies or malicious activities. Furthermore, analyzing the relationships between the dynamic variables and historical variables based on thresholds and consistency checks enables the detection of possible safety and security-rule violations. Achieving convergence and synchronization between the manufacturing physical space and cyberspace can radically reduce maintenance overheads, improve operations to alleviate product defects, respond to changes, and add value.

To enforce interoperability, a data wrangling method is introduced that is responsible for cleaning invalid, duplicate, or missing data and converting different

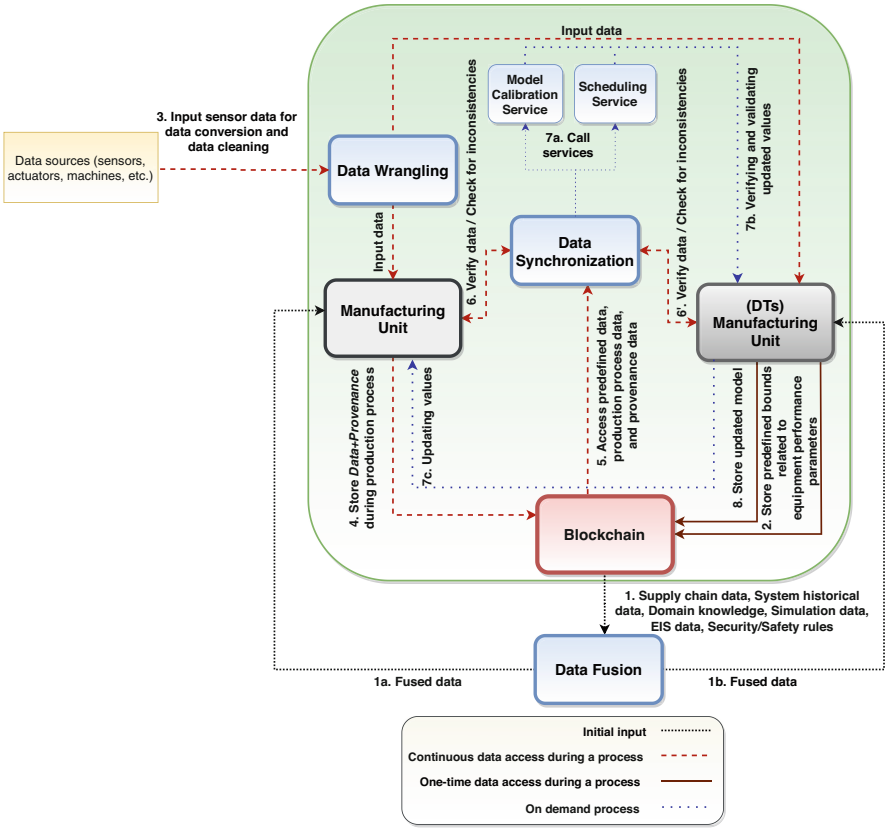


Fig. 5 During production process: Data flow through the main components (physical space (manufacturing unit), virtual space (digital twin), and blockchain) and methods (data wrangling, data synchronization, and data fusion) of the proposed framework (adapted from an image by Suhail et al. [13])

data formats into a unified format before entering the data to the twinned system (Step 3). As the production process initiates, the underlying process data along with provenance is recorded on the storage system (Step 4). Furthermore, a data synchronization method (Step 5) performs digital-physical mapping in order to check data inconsistencies or to detect anomalies. The data synchronization method carries out behavioral analytics and performs a continuous mapping between the predefined equipment performance parameters retrieved from the blockchain and real-time sensor data obtained from the manufacturing unit, to verify data consistency (Steps 6 and 6′).

The data synchronization method can access the updated process and provenance data directly from the ledger (other than the physical space) to eliminate the risk of untrustworthy data. In the case of data inconsistencies being reported by the data synchronization method, corresponding scheduling services (in the physical space)

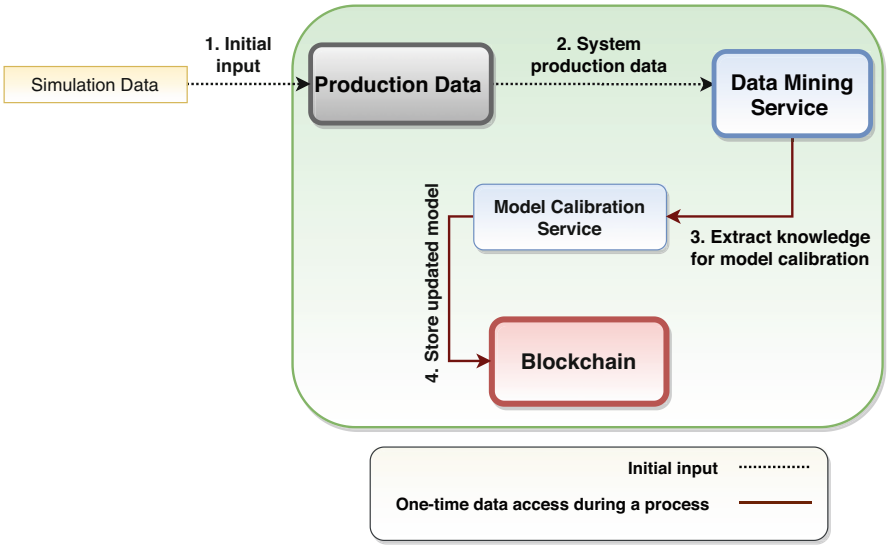


Fig. 6 After production process: data mining and model calibration

or model calibration services (in the virtual space) carry out the necessary measures (Step 7a) at the virtual system (Step 7b) to limit damage to the physical system and to detect system misconfigurations. Upon validation and verification at the virtual space, the changes are regulated at the physical system (Step 7c) afterward. The updated model is then stored in the blockchain (Step 8).

Finally, after the production process (Fig. 6), based on the simulation data (Step 1), the data mining service extracts new knowledge for model building and calibration (Steps 2 and 3). The updated models are stored in the blockchain to be used for the next production processes (Step 4).

In summary, the framework employs track and trace solutions through provenance data and enforces the use of a secure distributed blockchain ledger. Therefore, driven by data from reliable data sources and data storage, the framework makes use of DT models that mirror every facet of a product and thus help to achieve predictive maintenance.

4 Conclusion

In this chapter, we studied security and data trust challenges in smart manufacturing. Anomaly detection is one of the fundamental methods proposed in the literature to secure smart manufacturing. However, conventional anomaly detection methods are centralized and assume that log files are secure. To overcome these limitations, we introduced a blockchain-based solution where the log files are stored in the

blockchain. The method applies two layers of learning to detect anomalies, the site level and the organization level. This in turn keeps the delay in detecting attacks low.

Industrial processes rely on data for rational decision-making and precise execution. Consequently, extracting actionable insights from the collected data necessitates an infrastructure that can ensure the data's trustworthiness. To this end, we envisioned a blockchain-based framework to address the issues of data management and data security in manufacturing. From the trustworthy data recorded in the blockchain, product lifecycle events can be fed into data-driven systems such as DTs that can draw intelligent conclusions from data by identifying disruptions and recommending precautionary measures ahead of critical events.

This chapter studied the usage of blockchain for anomaly detection. As future work, sharing the information about the attack, referred to as threat intelligence, using blockchain can be considered. To share threat information, it is critical to protect the privacy of the companies as sharing the identity of the company will result in loss of reputation and revenue. The anonymity of the blockchain can reduce the privacy risk; however, how to trust an anonymized user is critical. Thus, the trade-off between privacy and trust in threat intelligence remains an open area for future research.

References

1. C. Feng, T. Li, D. Chana, Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks, in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (IEEE, New York, 2017), pp. 261–272
2. P. Fraga-Lamas, T.M. Fernández-Caramés, Ó. Blanco-Novoa, M.A. Vilar-Montesinos, A review on industrial augmented reality systems for the industry 4.0 shipyard. IEEE Access **6**, 13358–13375 (2018)
3. survey reveals digital twins are entering mainstream use (2019). https://www.gartner.com/en/newsroom/press-releases/2019-02-20-gartner-survey-reveals-digital-twins-are-entering-mai (Accessed on November 10, 2020)
4. E. Glaessgen, D. Stargel, The digital twin paradigm for future NASA and US air force vehicles, in *Proceedings of the 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA* (2012), p. 1818
5. M. Grieves, *Digital Twin: Manufacturing Excellence Through Virtual Factory Replication* (2014)
6. J. Inoue, Y. Yamagata, Y. Chen, C.M. Poskitt, J. Sun, Anomaly detection for a water treatment system using unsupervised machine learning, in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (IEEE, New York, 2017), pp. 1058–1065
7. Z. Jadidi, A. Dorri, R. Jurdak, C. Fidge, Securing manufacturing using blockchain. arXiv preprint arXiv:2010.07493 (2020)
8. G. Lee, M. Son, N. Choi, S. Hong, H. Kim, Blockchain based removable storage device log management system, in *2020 22nd International Conference on Advanced Communication Technology (ICACT)* (IEEE, New York, 2020), pp. 276–279

9. J. Li, W. Wu, D. Xue, P. Gao, Multi-source deep transfer neural network algorithm. Sensors **19**(18), 3992 (2019)
10. W. Pourmajidi, A. Miranskyy, Logchain: blockchain-assisted log storage, in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (IEEE, New York, 2018), pp. 978–982
11. S. Suhail, R. Hussain, M. Abdellatif, S.R. Pandey, A. Khan, C.S. Hong, Provenance-enabled packet path tracing in the RPL-based internet of things. Comput. Networks **173**, 107189 (2020)
12. S. Suhail, R. Hussain, A. Khan, C.S. Hong, Orchestrating product provenance story: When IOTA ecosystem meets the electronics supply chain space. Comput. Ind. **123**, 103334 (2020). ISSN 0166-3615. https://doi.org/10.1016/j.compind.2020.103334
13. S. Suhail, R. Hussain, R. Jurdak, C.S. Hong, Trustworthy digital twins in the industrial internet of things with blockchain. IEEE Internet Comput. **26**(3), 1–1 (2021). https://doi.org/10.1109/MIC.2021.3059320
14. Y. Taguchi, A. Kanai, S. Tanimo, A distributed log management method using a blockchain scheme, in *2020 IEEE International Conference on Consumer Electronics (ICCE)* (IEEE, New York, 2020), pp. 1–3

Blockchain for Data Sharing

Gowri Sankar Ramachandran and Bhaskar Krishnamachari

1 Introduction

"The world's most valuable resource is no longer oil, but data" is the title of the article published by Economist [1], which argues that the data is becoming an important and valuable commodity, just like oil. As the organizations and people around start to adopt digital technology, the emerging applications in the field of finance [2–4], supply chain [5, 6], Internet of Things [7–9], smart cities [10–12], healthcare [13–15], and agriculture [16–18] are already starting to become data-centric. These applications use data combined with artificial intelligence (AI) and machine learning (ML) technologies to analyze the data and make predictions. The growing advancements in AI and ML technologies and the organizations' eagerness to employ data-driven decision-making clearly show that the data will play a critical role in a wide range of applications.

In a data-driven application environment, the digital data is created, managed, and consumed by multiple organizations. In a supply chain application, information about a particular good is tracked throughout the supply chain process. Various warehouses, transportation agencies, and retailers may produce and consume data to reliably deliver the relevant stakeholders' products. Similarly, in the IoT application setting, the owner of a sensor device may share their data with the other application developers via a data marketplace [19–21]. Such examples show that in a multi-stakeholder application environment, the organizations and businesses that employ data-driven decision-making must coordinate and cooperate with other partners and stakeholders while fully understanding the trust, security, and privacy implications of the data.

G. S. Ramachandran (✉) · B. Krishnamachari
University of Southern California, Los Angeles, CA, USA
e-mail: gsramach@usc.edu; bkrishna@usc.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_20

The data produced by a given device or a user gets shared with other users, including the application developers, who then use the data to understand the trend, plan future actions, and estimate business risks. On the one hand, the data producer and the data consumer may not belong to the same organization, which forces the application developers and the data consumers to acquire data from other parties. On the other hand, the data producers may not fully utilize the data if it stays within their organization because sharing their data with other organizations may increase the utility while providing financial incentives. Naturally, large real-world applications involve multiple data producers and data consumers, highlighting the need for data sharing infrastructure.

The application developers and data consumers expect high-quality data from trustworthy sources to build reliable data-centric applications. Trust, security, and privacy are some of the critical concerns for the data consumers. The data sharing infrastructure must provide trust and quality guarantees to increase the data consumer's confidence. The emerging blockchain technology offers trust and transparency through a decentralized and distributed operational model, which is promising for data sharing applications.

In this chapter, we review the characteristics of digital data, data-driven applications, and specifications. Besides, we will also show how blockchain technology can be used to develop reliable, trusted, and secure data-driven applications involving multiple organizations.

2 Characteristics and Requirements of Data-Driven Applications

The digital data carry much useful information to fulfill the application requirements. This section reviews the characteristics of the digital data.

Static vs. Dynamic Data The data may change over time depending on the state of the application environment. When the data remain unchanged over time, then it is called *static data*. Examples of static data include historical data and other information that do not change throughout the application lifetime. On the other hand, the dynamic data exhibits fluctuations over time, which could be because of the application environment changes. A broad array of real-world applications fall under this category, including environmental monitoring and supply chain applications.

Copyable The digital data can be copied easily, which means many users and organizations could easily get a copy of the data. From the ownership perspective, it is difficult to claim ownership once the data from the initial owner.

Sensitive The digital data could contain sensitive details, including personally identifiable information, payment details, and location data, which could violate

the privacy of the users and organizations. Both the data producers and consumers may want to understand the privacy and legal restrictions attached to the data when developing data-driven applications.

Valuable The cost of producing, cleaning, and managing digital data could be high. Applications such as data marketplaces enable the data producers to sell their data in return for monetary benefit. It is, therefore, essential to estimate the value of the data.

Modifiable When the digital data is moving through networks and software systems owned and managed by multiple organizations, it becomes susceptible to corruption. Malicious entities could deliberately alter the data, making the data and the application unreliable.

Inconsistent The data may not follow a consistent pattern either due to the operational environment or other application-specific constraints. Without involving domain experts, it may not be easy to assess the digital data's quality and usefulness.

2.1 Requirements of Data Sharing Apps

In this section, we list down the needs and the expectations of the data consumers.

Trust Data consumers require trust guarantees from the data providers, which is critical for data-centric applications. As discussed in Sect. 2, the data can easily be copied or modified in transit, meaning any malicious entity can intercept the data at a point in the network and change it with malicious intent. Note that applications may use the data to make business decisions, including prediction. Incorrect or malicious data may lead to sub-optimal outcomes, which in the long run would discourage the organizations from building data-centric applications. It is, therefore, essential to provide trust guarantees to the application developers and their organizations.

Security Data security is another critical element of data-driven applications. Providing tight security to data through encryption schemes, cryptography primitives, and identity and access management solutions would increase the data consumers' confidence. Note that a secure data sharing platform may also gain the trust of the application developers.

Furthermore, the data may either be static or dynamic data depending on the application, as discussed in Sect. 2. In the case of static data, the security of the data store becomes an essential element. Attackers may corrupt the data store making the data unreliable. Therefore, the application developer may want to understand the security features of the data store and network security. Identity and access control solutions enable the data management platforms to secure access to the data store, which would be a good measure for the data owners to gain the data consumers' confidence.

Privacy The data consumers may want to understand whether the data includes any privacy-sensitive information, including personal details, location, and other metadata that may reveal delicate details about the person or an organization. With the emerging data protection regulations such as General Data Protection Regulation (GDPR) [22], and California Consumer Privacy Act [23, 24], the consumers want to ensure they are complying with the regulations to avoid legal issues. From the consumer's point of view, they want to understand who is providing the data, what information it carries, and the legal requirements of the source and the originating countries. Therefore, data providers and data sharing platforms must provide consumers with the necessary information to prevent privacy-related legal issues.

Price Fairness Peer-to-peer data selling protocols such as streaming data payment protocol (SDPP) also let the data owners sell their data to application developers [24] directly. Data sellers typically set the price for data products in such systems, wherein buyers may not want to pay a high price for data. A fair pricing model is therefore required to encourage consumers from buying data from marketplaces.

3 Terminologies and Data Sharing Modalities

This section will review the data sharing modalities and understand how it introduces trust, security, and privacy concerns for the relevant stakeholders. The term "stakeholder" refers to all the parties that have a stake in a data-centric application. A few key terminologies are discussed below:

- **Data Source:** The data originates from either a sensor, storage platform, or a processing device owned and managed by one or more organizations. The point at which the data is first created and entered into a digital world is referred to as *data source*. For any data, there can only be one data source. The right to share the data could belong to one or more organizations, depending on the application.
- **Data Sink:** Applications or users who typically receive or consume data referred to as *data sink*. There can be more than one data sink for given data since many consumers may use the same data for their applications.
- **Data Owner:** The data owner represents an organization or the user that has the right to share or sell the data. In some cases, data source and data owner may not be the same party.

In a data-centric application, the relationship between the data source and the data sink may be represented using the following patterns:

- **One-to-one:** In a one-to-one data sharing modality, the data produced by a source is shared with a single sink. Here, both the source and the sink may belong to the same organization. When both the source and the sink do not belong to the same organization, the data consumer and the data owner may want to negotiate and set up an agreement for data sharing to ensure trust, privacy, and security.

- **One-to-many:** Following this modality, the data from a single source may be consumed by multiple consumers belonging to various organizations.
- **Many-to-many:** In some applications, many data sources share their data with multiple consumers through a common data sharing middleware. Data marketplaces are an example of many-to-many data sharing modality.

Many real-world applications fall under one of these modalities. From the application developer's point of view, it is essential to understand these modalities. It provides insights into the data and the organizations and platforms involved in managing and sharing the data.

4 Data Sharing in Emerging Applications

This section reviews the emerging applications and their data sharing requirements.

Internet of Things The Internet of Things (IoT) technology provides remote monitoring and management capabilities to applications. IoT applications typically produce data by sensing their environment and report these data to a remote server for storage, analytics, and visualization purposes. In this setting, the industries that adopt the IoT technology use the data coming from remote devices to make business decisions by data analytics. Numerous studies illustrate the benefits of data and how it can be analyzed through machine learning and artificial intelligence algorithms [7–9].

Examples of data-driven IoT deployments include environmental monitoring and predictive maintenance. Air pollution affects numerous cities around the world. Air quality monitoring stations are deployed to understand air quality and help policymakers make decisions using evidence [25]. In the industrial setting, IoT devices are integrated into machines and production lines for studying their wear and tear, which allows the engineers to fix potential problems proactively [26].

Smart Cities Cities around the world are starting to adopt digital technologies to understand their citizen's needs better and make informed decisions about roads, water, electricity, and other utilities. Typically, a city consists of multiple departments for managing transportation, sanitation, law enforcement, etc. These departments collectively manage the resources and the needs of the city. Lately, these departments employ digital technologies, including cameras, environment monitoring stations, and mobile applications, to gather data from different neighborhoods in real time and then analyze those data to make informed decisions.

Data marketplaces are introduced to create large-scale data-driven applications using community members' data sources. Such solutions allow city administrators to buy data from various data providers. Such data marketplaces operate following "many-to-many" modality, which means multiple organizations and users consume multiple sellers' data sources. For example, the City of Los Angeles is building

a city-wide parking application using a data marketplace to connect parking data providers with application developers [27].

Healthcare In the healthcare domain, multiple hospitals deal with thousands of patients. While treating rare diseases, doctors may want to look at many patients' health records to better understand disease progression and potential medical interventions. Note that in handling pandemics such as COVID-19, hospitals are already starting to share data and knowledge to provide quality care to the patients. Such applications are showing the benefits of sharing medical records. However, healthcare data must comply with regulations such as HIPAA [28]. The organizations have to ensure that the data is not carrying any personally identifiable information (PII) to protect the patients' privacy.

Supply Chain Applications Consumer goods, including food supplies and electronic appliances, are transported to thousands of miles from the source location. The supply chain industry is responsible for delivering supplies to the desired location promptly. Lately, these applications are adopting digital technology to track the location of the goods in real time. Besides, there is a tremendous interest in gathering information about a product's journey, including who produced the item, when the item was first created, and what path did it take to reach a particular location.

The supply chain applications follow "many-to-many" data sharing modality because of multiple warehouses, retailers, and transportation agencies. In this setting, each stakeholder must share information about the state of a shipment in a timely fashion to other stakeholders to serve the end-consumers reliably and make logistical decisions. Therefore, supply chain applications demand a high level of trust from all the relevant parties.

Connected and Autonomous Vehicles The automotive industry is rapidly adopting connected and autonomous driving capabilities to vehicles. AI technology advancements drive such developments in combination with emerging low latency and high throughput wireless communication standards such as 5G. These technologies enable vehicles to gather and share enormous data about their operational environment in real time. For example, Cooperative and Adaptive Cruise Control are being developed to let vehicles collaboratively manage their speed through data-driven algorithms [29].

The future vehicles are also expected to autonomously buy and sell data and computation resources with other cars without human intervention [30]. For instance, the vehicles can share the charging availability at nearby electric-vehicle charging stations. Such applications are paving the way for the *data economy* involving the connected and autonomous vehicles.

Smart Grid Countries worldwide are beginning to employ data-driven approaches to efficiently distribute energy resources between neighborhoods, households, and grids based on the perceived and predicted demand. Understanding the energy demands of a household and predicting the amount of energy generated by a solar

panel in a house would help the grid operators plan their distribution strategies to ensure sustainability and resiliency.

In this domain, multiple governments and sustainability-friendly organizations offer incentives, including tax benefits and carbon credits, to motivate the community members and commercial organizations to adopt sustainable energy generation methods, including solar panels. Following such initiatives, multiple households and organizations can share their surplus energy with their neighbors and the grids creating an energy market. For such markets to operate efficiently, grid operators, distributors, producers, and consumers must share data about the consumption, production, and environmental parameters such as the amount of sunlight and wind speed.

Finance The growing adoption of digital technology in the financing sectors allows banks, stock exchanges, and other financial organizations to collect data about the market, consumer behavior, and the events that influence the market trends. Note that almost all the banks provide support for mobile applications to efficiently manage customers' funds while allowing them to pay for services without credit or debit cards. Such developments clearly show the importance of data for financial organizations and how it used to predict the trends, including customers' spending behavior and loan repayment capabilities.

5 Blockchain and Its Capabilities

The introduction of Bitcoin [31] paved the way for a novel and decentralized payment system with built-in mechanism for trust. Notably, the Bitcoin system removed the need for a centralized entity to manage and verify the payment transactions. Ever since the arrival of Bitcoin, a number of platforms have been developed to provide support for peer-to-peer transactions in a decentralized setup. Such an architectural model removes centralized intermediary, thereby preventing central points of failure. The Blockchain technology provides a combination of cryptography, consensus algorithm, distributed ledger, and, optionally, decentralized computation capability, to create a decentralized and trustworthy platform.

5.1 Cryptographic Digital Signature

The public-key cryptography is used in blockchain to generate a signature for Blockchain transactions. Users carry out transactions by creating a digital signature using their private keys. Recipients in the blockchain network verify the transaction using the public key of the sender to ensure that the transaction is indeed signed by the sender. Source or end-devices sign the transactions when they create a transaction.

5.2 Distributed Ledger

Blockchain use a distributed storage to record the transactions. In essence, all the platforms in the network store either all the transactions or a subset of transactions. All the nodes in the network come to a consensus (using a consensus algorithm) before entering the transactions into the ledger. This feature makes blockchain effectively immutable. While initially blockchain protocols were designed for cryptocurrency transactions, in general, any data could be included in the ledger.

5.3 Consensus Algorithm

Blockchain does not rely on a centralized server for verification and validation of transactions. Instead, Blockchain uses a peer-to-peer model, and all the decisions within the network are made by the participating members through a consensus protocol. There are many different consensus approaches ranging from permissionless proof of work with longest-chain adoption in Bitcoin to the permissioned Byzantine fault-tolerant protocol used in Tendermint [32, 33].

5.4 Decentralized Computation

Applications built on top of a blockchain or distributed ledger technology are called "decentralized applications" or Dapps. Such application logic may be executed by the full nodes in the blockchain network by executing smart contracts (e.g., in Ethereum) or through off-chain interactions with computational servers (e.g., in Hyperledger Fabric) following a protocol defined by the underlying blockchain platform.

Figure 1 shows how peer-to-peer transaction is handled through the blockchain technology and the role of cryptography, consensus protocol, and distributed ledger. All blockchain and distributed ledger technology platform follow the interaction pattern described in Fig. 1. A peer in a blockchain architecture refers to a device that can initiate a transaction. In contemporary blockchain systems, mobile phones, desktop computers, and servers can act as a peer since the devices require computation support for consensus process. Resource-constrained devices are not capable of initiating transactions due to their lack of hardware and processing support for encryption and decryption protocols used in blockchain platforms, unless a special hardware is integrated to these devices to run encryption protocols.

Blockchain or DLT platform typically consists of a collection of full nodes, which run the consensus algorithm and maintain the distributed ledger for storing the transactions. Due to the resource requirements and the networking demands of the full nodes, a server-grade computer is typically required for consensus process,

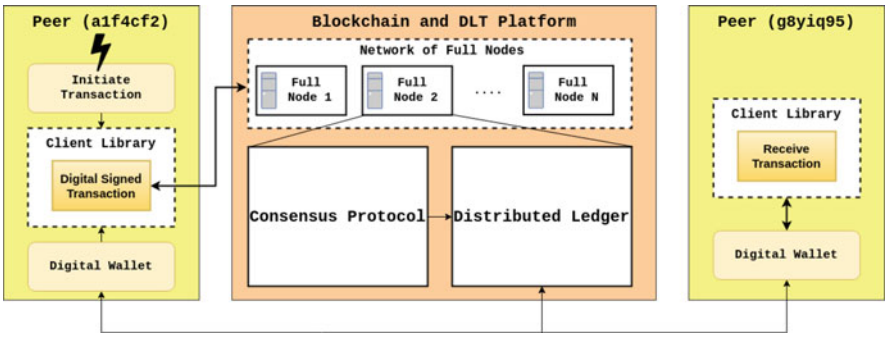


Fig. 1 Executing peer-to-peer transaction using blockchain

with the exception of Tendermint, which can run even on Raspberry Pi [32, 33]. All the peer-to-peer interactions happen via the blockchain using the client library provided by the underlying platform. Note that the client libraries can run on mobile phones and lightweight devices including IoT gateways.

6 Trusted Data Sharing Using Blockchain

The development of a data-driven application in a multi-stakeholder setting requires a significant amount of trust in the data and as well as the infrastructure that handles the data. Here, the movement of data, coupled with the participation of multiple stakeholders, forces the device owners and the application developers to seriously introspect the data flow and the role of each stakeholder in the end-to-end application process. When a single stakeholder runs the data sharing infrastructure, it provides an opportunity for the infrastructure owner to tamper with the data when it flows through his or her infrastructure.

Data sharing following the renowned publish-subscribe communication model have been widely used for exchanging data between various parties. This communication model is prominent in many real-world applications because of its resource efficiency and ease of use. However, the contemporary publish-subscribe frameworks, including MQTT [34], Apache Kafka [35], and RabbitMQ [36], follow a centralized architecture, wherein a single stakeholder runs a centralized data broker and manages the data communication between various parties. Trinity [37] presents a byzantine fault-tolerant publish-subscribe broker to enable multiple stakeholders to share data reliably, but it lacks support for both micropayments and data processing. And, it does not present any Quality-of-Service levels to handle different application requirements.

The publish-subscribe model is also employed in data marketplaces because of its versatility and scalability. For instance, the I^3 data marketplace [38] uses MQTT in combination with a custom payment software to enable the data or device owners to monetize their data using a publish-subscribe data sharing framework.

Table 1 Comparison of Trinity with other publish-subscribe frameworks

Brokers	Distributed	Crash tolerance	Byzantine fault tolerance	Payment support
Apache Kafka	Yes	Yes	No	No
RabbitMQ	Yes	Yes	No	No
MQTT	Yes	Yes	No	No
I3 data marketplace	No	No	No	Yes
Trinity	Yes	Yes	Yes	No
Pub-Pay-Sub	No	No	No	Yes

Similarly, the pub-pay-sub broker [39] supports micropayments on top of a publish-subscribe architecture, but it follows a centralized architecture assuming a single stakeholder. Therefore, it is susceptible to central points of failure including byzantine faults, and it did not present any mechanisms to guarantee trust.

Well-known brokers, including MQTT [34], Apache Kafka [35], and RabbitMQ [36], can tolerate crash failures, but they neither handle byzantine faults nor provide support for micropayments. To summarize, the existing publish-subscribe data sharing frameworks do not provide mechanisms to guarantee trust while providing support for micropayments and data processing when operating in a multi-stakeholder setting. Table 1 compares the contemporary publish-subscribe brokers and highlights the need for byzantine fault-tolerant and trusted publish-subscribe broker for data processing.

Trinity is a distributed Byzantine fault-tolerant pub-sub broker [37]. It is created for multi-stakeholder applications, in which each stakeholder is assumed to be operating a broker and a consensus node within their domain to serve its local publishers and subscribers, as shown in Fig. 2. All the domains are connected through a consensus layer, which is responsible for validating published messages. Each message must be approved by more than one-third of the consensus nodes, before it is sent to the subscribers via a broker.

The safety and liveness properties of Trinity are presented in [37]. Trinity's support for publish-subscribe communication model and its ability to operate with any blockchain platforms clearly show how a well-known and widely used data sharing protocol such as pub-sub can be interfaced with blockchain platform to enhance trust in a multi-stakeholder data sharing application. Similar architecture can be followed to interface other traditional and centralized data sharing protocols such as web services.

7 Peer-to-Peer Data Sharing Using Blockchain

The data adds value to a number of applications. In the context of IoT, machine learning and AI algorithms can use the environmental sensor data to better understand the climate change or correlate the environmental data with the traffic sensor

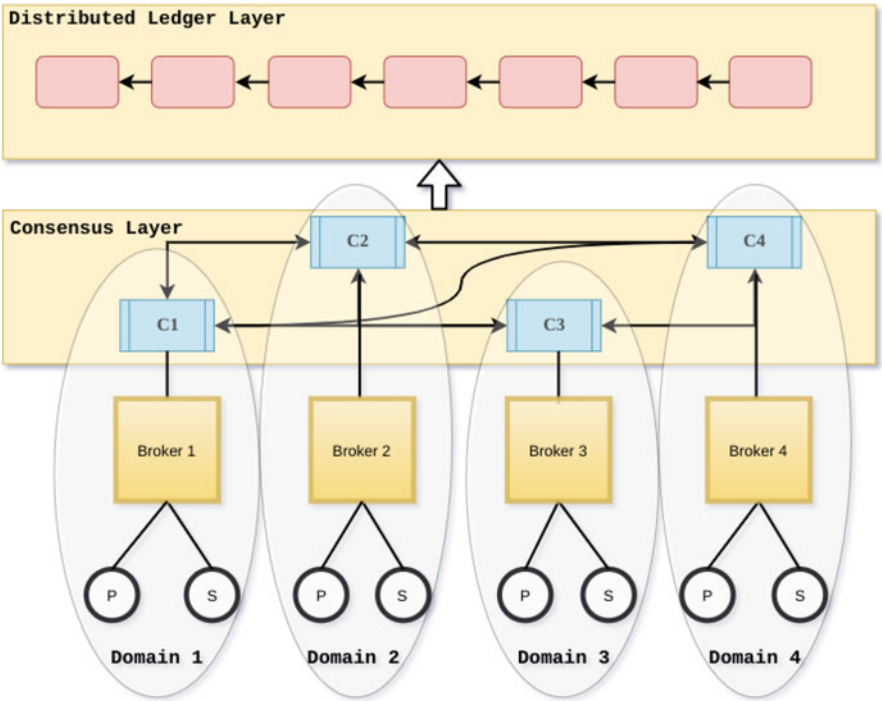


Fig. 2 Architecture of Trinity [37]

data to study the patterns and recommend routes that have less pollution. Such applications require data from a large collection of sensors from wide geographical neighborhood. But the deployment of sensors and the necessary communication infrastructure at city scale require significant financial resource while introducing high management and maintenance complexity. "Data economy and monetization" revolve around the concept of incentivizing the data owners and mobile phone users to share their data with application developers. This new type of "data-for-value" applications is starting to emerge at the IoT and smart cities' landscape through data marketplaces, which allow the device owners to stream their data in real time to the applications managed by the buyers. The involvement of multiple stakeholders and the emerging data economy allow applications to scale, but they require solutions for trust, payment, and identity management.

Support for micropayments is desired as the data owners start to monetize their data. Each owner could maintain a digital wallet to buy data or other services from other devices and service providers. On the other side, devices providing services receive money from other digital wallets. In this type of applications, devices initiate a transaction by digitally signing the transaction as shown in Fig. 3. The blockchain platform handles the transaction and updates the digital wallet on both the selling and buying peers (see Fig. 3). Notably, the transactions are submitted to

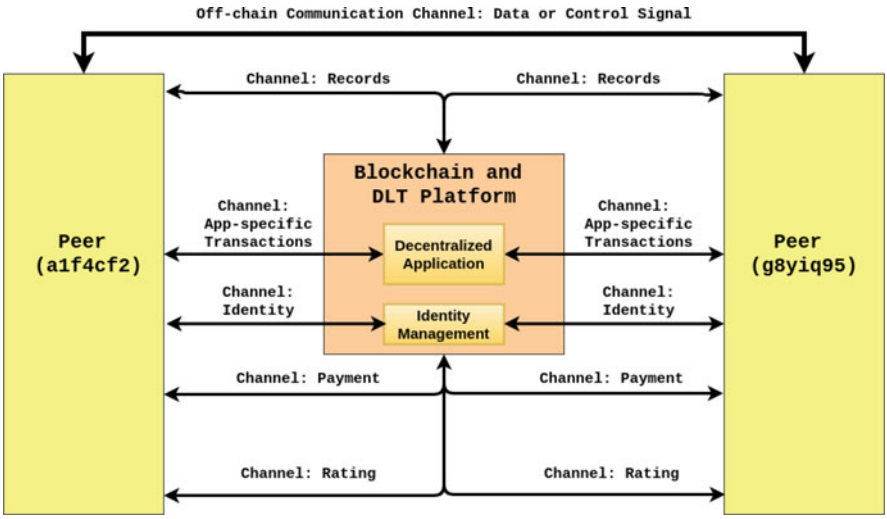


Fig. 3 Channels in blockchain-based peer-to-peer marketplaces

the blockchain platform using a write operation, while the digital wallet updates and reads its current account balance through a read operation.

Almost all the blockchain-based data sharing applications would employ communication channels presented in Fig. 3, which are discussed below:

App-Specific Transaction Channel IoT applications employing blockchain for micropayments and other types of applications require a channel to either write to or read from the blockchain platform. This channel is typically used to interact with a smart contract or other decentralized application hosted on a blockchain platform.

Payment Channel The payment channel is responsible for handling micropayments between peers. The peer that initiates the transaction requires hardware support to prepare and submit the transaction to the blockchain platform. Both the sender and the receiver interact with the blockchain platform through the payment channel to send and receive money. Peers are expected to maintain a digital wallet to store and manage currencies.

Identity Channel The large-scale IoT deployments are expected to involve multiple stakeholders to perform sensing, processing, and communication functionalities. When IoT devices buy or consume services from other stakeholders, it is important to verify the identity of the service providers and sellers. This channel is used to register, manager, and query identities from the underlying blockchain-based identity management platform.

Rating Channel Blockchain and DLT technologies allow devices and service providers to provide and consume data and other services from various stakeholders. To make sure that the stakeholders are honest and trustworthy, it is important to

maintain the trust and reputation rating of the peers. Misbehaving peers are removed from the platform when their rating is not satisfactory. The Rating Channel assumes the availability of identities.

Records Channel This channel records all the transactions in the ledger for future verification. Since the peers exchange services with each other through a blockchain platform, each and every transaction has to be recorded in the ledger to go back and verify the activities in case of disputes. Note the seller of a data or service may be cheated when the buyer leaves from the application network without paying for the service after consuming a service or data from the buyer. At the other side, the buyer can pay for the service or data prior to consumption and the seller may leave the application network without providing the service or data to the buyer. This problem is referred to as "seller's and buyer's dilemma [40]." To prevent disputes in such scenarios, it is important to record all activities such as ordering, invoices, payment receipts, etc. on the blockchain/DLT platform.

Off-Chain Communication Channel Devices (or peers) exchanging data or compute services or sending control signals for actuation are not recommended to use the blockchain platform due to the large amount of data. Blockchain platforms are both expensive and less responsive when exchanging large quantity of data. We recommend the application developers to use the blockchain platform only to store the record of data transactions and use a dedicated data channel off-chain through UDP, TCP, or other types of communication frameworks using this communication channel.

A Note on Smart Contracts Some blockchain platforms come with support for smart contracts. Applications employing smart contract would still use a client library to digitally sign the transaction before submitting it to a smart contract, and hence the interaction of the client with the blockchain would remain the same whether or not a smart contract is used. Functionally, the smart contract provides a way to automate decentralized application functionalities through transaction events, which means each transaction event triggers one or more functions on the smart contract.

Figure 4 shows how blockchain-based IoT applications can sell and buy data through a peer-to-peer data sharing mechanism. IT shows how devices at IoT layer can interact with the blockchain through a client library using the different communication channels presented in Fig. 3.

Among all the layers presented in the IoT layer, "connected things" devices are less suitable for running blockchain-based applications due to their resource constraints. Notable exceptions are smartphones and Raspberry Pi class devices as they possess sufficient resources for digitally signing blockchain transactions using public-key cryptography. Internet connectivity is critical for applications that use blockchain platforms.

Streaming Data Payment Protocol Streaming data payment protocol (SDPP) enables a buyer and seller to easily connect and transact with each other using

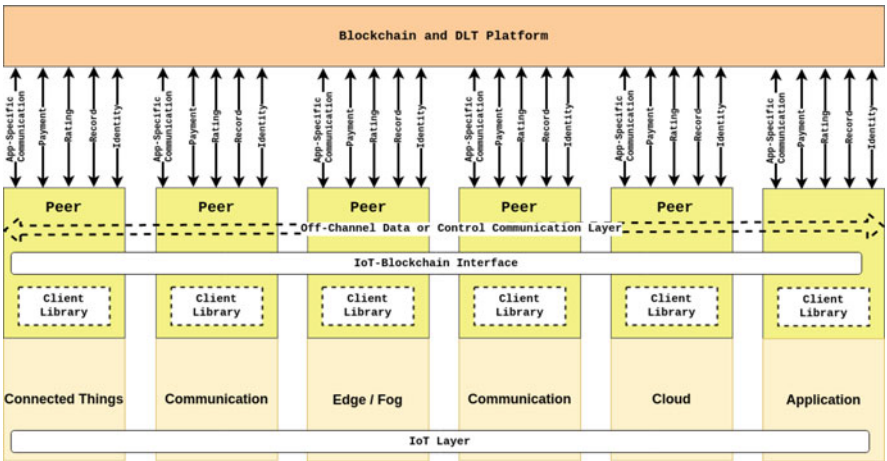


Fig. 4 Architecture showing how IoT applications can share IoT data using a peer-to-peer marketplace

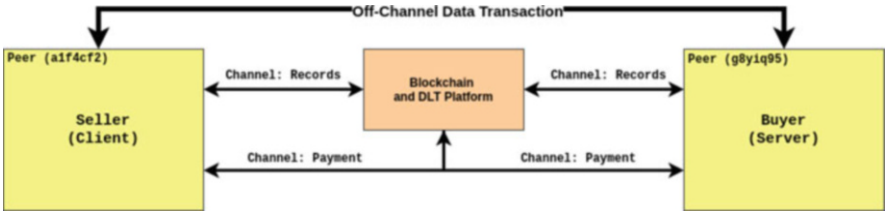


Fig. 5 Communication channels in streaming data payment protocol [42]

micropayments for streaming IoT data [41, 42]. SDPP is a peer-to-peer data sharing protocol. Its design carefully separates out three key components: the off-chain data communication channel (which is operated as a traditional Internet client-server application-layer protocol, atop TCP), a payment channel (implemented using a cryptocurrency protocol), and a records medium (implemented using a distributed ledger technology), as shown in Fig. 5.

Many blockchain-based applications use one or more channels shown in Fig. 3. MOTIVE is a novel vehicle-to-infrastructure and vehicle-to-vehicle (V2x) platform with support for trust management, micropayments, and mechanisms to provide and consume data and compute services with other vehicles and road-side units following a decentralized architecture. It uses a blockchain and protocol agnostic framework for V2X involving blockchain and distributed ledger technologies, which uses rating, payment, identity management, and an off-chain communication channel [30] to reliably exchange data and payment services using a peer-to-peer application model.

Similarly, FairAccess [43] uses the blockchain technology to create an access control framework for the IoT applications. The implementation of FairAccess consists of a channel for managing and requesting access to remote devices and services through an application-specific communication channel. Lin et al. [44] present a framework to store the data from LoRaWAN devices to an immutable storage provided by the blockchain platform by connecting the LoRaWAN gateway to a blockchain platform. This framework uses the records channel to store the information on the blockchain. Dorri et al. [45] apply the blockchain technology to smart home applications for handling the security and privacy concerns. All these applications connect the IoT applications to the blockchain platform using communication channels that are a subset of those presented in Fig. 3. Based on the several examples above, it is clear that blockchain-based peer-to-peer data sharing applications can be architected using the generic communication channels presented in Fig. 3.

8 Blockchain-Based Data Marketplaces for Data Economy

The marketplace approach aims to enable a world where application developers can gain access to the myriad of sensors, actuators, and other networked devices that others have deployed and connected to the network. The device owners can take the initiative and deploy intelligent sensors in anticipation of an emerging and independent application market that will utilize their data for the benefit of its users. This is especially useful in smart cities that may be eager to utilize existing infrastructure and deployments in unique ways but are unable to deploy the IoT devices themselves.

Given the economic value associated with data, it becomes essential for parties involved to sell, find, and buy the data as easily as possible. Such an issue gives rise to the need for data marketplace platforms. Like marketplaces for physical goods such as books and electronic items, the data marketplaces are being created to help the device owners and data providers to sell their data with application developers. In recent years, such an idea has been independently proposed and started to be explored by several researchers and practitioners [20, 21, 46–51].

I3 [20] is an open-source IoT data marketplace [19] for smart communities. A city-wide parking application is currently being developed using the I3 data marketplace in Los Angeles, USA [27], to connect multiple parking data providers with the application developers. Another active data marketplace is Terbine [49], which sells traffic data and environmental data such as temperature, humidity, and air quality for multiple countries. Data from various sources, including financial and governmental organizations, are available on the Snowflake data marketplace [50]. QueXopa [51] is an emerging data marketplace in Latin America, providing data about the transportation industry and environment for application developers. These efforts show that the data marketplaces are starting to gain traction, and

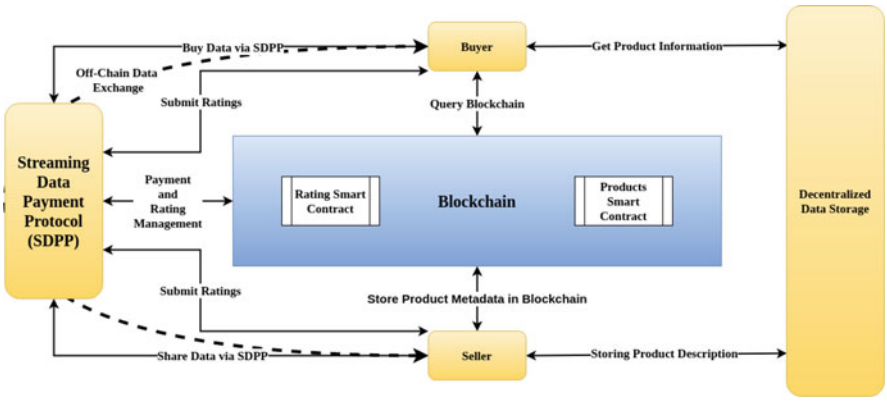


Fig. 6 Architecture of decentralized data marketplace (DDM) [52]

many application developers are leveraging such platforms to build data-driven applications.

While centralized marketplaces allow the sellers to connect with the buyers, it relies on a centralized server owned and managed by a single organization. Centralized infrastructures involving single organization are susceptible to central point of failure. In other words, the organization that manages the infrastructure may act with a malicious intent by removing data products, hiding products from selected sellers, or selling a product without the seller's knowledge. Such behaviors could discourage the buyers to utilize data marketplaces for their application model, since it does not provide any trust guarantees. Blockchain-based data marketplaces have been proposed to enable decentralization and transparency [52–56]. DDM [53], Ocean [55], and Streamr [56] are examples of decentralized data marketplaces, which leverage blockchain technology and smart contracts to involve multiple organizations in the administration process, enhancing trust and transparency.

DDM [52] is a decentralized data marketplace which combines the blockchain technology, smart contract, decentralized data storage, and a payment protocol to create a trusted and decentralized data marketplace. The core marketplace components either executed inside the blockchain platform in the form of a smart contract or they coordinate with the blockchain platform to store the transaction details.

Figure 6 shows the architecture of DDM. It uses a smart contract to store product metadata, including the name of the product and a pointer to its detailed product information. Note that the blockchain platforms are not capable of storing thousands of kilobytes of data. DDM uses an off-chain decentralized data storage platform to store product information while using the blockchain smart contract to advertise the product to potential buyers. A buyer is required to interact with the DDM smart contract to learn about the data products. When a buyer decides to buy the product, he/she is required to use the streaming data payment protocol (SDPP) to directly get the data from the seller in return for the payment. Since the buyers and

sellers are interacting without a centralized intermediary, there may be disputes. To prevent cheating and handle disputes, DDM records all the transaction details in the blockchain's immutable storage. If there is a dispute, blockchain can be used to verify the history of transactions. Besides, a rating smart contract is used to keep track of the users and their reputation.

9 Protecting Privacy Using Blockchain

The data may carry sensitive information that the users may not be willing to share with other organizations and application developers. Transactions in public blockchain platforms such as Bitcoin [31] and Ethereum use the user's public key, which doesn't reveal the identity of the user. Therefore, the user's identity is not revealed when handling payment or other smart contract transactions in a public blockchain platform. In the case of permissioned or private blockchain platforms, the organizations sign agreements and join the blockchain network. Under this operational model, users cannot entirely hide their identity, which means privacy could be an issue. Note that the nodes that run the consensus protocol may have to reveal their identities in permissioned blockchain nodes. However, the clients' privacy (which submit transactions) depends on the application's identity and access control protocol.

When using a blockchain platform as an immutable data store, the application developers must ensure that the data do not contain sensitive information. Remember, the blockchain ledger is visible to everyone, especially in public blockchain platforms. Encryption schemes are typically used to hide sensitive information. Even systems that use encryption schemes have to decrypt the data before performing any encrypted data operations. However, homomorphic encryption schemes preserve privacy by performing computation on encrypted data [57]. In other words, the encrypted result of the computations on encrypted data will match the result of the computations performed on plain text. Homomorphic encryption schemes have been developed for several applications in the last two decades, but the practical use of such systems is still under scrutiny due to their resource demands [58].

The secure multi-party computation was introduced in 1982 by Andrew Yao [59] to allow two parties to perform computation without relying on third parties while keeping the local data or input secret [60]. Approaches such as zero-knowledge proof allow the worker and validator to verify the result without exchanging any information except that the verifier knows the correct result [61]. ZKSnark [62] is one of the practical implementations of Zero-knowledge proof, and it uses the libsnark library. Such schemes are capable of protecting the user's privacy. The growing interest and the potential in blockchain technology encourage cryptographic enthusiasts to implement privacy-preserving protocols and release them as open-source software to increase adoption. The future data sharing applications can leverage schemes such as Zero-Knowledge Proof or Secure Multi-party Computation to process data without violating user's privacy.

Ekiden [63] is a privacy-preserving computation platform for the execution of smart contracts. Intel SGX is used for executing the computation in a cryptographically secure execution environment. Similarly, Ohrimenko et al. [64] contribute a framework for running machine learning applications in a trusted hardware environment. Such hardware-based approaches rely on the underlying hardware's security mechanisms and perform all the computation inside the secure environment to preserve privacy. Only the hardware with the support for SGX can participate in the network, which is a significant limitation of hardware-based approaches.

10 A Note on Tamperproof and Secure Storage of Data Using Blockchain

The blockchain technology also provides tamperproof storage using a Merkle Tree data structure. Data-driven applications can leverage the blockchain's storage mechanism to ensure that the data has not been modified during the transit from the data producer's infrastructure. Note that the blockchain technology's tamperproof storage mechanism prevents the data from being manipulated or altered after it is stored in the ledger. When a malicious actor tries to modify the data, he/she can only change the ledger on his/her local node. It is impossible to tamper with the transactions and data stored on hundreds to thousands of nodes that are part of the blockchain network. Therefore, the blockchain ledger is a reliable data storage platform for data-driven applications. While this is good for applications, it isn't easy to delete data when the user wants his/her data to be removed from the blockchain, which is one of the open challenges.

11 Opportunities and Design Challenges

In this section, we will review the opportunities and open challenges. Addressing these existing challenges would benefit data-centric applications.

11.1 Big Data and Blockchain

Machine learning and AI algorithms tend to perform better with the large data set. The application developers, therefore, prefer big data to build accurate and high-quality models. However, the blockchain-based data sharing platforms cannot store big data in the blockchain ledger due to their storage limitations. Emerging blockchain-based data sharing and management solutions store the big data outside the blockchain in either a decentralized data storage, database, or other servers and

register only the hash and the pointer to the data in the blockchain. Such solutions still guarantee trust, but it introduces additional overhead for the developers. Future data-driven applications would benefit from large-scale storage.

11.2 On-Chain Computation Using Data

The application developers use data in their applications to analyze the trends and predict events. Such analysis requires computation support. Blockchain platforms such as Ethereum [65] provide support for smart contracts. Such platforms do not support complex computations within blockchain due to the processing and protocol limitations. If blockchain platforms provide high computation capabilities, the application developers can run their analysis within the blockchain platform using multiple providers' data.

TrueBit [66] presents trustless smart contracts to enable secure computation on the blockchain. In PoW-based blockchain systems, each node in the network executes the smart contract to verify the result, and the nodes are not rewarded for their verification. The node that solves the cryptographic puzzle gets an incentive since it is authorized to create the next block. In contrast, all the other nodes verify the block's integrity to make sure they are not attaching themselves to an invalid or malicious chain. TrueBit proposes a novel approach, by which a set of special nodes are elected to perform the verification instead of asking all the validator nodes to expend resources for the computation [66]. TrueBit injects faults to the computation to check the behavior of the validating nodes. Nodes that fail to report the "forced errors" are penalized for their dishonest behavior. This feature enables TrueBit to remove dishonest nodes from the network. Alternatively, platforms such as Hyperledger Fabric [67] and Tendermint [32, 33] separate the smart contract execution from the blockchain's consensus and ledger, thereby providing support for complex computations. From the application developer's point of view, it is essential to understand the security and the reliability of such solutions.

11.3 Garbage-In-Garbage-Out

As discussed in Sects. 6 and 10, the blockchain technology allows the application developers to trust the data that is managed by and stored in the blockchain. However, the data providers may share malicious or incorrect data, leading to the "Garbage-In-Garbage-Out" problem. While blockchain technology can store the data securely in the immutable ledger, it cannot guarantee that the owner's data is genuinely representing the ground truth. Therefore, application developers may have to assess the credibility of the data.

Existing solutions include rating and reputation management, allowing the data consumers to rate the data based on their experience. DDM [52], for example, uses

a rating smart contract to rate the data and its seller. The rating would indicate the quality of the data, but more consumers must buy and use the data to gain a good rating for a given product. Besides, in a competitive data market, sellers may buy and leave bad ratings for their competitors. Therefore, the data quality assessment is still an open problem, which must be solved to increase trust in an open data marketplace.

11.4 Bootstrapping Multi-Stakeholder Data-Centric Applications

The data-centric applications typically involve multiple stakeholders, as discussed in Sect. 4. The blockchain-based data sharing solutions require all the relevant stakeholders to go through the business process and translate it into blockchain-based software, including smart contracts. Each organization that holds a stake in the application must set up an infrastructure, share their necessary keys, sign up for an authentication and access control model, install new applications, and train their employees. From the management point of view, these are time-consuming processes. When companies have conducted business in the past, it may be easier to sign agreements due to prior knowledge. It may be challenging to bootstrap a blockchain-based data sharing infrastructure if the organizations and the stakeholders do not know each other.

From the blockchain perspective, a consortia model is widely employed to educate the companies about the blockchain and its potential and simplify the bootstrapping processes. Note that organizations with mutual interests join these consortia to explore emerging technologies such as blockchain and leverage these technologies in their business. Such efforts promise to ease the bootstrapping challenge. Still, it may be essential to develop simple yet secure and trustworthy solutions to bring up a blockchain-based data sharing infrastructure involving multiple stakeholders quickly.

11.5 Heterogeneity of Legal Standards

Emerging regulations such as General Data Protection Regulation (GDPR) [22] and California Consumer Privacy Act [23, 24] force the data providers and consumers to carefully collect, store, and share the data. Organizations are required to comply with these regulations to protect the privacy of the end-users. Although such regulations are meant to protect the user's sensitive data, subtle policy variations may depend on the user's geographic location or country of origin. For example, when an organization based in the European Union (EU) wants to set up a data sharing platform with an organization in California, they will have to carefully review

what data can leave the EU zone to California, and vice versa. As mentioned in Sect. 11.4, the complexity of the bootstrapping process may discourage companies from adopting blockchain-based data sharing solutions; on top of it, the legal challenges may further jeopardize the adoption. It may be important to simplify these regulations through a global standard, which may happen in the near future.

11.6 Real-Time Data Sharing Limitations

Blockchain platforms run on a distributed network of compute nodes. Each transaction, including data, must undergo the consensus process before it can be stored on the ledger. The consensus process requires multiple compute nodes to participate in the transaction verification and approval process collectively. In the Proof-of-Work (PoW) consensus model, the compute nodes must solve a puzzle, and the winner is allowed to verify the transactions and propose the next block. Such a model offers security and enables decentralization, but it comes at the cost of low throughput, which leads to high transaction delays. Other blockchain platforms such as Tendermint and Hyperledger Fabric could provide better performance than PoW. Still, it is essential to assess different consensus models' pros and cons when selecting a blockchain platform for data sharing. The application developer may have to understand the lack of real-time transaction support when considering a blockchain-based solution. But a real-time and trusted data exchange involving blockchain remains an open challenge.

11.7 Rights Management: Who Is the Owner?

As mentioned in Sect. 2, the data can easily be copied once it leaves the owner's infrastructure. Blockchain technology provides a secure and tamperproof storage of the data. Once the data is shared with a buyer or consumer, it can be easily copied and resold via various channels outside the marketplace. Digital Rights Management (DRM) tools enable digital media owners to restrict unauthorized distribution or copying of content. Similar tools and technologies are needed to retain ownership rights in a data marketplace.

11.8 Data Deletion Challenges

Blockchain technology uses immutable storage to retain the history of transactions in the form of blocks. Each node in the distributed blockchain network keeps a copy of the ledger. When a node tampers with data in its local ledger, it no longer matches the copy retained in the network, making blockchain tamperproof. However, the

data-driven applications may want to delete the data from a ledger due to privacy concerns. In such cases, it is impossible to remove all the transactions associated with a particular user from all the nodes in the blockchain network.

When the data producers expect that the data may need to be deleted, they may have to select the technologies carefully. In this case, the blockchain ledger itself may not be the right solution. Alternatively, the data could be stored on a database, while the hash of the data lives on the blockchain. Such solutions help the user keep the data outside the blockchain. It is up to the application developer and the data owner to carefully consider the data deletion requirements at design time to avoid issues in the future.

12 Summary

The interest in data and data-driven decision-making is on the rise due to advancements in machine learning and artificial intelligence. A wide array of applications in domains such as the Internet of Things, Supply Chain, Smart Cities, Connected and Autonomous Vehicles, and Smart Grids are increasingly relying on data while operating in multi-stakeholder environments. In these application environments, the data is produced and shared among multiple organizations. Application developers and data consumers expect quality data to make reliable business decisions. Trust, security, and privacy concerns induce skepticism for the application developers and data consumers. Blockchain-based data sharing solutions enable the data producers and data consumers to reliably exchange data while storing the data in an immutable ledger for long time usage. The decentralized data marketplaces create a transparent and reliable medium for data sellers and data buyers to exchange, sell and buy data.

While the blockchain offers decentralized, transparent, and trusted data sharing platforms, there are still open challenges around regulations, scalability, data quality, and storage. The emerging solutions are starting to address these issues; therefore, the future looks promising for data-driven applications.

References

1. The Economist, The world's most valuable resource is no longer oil, but data, in *The Economist: New York, NY, USA* (2017)
2. E. Brynjolfsson, K. McElheran, The rapid adoption of data-driven decision-making. Amer. Econ. Rev. **106**(5), 133–39 (2016). https://doi.org/10.1257/aer.p20161016. https://www.aeaweb.org/articles?id=10.1257/aer.p20161016
3. R. Culkin, S.R. Das, Machine learning in finance: the case of deep learning for option pricing. J. Invest. Manag. **15**(4), 92–100 (2017)
4. H. Arslanian, F. Fischer, *The Future of Finance: The Impact of FinTech, AI, and Crypto on Financial Services* (Springer, Berlin, 2019)
5. S.A. Gawankar, A. Gunasekaran, S. Kamble, A study on investments in the big data-driven supply chain, performance measures and organisational performance in Indian retail 4.0 context. Int. J. Prod. Res. **58**(5), 1574–1593 (2020)

6. G. Baryannis, S. Dani, G. Antoniou, Predicting supply chain risks using machine learning: the trade-off between performance and interpretability. Future Gener. Comput. Syst. **101**, 993–1004 (2019)
7. A. Ghosh, D. Chakraborty, A. Law, Artificial intelligence in internet of things. CAAI Trans. Intell. Technol. **3**(4), 208–218 (2018)
8. G. Dartmann, H. Song, A. Schmeink, *Big Data Analytics for Cyber-Physical Systems: Machine Learning for the Internet of Things* (Elsevier, Amsterdam, 2019)
9. F. Piccialli, S. Cuomo, N. Bessis, Y. Yoshimura, Data science for the internet of things. IEEE Int. Things J. **7**(5), 4342–4346 (2020)
10. Z. Allam, Z.A. Dhunny, On big data, artificial intelligence and smart cities. Cities **89**, 80–91 (2019)
11. S.M. Wu, T.-C. Chen, Y.J. Wu, M. Lytras, Smart cities in taiwan: a perspective on big data applications. Sustainability **10**(1), 106 (2018)
12. S. Suma, R. Mehmood, A. Albeshri, Automatic event detection in smart cities using big data analytics, in *International Conference on Smart Cities, Infrastructure, Technologies and Applications* (Springer, Berlin, 2017), pp. 111–122
13. F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, Y. Wang, Q. Dong, H. Shen, Y. Wang, Artificial intelligence in healthcare: past, present and future. Stroke Vascular Neurol. **2**(4), 230–243 (2017)
14. B. Kalis, M. Collier, R. Fu, 10 promising ai applications in health care. Harvard Business Rev. (2018). https://hbr.org/2018/05/10-promising-ai-applications-in-health-care
15. P. Groves, B. Kayyali, D. Knott, S. Van Kuiken, The'big data'revolution in healthcare: Accelerating value and innovation (2016)
16. A. Kamilaris, A. Kartakoullis, F.X. Prenafeta-Boldú, A review on the practice of big data analysis in agriculture. Comput. Electron. Agricult. **143**, 23–37 (2017)
17. M.R. Bendre, R.C. Thool, V.R. Thool, Big data in precision agriculture: Weather forecasting for future farming in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)* (IEEE, Piscataway, 2015), pp. 744–750
18. N. Shakoor, D. Northrup, S. Murray, T.C. Mockler, Big data driven agriculture: Big data analytics in plant breeding, genomics, and the use of remote sensing technologies to advance crop productivity. Plant Phenome J. **2**(1), 1–8 (2019)
19. X. Zhao, K.K. Sajan, G.S. Ramachandran, B. Krishnamachari, Demo abstract: The intelligent iot integrator data marketplace-version 1, in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)* (IEEE, Piscataway, 2020), pp. 270–271
20. B. Krishnamachari, J. Power, S.H. Kim, C. Shahabi, I3: An iot marketplace for smart communities, in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (2018), pp. 498–499
21. K. Mišura, M. Žagar, Data marketplace for internet of things, in *2016 International Conference on Smart Systems and Technologies (SST)* (IEEE, Piscataway, 2016), pp. 255–260
22. P. Voigt, A. Von dem Bussche, *The EU General Data Protection Regulation (GDPR)*, vol. 18. (Springer, Berlin, 2017)
23. H. Kelly, California passes strictest online privacy law in the country. https://money.cnn.com/2018/06/28/technology/california-consumer-privacy-act/index.html. Accessed 14 Aug 2018
24. J.S. Baik, Data privacy against innovation or against discrimination?: The case of the california consumer privacy act (CCPA). Telematics Informat. **52** (2020). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3624850
25. S.J. Johnston, P.J. Basford, F.M.J. Bulot, M. Apetroaie-Cristea, N.H.C. Easton, C. Davenport, G.L. Foster, M. Loxham, A.K.R. Morris, S.J. Cox, City scale particulate matter monitoring using lorawan based air quality iot devices. Sensors **19**(1), 209 (2019)
26. F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani, M. Petracca, Industrial internet of things monitoring solution for advanced predictive maintenance applications. J. Ind. Informat. Integr. **7**, 4–12 (2017)
27. G.S. Ramachandran, J. Stout, J.J. Edson, B. Krishnamachari, Parkingjson: an open standard format for parking data in smart cities. Open J. Int. Things **6**(1), 105–118 (2020)

28. D.J. Solove, HIPAA turns 10: analyzing the past, present, and future impact. (2013)
29. K.C. Dey, L. Yan, X. Wang, Y. Wang, H. Shen, M. Chowdhury, L. Yu, C. Qiu, V. Soundararaj, A review of communication, driver characteristics, and controls aspects of cooperative adaptive cruise control (CACC). IEEE Trans. Intell. Transport. Syst. **17**(2), 491–509 (2016)
30. G.S. Ramachandran, X. Ji, P. Navaney, L. Zheng, M. Martinez, B. Krishnamachari, Micropayments for trusted vehicular services using motive (video), in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (2019), pp. 701–702
31. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. Technical Report, Manubot (2019)
32. J. Kwon, Tendermint: Consensus without mining. Draft v. 0.6 fall **1**(11), 1–11 (2014)
33. E. Buchman, Tendermint: Byzantine fault tolerance in the age of blockchains. PhD Thesis, 2016
34. A. Banks, R. Gupta, MQTT version 3.1.1. OASIS Standard **1**, 29 (2014)
35. Apache Kafka (2018). https://kafka.apache.org/
36. A. Videla, J.J.W. Williams, *RabbitMQ in Action: Distributed Messaging for Everyone* (Manning, Shelter Island, 2012)
37. G.S. Ramachandran, K. Wright, L. Zheng, P. Navaney, M. Naveed, B. Krishnamachari, J. Dhaliwal. Trinity: A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence, in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2019), pp. 227–235
38. B. Krishnamachari, J. Power, S.H. Kim, C. Shahabi, I3: An IoT marketplace for smart communities, in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18 (ACM, New York, 2018), pp. 498–499. ISBN 978-1-4503-5720-3. https://doi.org/10.1145/3210240.3223573
39. G.S. Ramachandran, S.L.G. Contreras, B. Krishnamachari, Publish-pay-subscribe protocol for payment-driven edge computing, in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)* (USENIX Association, Renton, 2019)
40. A. Asgaonkar, B. Krishnamachari, Solving the buyer and seller's dilemma: A dual-deposit escrow smart contract for provably cheat-proof delivery and payment for a digital good without a trusted mediator, in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2019), pp. 262–267
41. R. Radhakrishnan, B. Krishnamachari, Streaming data payment protocol for the internet of things, in *The 1st International Workshop on Blockchain for the Internet of Things, Held in Conjunction with IEEE Blockchain* (2018)
42. R. Radhakrishnan, G.S. Ramachandran, B. Krishnamachari, SDPP: Streaming data payment protocol for data economy, in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (IEEE, Piscataway, 2019), pp. 17–18
43. A. Ouaddah, A. Abou Elkalam, A. Ait Ouahman, Fairaccess: a new blockchain-based access control framework for the internet of things. Security Commun. Netw. **9**(18), 5943–5964 (2016)
44. J. Lin, Z. Shen, C. Miao, Using blockchain technology to build trust in sharing lorawan IoT, in *Proceedings of the 2nd International Conference on Crowd Science and Engineering*, ICCSE'17 (Association for Computing Machinery, New York, 2017), pp. 38–43. ISBN 978-14-50353-75-5. https://doi.org/10.1145/3126973.3126980
45. A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, Blockchain for iot security and privacy: The case study of a smart home, in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2017), pp. 618–623 https://doi.org/10.1109/PERCOMW.2017.7917634
46. C. Perera, C.H. Liu, S. Jayawardena, The emerging internet of things marketplace from an industrial perspective: A survey. IEEE Trans. Emerg. Topics Comput. **3**(4), 585–598 (2015)
47. Satori: Toward a crypto-economic trading platform using hedera hashgraph. https://glomoinvesting.com/satori-crypto-platform/. Accessed 14 Aug 2018
48. David sønstebø, IOTA data marketplace. https://blog.iota.org/iota-data-marketplace-cb6be463ac7f. Accessed 14 Aug 2018

49. Terbine.io. TERBINE Data Marketplace. https://www.terbine.io/#/ (2020)
50. B. Dageville, T. Cruanes, M. Zukowski, V. Antonov, A. Avanes, J. Bock, J. Claybaugh, D. Engovatov, M. Hentschel, J. Huang, A.W. Lee, A. Motivala, A.Q. Munir, S. Pelley, P. Povinec, G. Rahn, S. Triantafyllis, P. Unterbrunner, The snowflake elastic data warehouse, in *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16 (Association for Computing Machinery, New York, 2016), pp. 215–226. ISBN 978-14-50335-31-7. https://doi.org/10.1145/2882903.2903741
51. QueXopa. QueXopa Latin America Insights. https://quexopa.io/alternative-data/ (2020)
52. G.S. Ramachandran, R. Radhakrishnan, B. Krishnamachari, Towards a decentralized data marketplace for smart cities, in *2018 IEEE International Smart Cities Conference (ISC2)* (2018), pp. 1–8
53. G.S. Ramachandran, R. Radhakrishnan, B. Krishnamachari, Towards a decentralized data marketplace for smart cities, in *2018 IEEE International Smart Cities Conference (ISC2)* (IEEE, Piscataway, 2018), pp. 1–8
54. K.R. Özyilmaz, M. Doğan, A. Yurdakul, IDMoB: IoT data marketplace on blockchain, in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (IEEE, Piscatway, 2018), pp 11–19
55. Ocean Protocol, Ocean protocol: A decentralized substrate for ai data & services-technical whitepaper. Technical Report, Tech. Rep., 2018. International Journal of Aerospace Engineering Hindawi www . . . (2018)
56. Streamr, Streamr Data Marketplace. https://streamr.network/ (2020)
57. R.L. Rivest, L. Adleman, M.L. Dertouzos, On data banks and privacy homomorphisms. Found. Secure Comput. **4**(11), 169–180 (1978)
58. M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical?, in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11 (ACM, New York, 2011), pp. 113–124. ISBN 978-1-4503-1004-8. https://doi.org/10.1145/2046660.2046682. http://doi.acm.org/10.1145/2046660.2046682
59. A.C. Yao, Protocols for secure computations, in *23rd Annual Symposium on Foundations of Computer Science, 1982. SFCS'08* (IEEE, Piscataway, 1982), pp. 160–164
60. S. Goldwasser, Multi party computations: Past and present, in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '97 (ACM, New York, 1997), pp. 1–6. ISBN 0-89791-952-1. https://doi.org/10.1145/259380.259405. http://doi.acm.org/10.1145/259380.259405
61. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems. SIAM Journal on computing **18**(1), 186–208 (1989)
62. E. Ben-Sasson, A. Chiesa, E. Tromer, M. Virza, Succinct non-interactive zero knowledge for a von neumann architecture, in *23rd USENIX Security Symposium (USENIX Security 14)* (USENIX Association, San Diego, 2014), pp. 781–796. ISBN 978-1-931971-15-7. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson
63. R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, D. Song, Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution (2018). Preprint arXiv:1804.05141
64. O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, M. Costa, Oblivious multi-party machine learning on trusted processors, in *25th USENIX Security Symposium (USENIX Security 16)* (USENIX Association, Austin, 2016), pp. 619–636. ISBN 978-1-931971-32-4. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/ohrimenko.
65. G. Wood, Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**, 1–32 (2014)
66. J. Teutsch, C. Reitwießner, A scalable verification solution for blockchains (2017). https://people.cs.uchicago.edu/teutsch/papers/truebit.pdf
67. C. Cachin, Architecture of the hyperledger blockchain fabric, in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, vol. 310 (2016)