Blockchains and Overlay Networks (BC-OLN)

Prof. Dr. Burkhard Stiller, Dr. Bruno Rodrigues
Communication Systems Group CSG
Department of Informatics Ifl, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Phone: +41 44 635 6710
FAX: +41 44 635 6809
E-Mail: stiller@ifi.uzh.ch

Universität Zürich[UZH]

Assistants:
Christian Killer, Jan von der Assen
Phone: +41 44 635 4378, FAX: +41 44 635 6809
E-Mail: [killer¦vonderassen]@ifi.uzh.ch

CSG Communication Systems Group

© 2022 UZH, CSG@Ifl

Motivation (1)

- What are "Blockchains"?
  - Distributed Ledgers based on Overlay Network technology
  - A "technology" for cryptographic currencies
- What are "Overlay Networks"?
  - Depends on whom you ask!
- Overlay networks in general
  - All telecommunication transport networks and IP networks over fibers
  - Akamai networks for reliability and improved access times
- Overlay networks in case of P2P in the media
  - Illegal file sharing, copyright infringement

© 2022 UZH, CSG@IfI

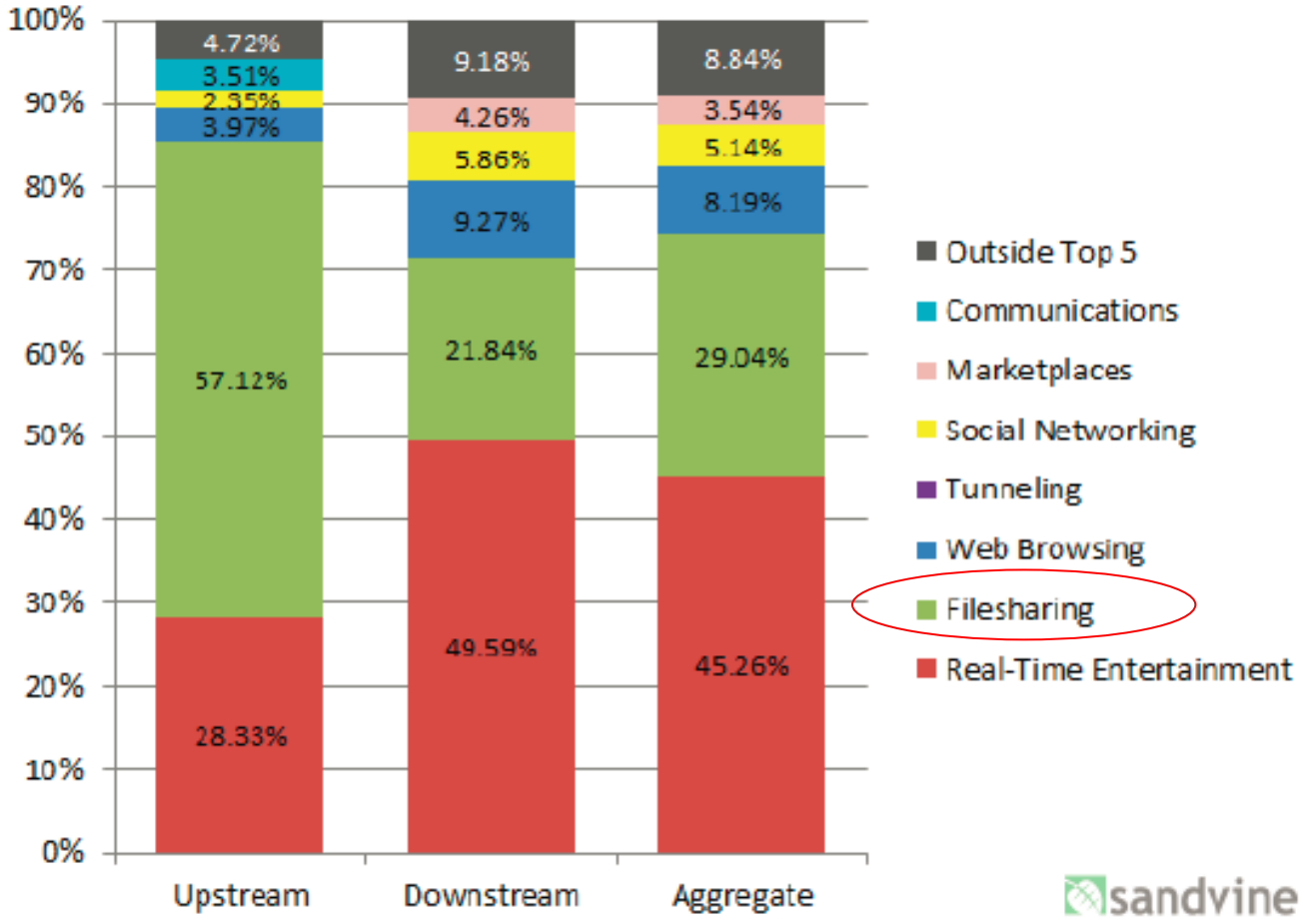
Motivation (2)

- Overlay networks in case of P2P in academia
  - A key paradigm for communications in/on top of the Internet
- Overlay networks in case of Blockchains
  - The essential underlying communication protocol to exchange transactions and consensus results
- Are Overlay Networks really new?
  - The Internet in the late 60s (ARPANET) was designed as a decentralized system, especially as a Peer-to-peer system without any centralized control
  - Only the emergence of the Domain Name System (DNS) and the World Wide Web (WWW) changed the Internet into a decentralized Client/Server system with centralized control

© 2022 UZH, CSG@IfI

Peak Period Traffic Composition (2016)

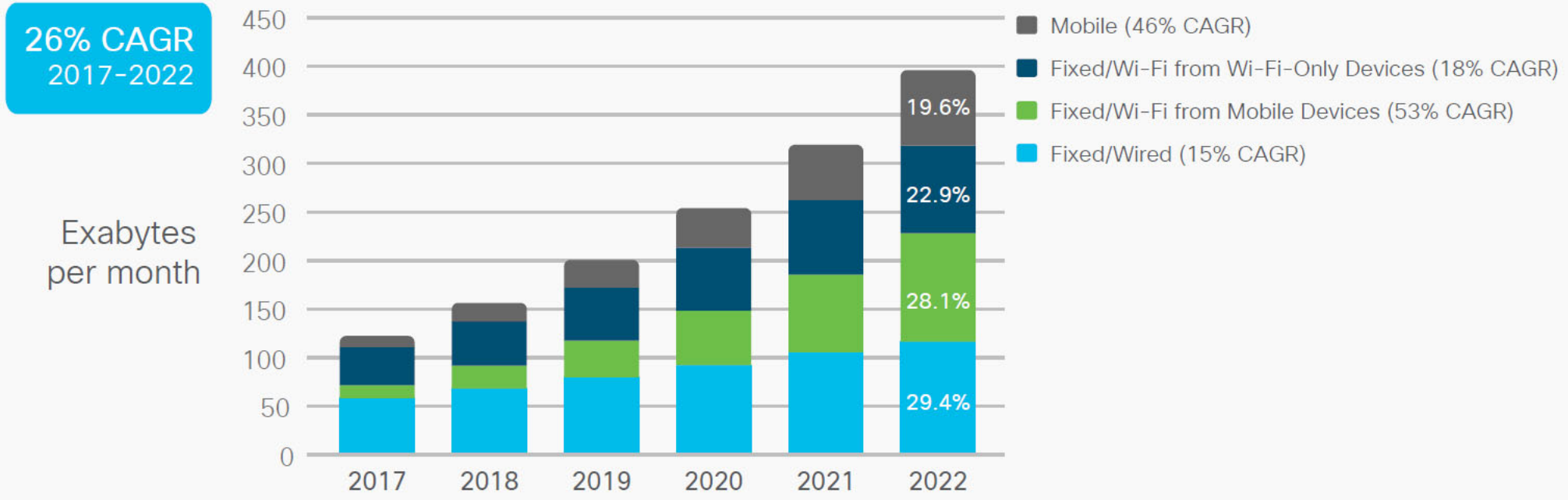
- Asia-Pacific Region, fixed networks



https://www.sandvine.com/resources/global-internet-phenomena/2016/asia-pacific-africa-and-the-middle-east.html

Global IP Traffic (2017-2022)

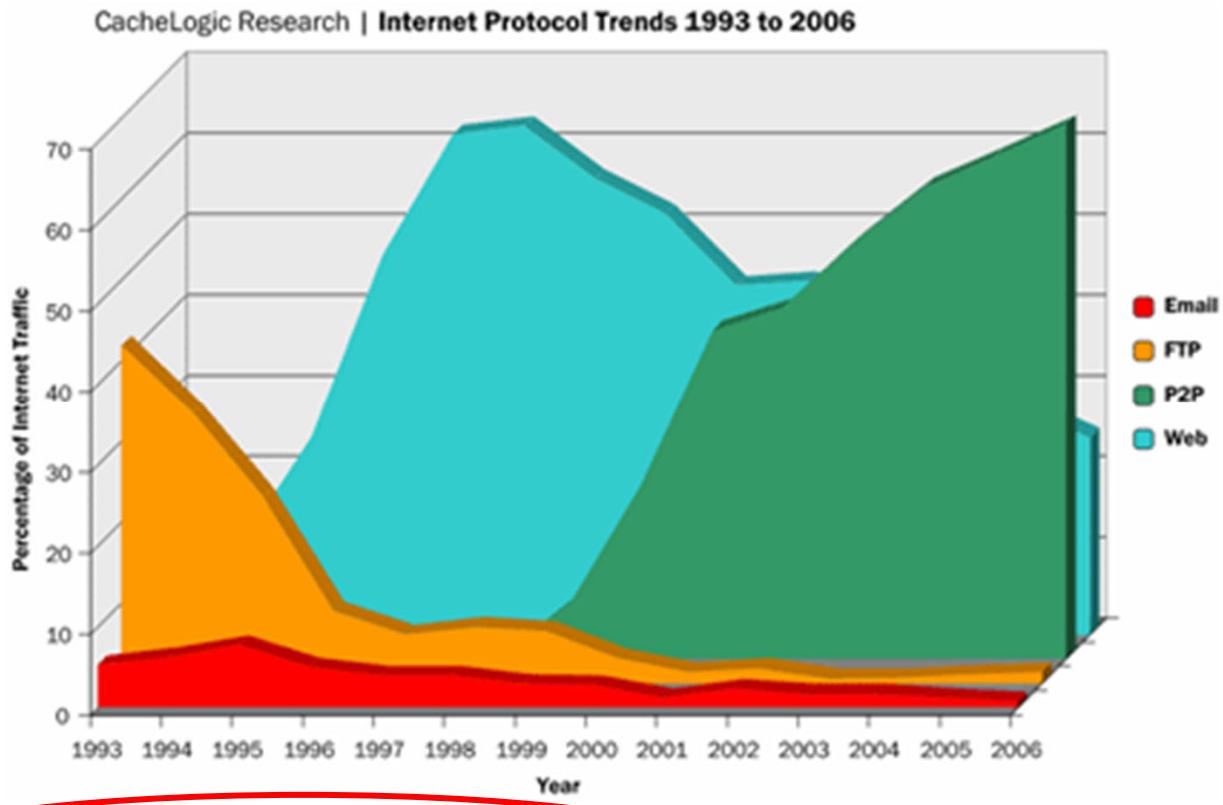
- 2017: wired networks at 48%, Wi-Fi at 43%, and mobile/cellular networks at 9% of total global IP traffic
- 2022: wired networks 29% of IP traffic, Wi-Fi and mobile networks 71% of IP traffic



https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

© 2022 UZH, CSG@IfI

P2P File Sharing Traffic (Past)

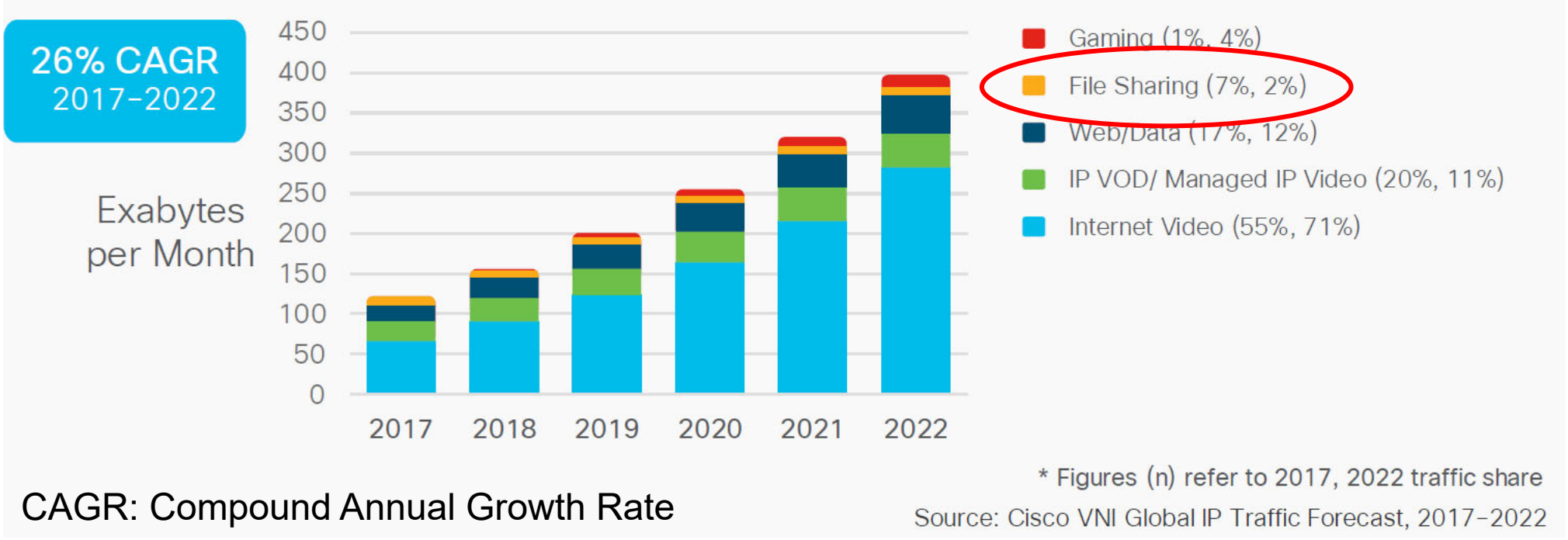


- 70% of Internet traffic was due to P2P applications!
  - Starting from 2000 until approximately 2010+

Source: CacheLogic Research

P2P File Sharing Traffic (Current/Future)

- Sum of all forms of IP video
  - Including Internet video, IP VoD, video files exchanged through file sharing, video-streamed gaming, and video conferencing,



CAGR: Compound Annual Growth Rate

https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

P2P File Sharing Traffic Details (2011)

Table 6 - Europe, Fixed Access, Peak Period, Top Applications by Bytes

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	59.68%	BitTorrent	21.63%	BitTorrent	28.40%
2	Skype	7.16%	HTTP	20.47%	HTTP	18.08%
3	HTTP	7.02%	YouTube	14.13%	YouTube	11.93%
4	PPStream	3.64%	RTMP	4.58%	RTMP	3.90%
5	Spotify	2.91%	Flash Video	3.99%	Flash Video	3.38%
6	SSL	2.66%	iTunes	3.65%	SSL	3.09%
7	eDonkey	1.76%	SSL	3.18%	iTunes	3.07%
8	YouTube	1.76%	NNTP	2.73%	Skype	2.44%
9	Facebook	1.42%	Facebook	1.71%	NNTP	2.30%
10	Teredo	1.18%	Skype	1.42%	PPStream	1.77%
	Top 10	89.19%	**Top 10**	77.49%	**Top 10**	78.36%

SOURCE: SANDVINE NETWORK DEMOGRAPHICS

sandvine

P2P File Sharing Traffic Details (2014)

- A decade ago

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	Facebook	22.36%	YouTube	19.75%	Facebook	19.43%
2	Google Cloud	11.97%	Facebook	19.05%	YouTube	18.02%
3	HTTP	9.85%	HTTP	11.44%	HTTP	11.26%
4	SSL	9.22%	MPEG - OTHER	6.32%	MPEG - OTHER	5.72%
5	YouTube	4.56%	Netflix	4.51%	SSL	4.63%
6	Instagram	2.55%	Instagram	4.49%	Instagram	4.27%
7	Snapchat	1.94%	SSL	4.03%	Netflix	4.10%
8	BitTorrent	1.88%	iTunes	3.20%	Google Cloud	4.09%
9	FaceTime	1.59%	Google Cloud	3.07%	iTunes	2.96%
10	Skype	1.53%	Pandora Radio	2.72%	Pandora Radio	2.53%
		67.44%		78.57%		77.02%

http://broabandtrafficmanagement.blogspot.ch/2013/11/sandvine-na-fixed-traffic.html

P2P File Sharing Traffic Details (2016)

- More recent

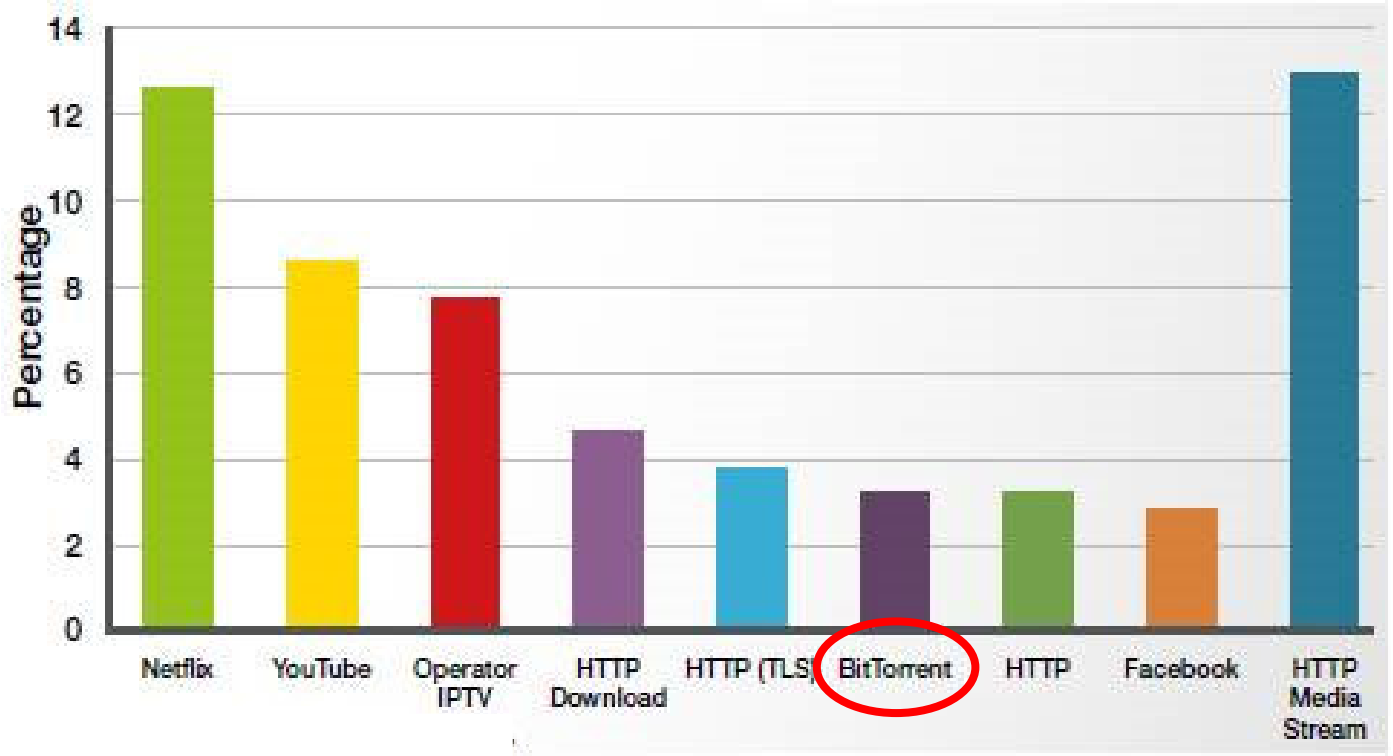
	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	HTTP (Other)	11.84%	HTTP (Other)	22.70%	HTTP (Other)	20.54%
2	WhatsApp	7.94%	YouTube	11.78%	YouTube	9.87%
3	Facebook	6.85%	WhatsApp	6.83%	WhatsApp	7.42%
4	SSL (Other)	5.77%	Facebook	6.38%	Facebook	6.14%
5	Skype	3.82%	Google Market	5.96%	Google Market	3.98%
6	BitTorrent	3.22%	BitTorrent	4.77%	SSL (Other)	3.95%
7	Viber	3.18%	Opera Mini	3.77%	BitTorrent	3.66%
8	BlackBerry	3.05%	BlackBerry	3.65%	Opera Mini	3.30%
9	Opera Mini	2.25%	SSL (Other)	3.51%	BlackBerry	3.25%
10	YouTube	1.74%	WAP v2	2.93%	WAP v2	2.69%
		49.66%		72.29%		64.80%

http://broabandtrafficmanagement.blogspot.ch/2013/11/sandvine-na-fixed-traffic.html

Global Application Traffic Share (2019)

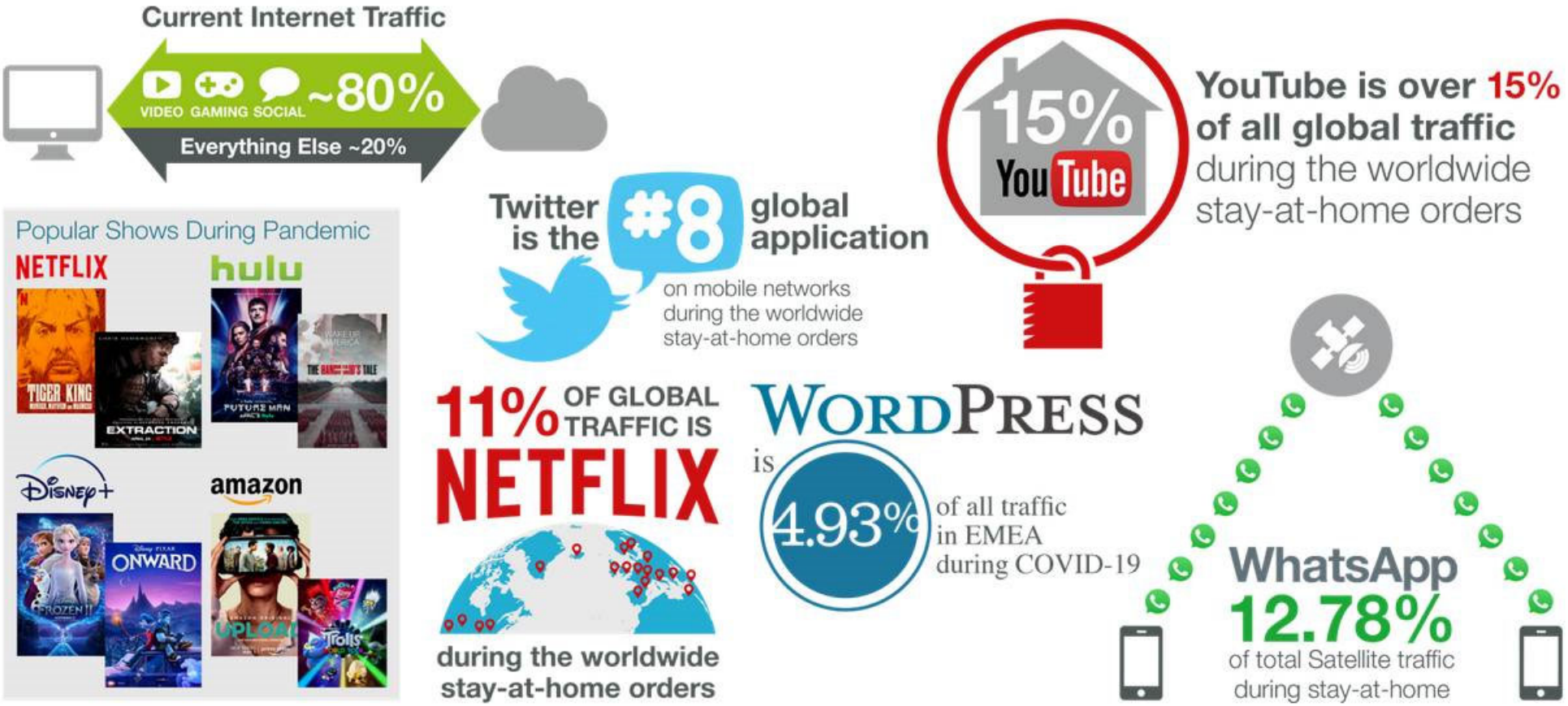
GLOBAL APPLICATION TRAFFIC SHARE

1	HTTP MEDIA STREAM	12.8%(-0.1)	6.9%(+1.4)
2	NETFLIX	12.6%(-2.3)	2.1%(-0.8)
3	YOUTUBE	8.7%(-2.9)	2.4%(-0.6)
4	OPERATOR IPTV	7.2%(+2.8)	3.5%(-0.6)
5	HTTP DOWNLOAD	4.5%(+0.8)	1.0%(-0.9)
6	HTTP (TLS)	4.4%(+0.4)	2.2%(+0.1)
7	HTTP	3.9%(-0.1)	3.5%(-1.2)
8	FACEBOOK	3.0%(+1.2)	2.2%(+0.7)
9	PLAYSTATION DOWNLOAD	2.8%(+0.2)	0.4%(+0.0)
10	BITTORRENT	2.4%(+0.2)	27.5%(+6.4)



https://www.sandvine.com/global-internet-phenomena-report-2019

COVID-19 Global Internet Phenomena



https://www.sandvine.com/phenomena

Blockchains

- A "decentralized database" with P2P data storage
  - Digital record of who-owns-what w/o a central storage
    - Ledger is replicated among nodes in a distributed network
- Consensus algorithm ensures that each node's copy of the ledger is identical to every other node's copy
- Access to ledger from asset owners via cryptographic signatures to persist …
  - … "incoming" data (assets) on their private or a public ledger
  - … "outgoing" data to any other stakeholder
- Advantages of distributed ledgers:
  - Unforgeable, immutable, and preventing "double spending"
  - The unknown: public/governmental acceptance

© 2022 UZH, CSG@IfI

Lecture Goals

- To provide an in-depth insight into Overlay Networks, *e.g.*, Peer-to-peer networks, and Blockchains
  - Fully decentralized communication paradigm
    - Alternative to client/server paradigm
  - Benefits and drawbacks
    - Scalability, security, efficiency, fault-tolerance, manageability
  - Basic concepts and approaches
    - Structured vs. unstructured overlay systems, Distributed Hash Tables
    - Public/private (consortium-based) Blockchains and their applications (*e.g.*, Fintech and Food or Supply Chains)
  - Related systems
    - Grids, Mobile P2P systems, Cloud Computing

Lecture Content

- Introduction and Introduction
- Overlay Generations
- Distributed Hash Tables (DHT)
- Blockchain Basics
- Blockchain Areas, App Domains, Consensus Mechanisms
- Cryptocurrencies and Bitcoin
- Ethereum, Smart Contracts, Solidity
- Blockchains' Probabilities and Security
- Blockchain Interoperability
- Blockchain-based eVoting
- Other Blockchains
- Selected P2P and Blockchain Projects and Approaches
- Blockchain Evaluations

Detailed order may change!

© 2022 UZH, CSG@IfI

Teaching Aids and Exercises

- Basics and background information available within:
  - Journal, conference, or workshop papers
  - Web sites and white papers
  - Selected chapters from books
  - Code examples
- Additional references within lecture slides and at modules' end
- Exercises: lecture includes 2 types of exercises
  - Theoretical part (4 exercises) and
  - Practical part (Challenge Task, CT)

Exercises, Organizational Issues, On-line Information

- Exercises on a per student and group basis:
  - Knowledge and background questions reflecting the lecture
  - Practical tasks and application of lecture content
- Exercises Thursdays: Time: 14.00 – 15.45 hours
- Lectures Tuesdays: Time: 12.15 – 13.45 hours
- http://www.csg.uzh.ch/teaching/fs22/p2p.html/
  - In case of questions, concerns, help, or ideas concerning the lecture please contact by e-mail the assistants (see cover) or Prof. Dr. Burkhard Stiller and talk to them (e-mail)
  - This Web site is used for all material (slides, exercises)

Introduction and Basic Details



Universität Zürich[UZH]

CSG
Communication Systems Group

Slides include material and inputs from David Hausheer (University of Zürich, Department of Informatics, Communication Systems Group CSG, Switzerland, now OVGU Magdeburg, Germany) and Detlef Schoder (University of Cologne, Germany).

© 2022 UZH, CSG@IfI

ifi

ToC

- Introduction
  - Definitions and Characteristics
  - Classification and Properties
- Application Areas and Information
  - Definition and Classification
  - Information Types
- File Sharing
  - Centralized Directory Model
  - Flooded Request Model
  - Structured Routing Model
- Performance Factors
  - Bandwidth, Storage, Processor Cycles

© 2022 UZH, CSG@IfI

P2P (Peer-to-peer)

Definitions and Characteristics

Definitions and Characteristics (1)

- Overlay Network (ON)
  - is a computer network built on the top of another network.
  - Nodes in the overlay network are connected by virtual or logical links.
  - Such links correspond to a path, possibly through many physical links, in the underlying network.
- Decentralized System

Based on Wikipedia

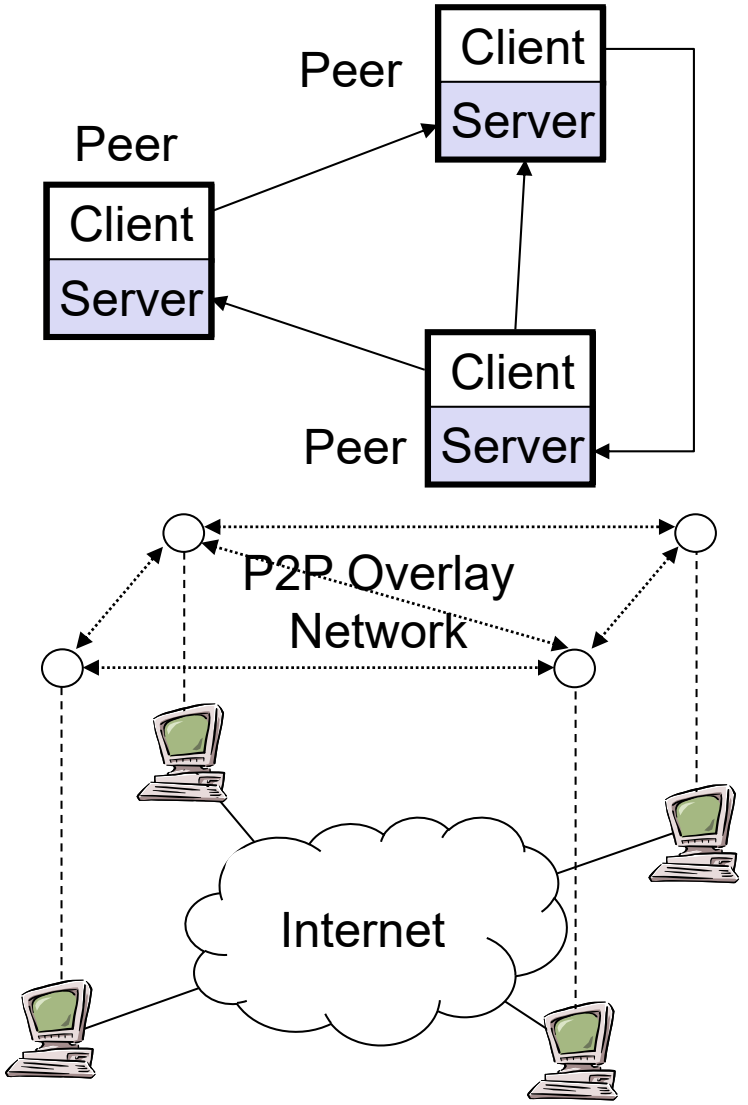
  - is a system in which lower level components operate on local information to accomplish global goals.
  - Global pattern of behavior is an emergent property of dynamic mechanisms that act upon local components, such as indirect communication, rather than a result of a central ordering influence.

Definitions and Characteristics (2)

- Oram et al.:
  - A Peer-to-Peer (P2P) system is *„a self-organizing system of equal, autonomous entities (peers)* [which] *aims for the shared usage of distributed resources in a networked environment avoiding central services.“*
- http://dictionary.com:
  - *Peer-to-peer; denoting a network or data communications in which no dedicated server is involved*.
- Derived key characteristics of a P2P system:
  - Equality – All peers are equal (peer = „gleichgestellt“)
  - Decentralization – No centralized services
  - Self-organization – No coordination from outside
  - Shared resources – Peers use resources provided by other peers
  - Direct interaction – Peers communicate directly with other peers

Definitions and Characteristics (3)

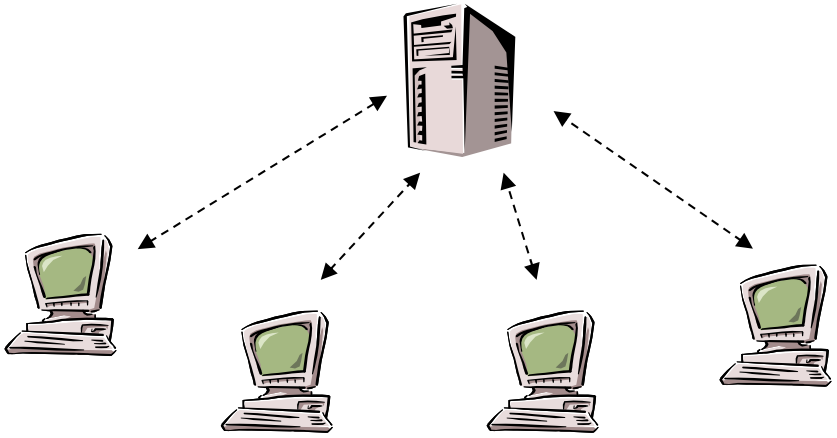
- Peers
  - Have all the same capabilities (ability to act in any role)
  - Can act as ”clients“ and “servers“ at the same time
- Overlay (Peer-to-Peer) Network
  - Composed of direct connections between peers
  - Addressing scheme different than the underlay network
  - Typically, an “overlay“ network on top of a network (*e.g.*, Internet)



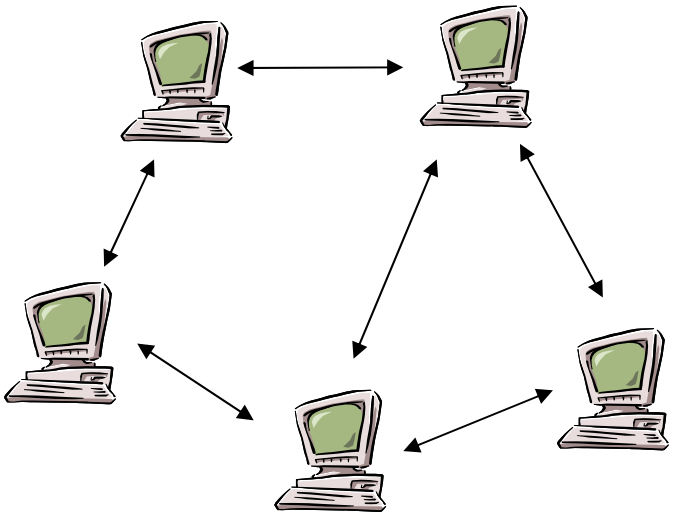
Desirable Properties of an ON

- Fault-tolerance
  - Churn
  - Connection problems
  - Malicious behavior
- Heterogeneity
  - Mobile devices vs. powerful devices, low vs. high bandwidth, battery vs. connected to power source
- Fairness
  - Each peer should have a similar workload (exceptions!)
- Scalability
  - Workload proportional to number of peers
  - Ideally, no limit on the number of peers

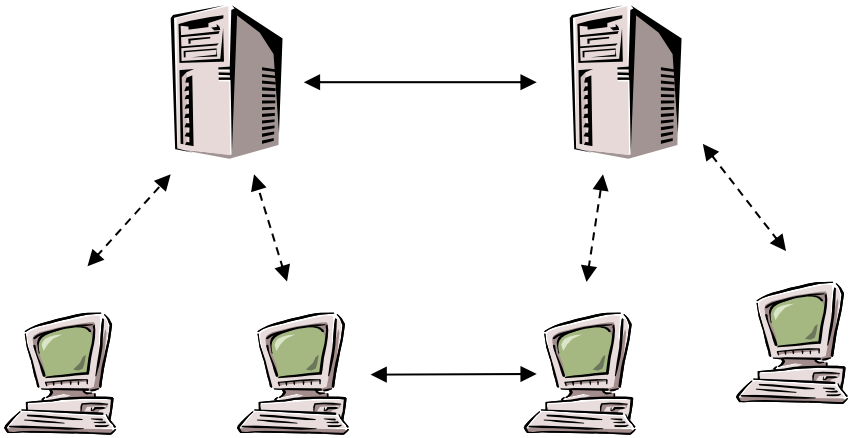
Classification of P2P Systems (1)



Client/Server System



Pure Peer-to-Peer System



Hybrid System



Server or Superpeer

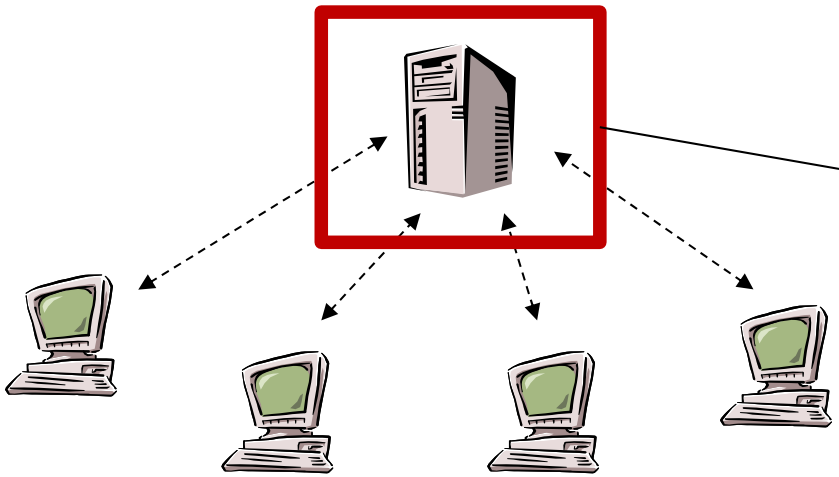


Peer or Client

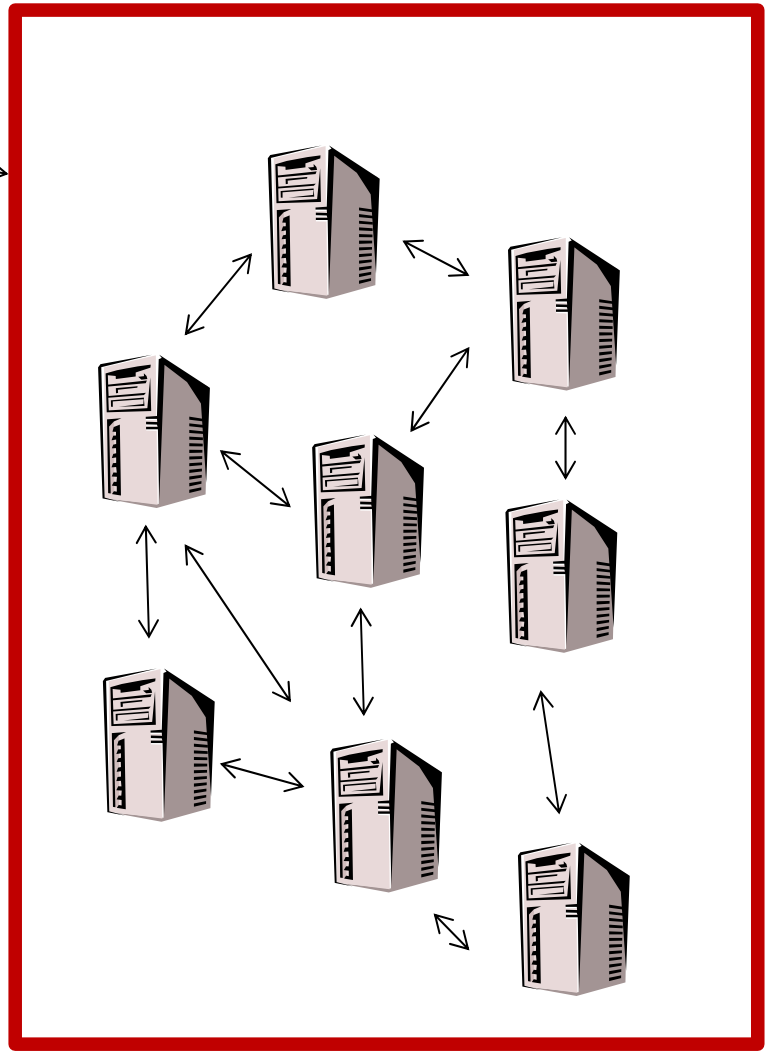
C/S-Relation

P2P-Relation

Classification of P2P Systems (2)

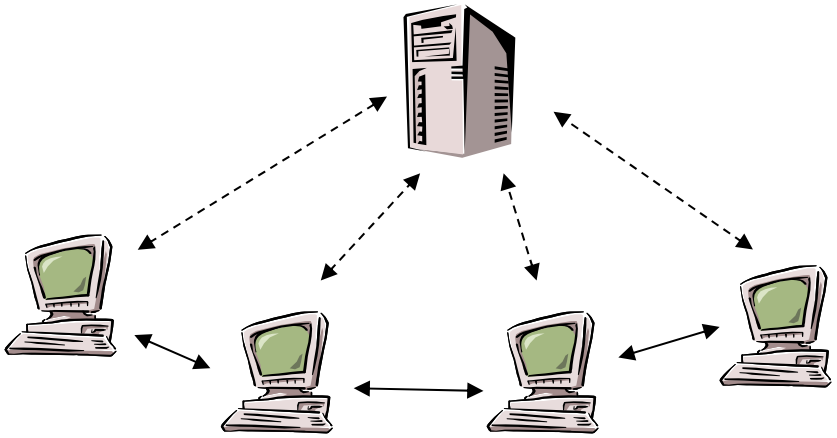


Client/Server System

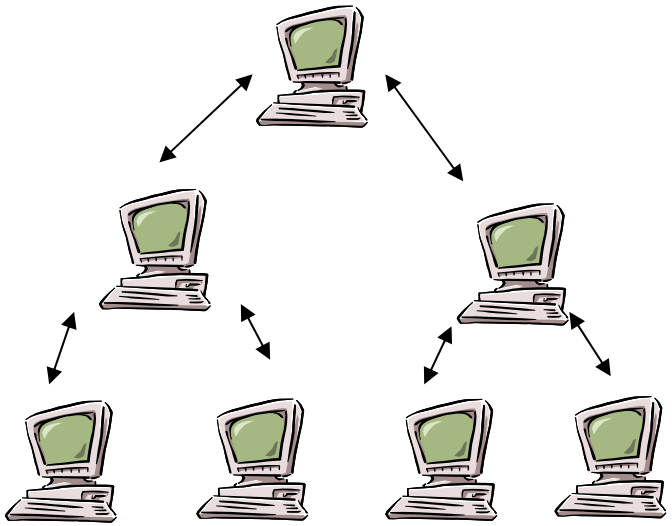


- Servers are often not a single machine, but a distributed system
  - Can use P2P mechanisms to organize
  - Often uses master/coordination node

Classification of P2P Systems (3)



Centralized P2P System



Structured P2P System

- Structured vs. unstructured
  - Identifier (resource/peer)
  - Find peer to store resources
- Main task
  - Find resources (and where to put)



Server or Superpeer



Peer or Client

C/S-Relation

P2P-Relation

Properties – Client/Server vs. P2P Systems

Properties	Description	C/S	P2P
Manageability	How hard is it to keep the system working?	+	-
Information coherence	How authoritative is information in the system?	+	-
Fault-tolerance	How well can the system handle failures?	-	+
Security	How hard is it to subvert the system?	+	-
Scalability	How large can the system grow?	+/-	+

Adapted table, original source: N. Minar: *Distributed Systems Topologies: Part 2*; O'Reilly Open P2P, December 2001.

- Legal aspects, P2P vs. C/S (take-down)
  - Not always part of this lecture

Application Areas and Information

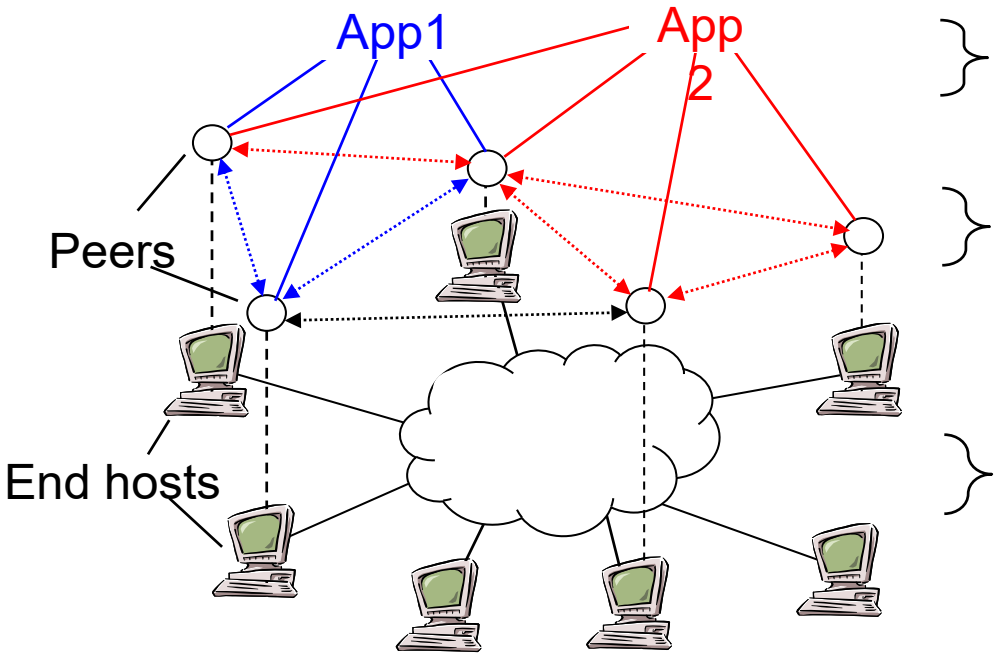
Definitions and Characteristics

Information Types

© 2022 UZH, CSG@IfI

Definition of Overlay Applications

- Overlay Applications
  - Application of an overlay network for a specific purpose
  - Drawing on the cooperation of peers in the form of services these peers provide each other in order to achieve a common goal



P2P Application
e.g., KaZaA, eMule, BitTorrent

P2P Overlay Network
e.g., random, structured, hybrid,

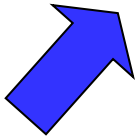
Underlying Network
e.g., Internet

Overlay Application Classifications

- Conventional Classification of Overlay Applications
  - File Sharing (*e.g.*, Napster, Gnutella, Freenet)
  - Social networking (Social-P2P, Diaspora*, Frendica)
  - Instant Messaging (ICQ, AIM, Skype)
  - P2P Currency (Bitcoin)
  - P2P TV/Streaming

- Classification by Means of Shared Resources
  - Traffic / Bandwidth
  - Information / Files
  - Storage space
  - Processor cycles





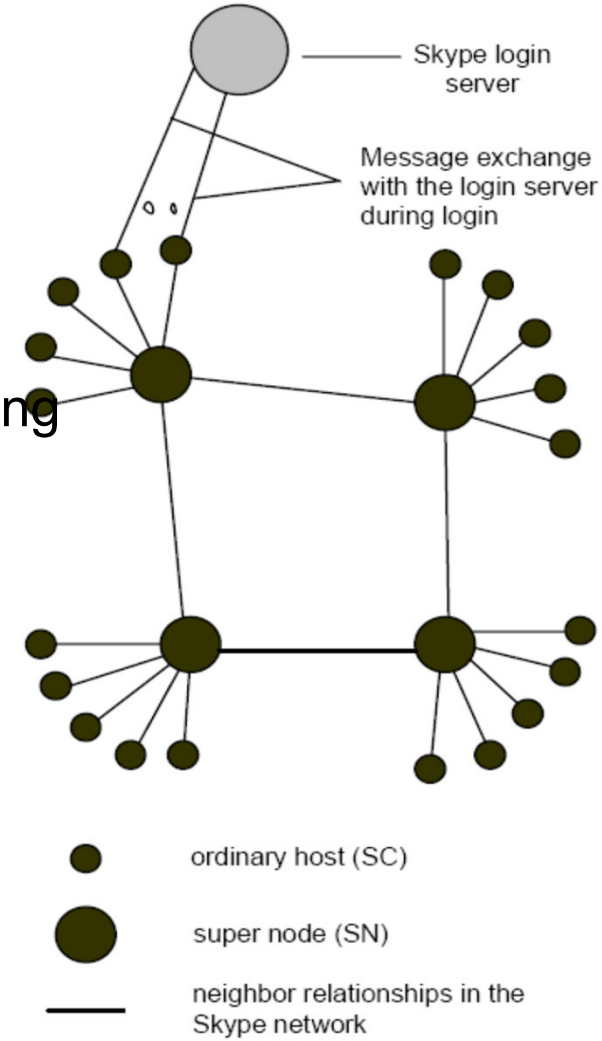
- No clear distinction
- Some cases even misleading

Information (1)

- Social-P2P: integrates a social network into a P2P network for efficient and trustworthy file sharing
  - Integrating issues of efficient and trustworthy searching
    - (1) interest/trust-based structure
    - (2) interest/trust-based file searching
    - (3) trust relationship adjustment
- Social Networking
  - Friendica (PHP/MySQL/Apache) ~ 8200 users
  - Libertree (Ruby - alpha)
    - Founded by two former Diaspora* developers
  - Data center vs. personal devices
    - Router: 700 MHZ MIPS CPU 64 MB RAM, 8 MB Flash - OpenWRT

Information (2)

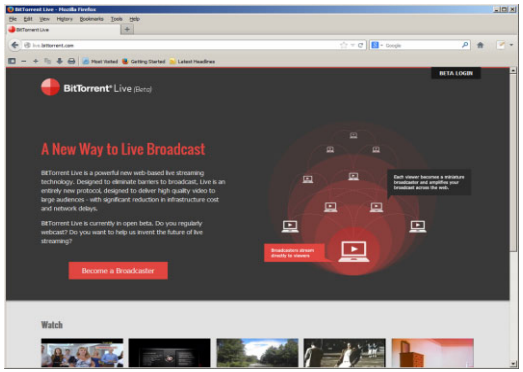
- Skype is a peer-to-peer VoIP client
  - Since 2003 (organization created Kazaa)
    - Skype encrypts calls end-to-end
    - Skype stores user infos decentralized
    - Skype supports instant messaging, conferencing
  - Two node types (see right)
    - Super node: ordinary host's end-point
    - Any node with a public IP address having sufficient CPU, memory, network bandwidth can become a super node
  - Bought by eBay for 2.6B US$ in 2005
  - Bought by Microsoft for 8.5B US$ in 2011
  - Active daily users ~4.9 M and 3 billion minutes used per day



Information (3)

- Media streaming using P2P (P2PTV)
  - Other problems than with file sharing
  - Time matters!
- Zattoo
  - Started as P2P, but due to too small Upload rates in 2009 of DSL replaced by Server stream
- TVUPlayer (shutdown on February 25, 2013), PPLive, QQLive, PPStream, SopCast
  - Mainly used in China
- BitTorrent Live (closed by Feb 28, 2014)
  - Live streaming events (private beta)
  - http://live.bittorrent.com/

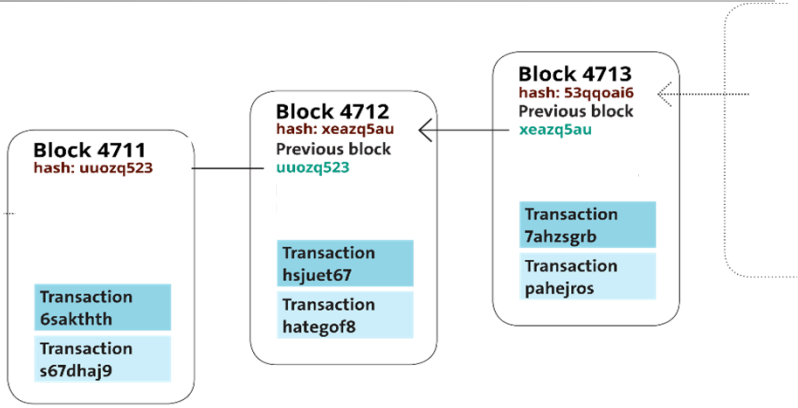
http://zattoo.com/



© 2022 UZH, CSG@IfI

Information (4)

- Transactions collected in blocks
  - New blocks created regularly
- A block contains a pointer to the previous block → Blockchain!



- Consensus mechanism required to determine the block to be integrated into the blockchain
  - Public blocks typically contain solved crypto puzzles
  - *E.g.*, a form of partial hash collisions (SHA256)



- Creation of valid blocks is called mining (reward)
  - Computational expensive → Avoids double spending
  - Mining ≡ confirmation of blocks ≡ solving crypto puzzles



© 2022 UZH, CSG@IfI

File Sharing

Centralized Directory Model

Flooded Request Model

File Sharing

- Store content at individual nodes instead of in one central
- Peers downloading files subsequently make them available for other peers
- File sharing is most widespread P2P application

- Lookup Problem
  - Locating resources is the central problem for P2P networks in general and for file sharing in particular
  - Peer-to-Peer networks provide different methods for storing, searching, and retrieving files:
    - Centralized directory model
    - Flooded requests model
    - Structured routing model

Centralized Directory Model

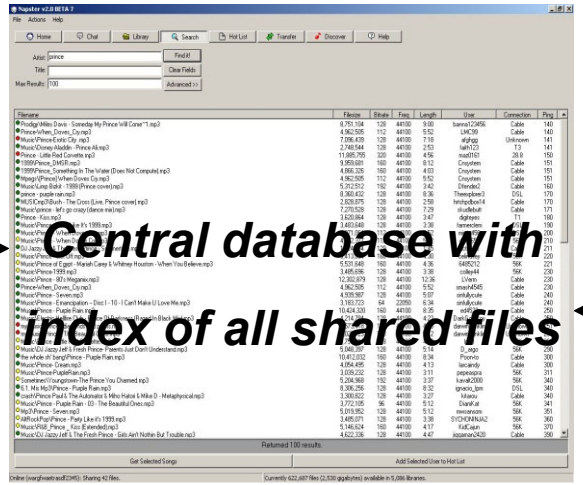
- Perfect example of a hybrid P2P system
  - Index service is provided centrally by a coordinating entity
  - Search request is issued to the coordinating entity which delivers a list of peers having the desired files available for download
  - Requesting peer obtains respective files directly from the peer offering them
- Characteristics
  - Lookup of existing documents can be guaranteed
  - Index service is a “Single Point of Failure”
- Example: “Early” Napster
  - Sharing of music files
  - Switched off by interim injunction

Centralized Directory Model — Example



Joe shares on his client several MP3 files.

Melissa shares on her client several MP3 files, too.



Central database with index of all shared files

? Prince purple rain ?

! Prince purple rain !

! Prince purple rain ! @ Melissa

? Prince purple rain ?

Prince - purple rain

Prince - purple rain

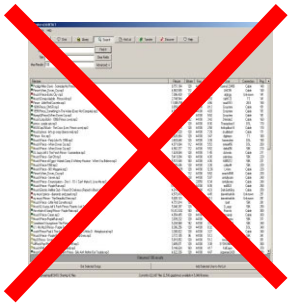
Flooded Request Model

- Atomic P2P system
  - Without central coordination authority (all peers are equal)
  - Search request is passed on to a predetermined number of peers
  - If they cannot answer the request, they pass it on to various other nodes until a predetermined search depth (ttl = time-to-live)
  - Requested file located, positive search results sent to requesting entity
  - Requesting peer can download desired file directly from entity offering
- Characteristics
  - Lookup of existing documents cannot be guaranteed
  - System does not scale
- Example: Gnutella (Extension later as “FastTrack”)
  - “Super Peers” as proxies
  - No atomic P2P system anymore

Flooded Request Model — Example

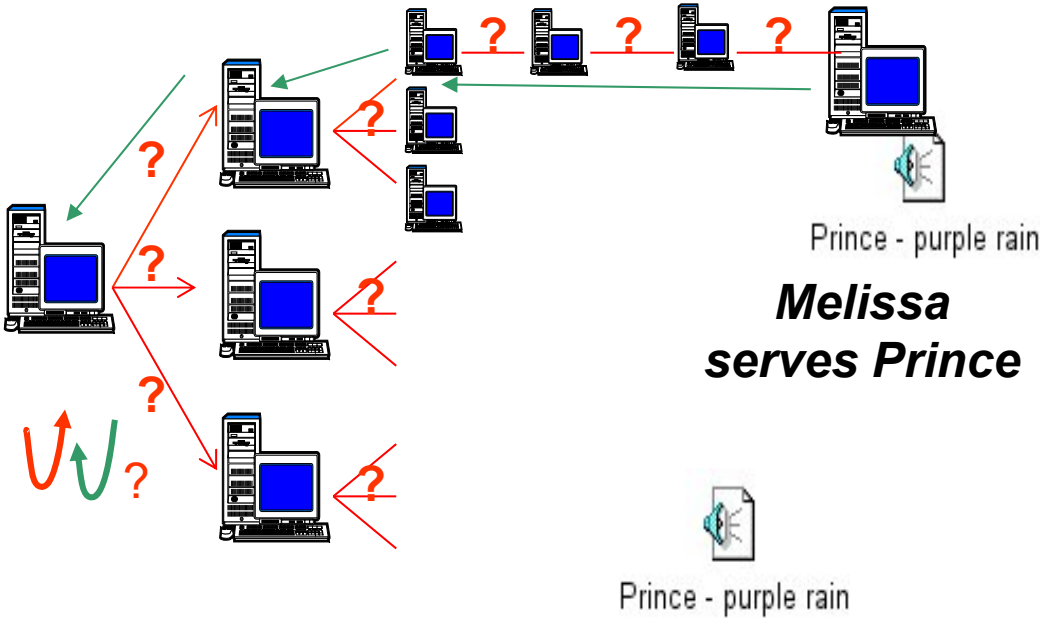


http://www.gnutelliums.com/



No central Database

Joe is searching for Prince



Melissa serves Prince

Structured Routing Model

- Without central coordination authority (all peers are equal)
- Files not stored on hard disk of the peers responsible for them
  - Storing pointers (tracking)
- Files are stored at peers in the tracker
  1. Find tracker
  2. Get peers from tracker
  3. Get data from peers.
- Using a hash function: “Distributed Hash Table (DHT)”

- Characteristics
  - Self organized adaptation in case of entering or leaving peers
- Examples
  - Pastry, Chord, CAN, Tapestry, TomP2P (cf. later in lecture)

Performance Factors

Bandwidth

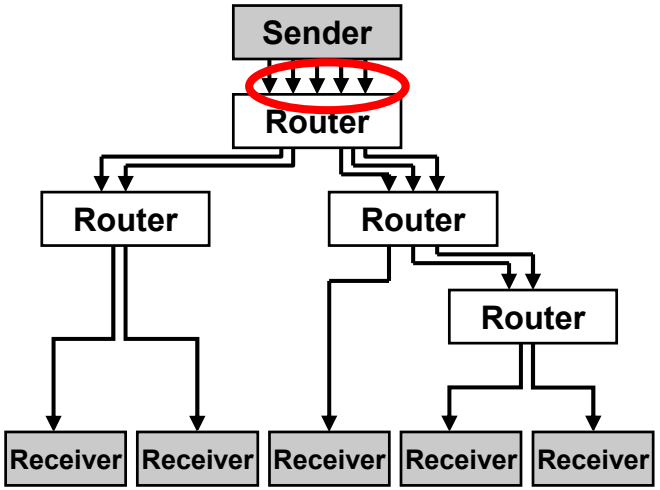
Storage Space

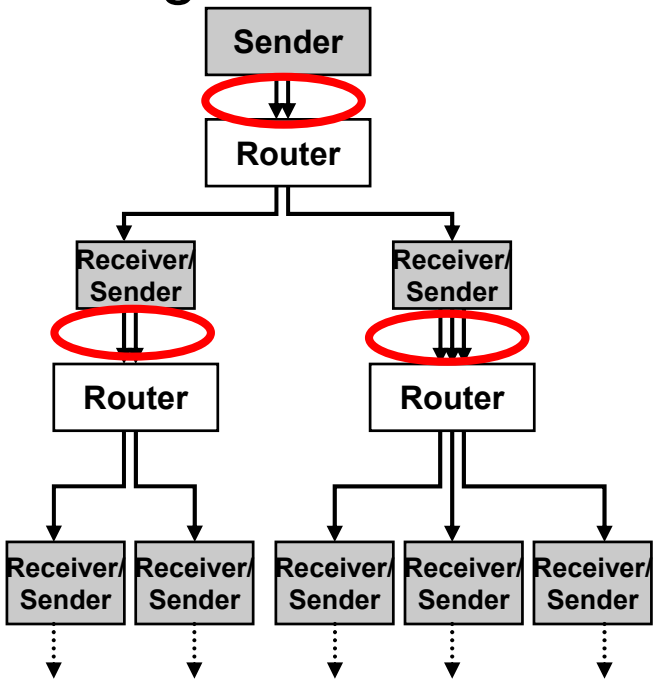
Processor Cycles

Bandwidth

- Typical Centralized Approach
  - Files are held on server of an information provider
  - Files are transferred from there to requesting client

Unicast

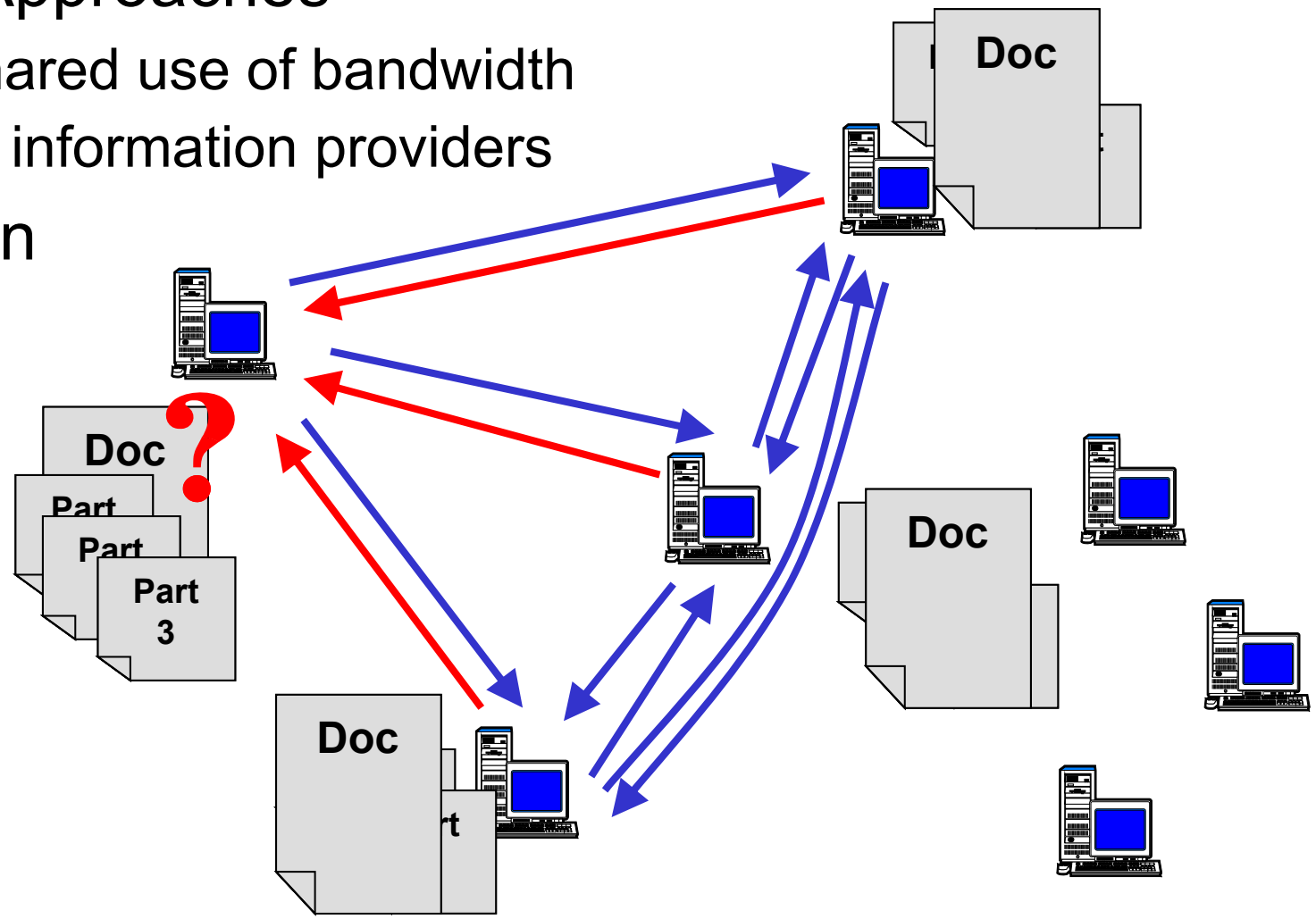




- P2P-Based Approaches
  - Achieve increased load balancing by taking advantage of transmission routes which are not being fully exploited

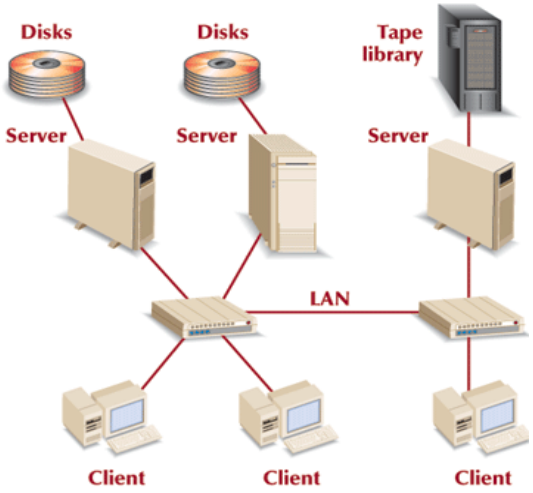
Shared Use of Bandwidth

- P2P-based Approaches
  - Facilitate shared use of bandwidth provided by information providers
- Segmentation Approach

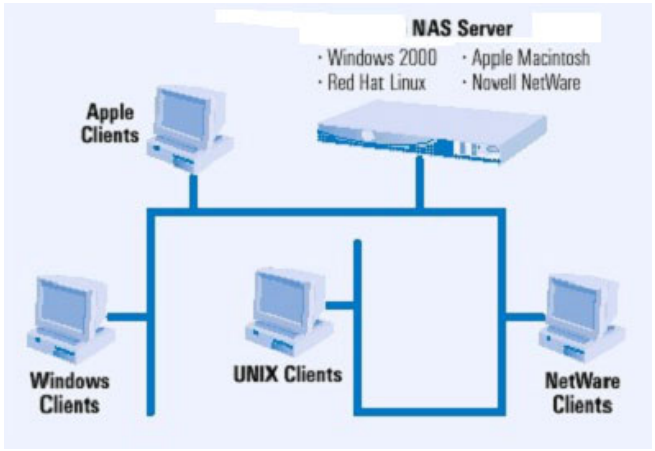


Centralized and P2P Storage

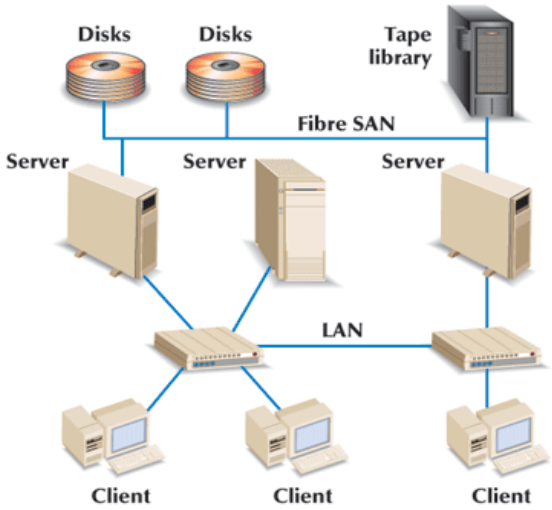
- Centralized design concepts used to store data



Direct Attached Storage (DAS)



Network Attached Storage (NAS)



Storage Area Networks (SAN)

- Advantages of P2P Storage
  - Scalability
  - Fault tolerance built-in (replication/redundancy)
  - May be based on inexpensive commodity HW

Overlay Storage Networks

- Cluster of computers formed on basis of existing networks, sharing all storage available in the network
  - Examples: PAST, Pasta, OceanStore, (Wuala), AeroFS
- Organization
  - Each peer creates a public/private key pair
  - Public key is used to create identification number for each peer (with the aid of a hash function).
  - Each peer must make available its own storage, or pay a fee
  - Corresponding to its contribution, each peer is assigned a maximum volume of data which can be added to the network
  - File is assigned an unambiguous identification number (hash function from name or content and public key of the owner)

Processor Cycles

- Increasing requirements for HPC
  - *I.e.,* in field of bio-informatics, logistics, or financial sector
  - Available computing power of networked entities often unused
- Using P2P applications to bundle processor cycles
  - Forming cluster of independent, networked computers in which a single computer is transparent, networked nodes “logical”
  - Achieving computing power which even the most expensive super-computers can scarcely provide
- Examples
  - Bitcoin: Calculation of coins using GPU (creation of Bitcoins)
  - Bitcoin miners with more processing power (although highly specialized) than all top500 supercomputers of the world

HPC: High Performance Computing

References

- *Distributed Ledger Technology: Beyond Blockchain;* A report by UK Government Chief Scientific Adviser, 2016
- A. Oram: *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*; O'Reilly and Associates, 2001
- R. Steinmetz, K. Wehrle (Eds.): *Peer-to-Peer Systems and Applications;* Springer Publisher, Berlin, Germany, Lecture Notes in Computer Science LNCS, Vol. 3485, 2005 (Chapters 2 and 4)
- Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim: *A Survey and Comparison of Peer-to-peer Overlay Network Schemes;* Com-munications Surveys & Tutorials, Vol. 7, No. 2, 2005, pp 72 - 93
- W. Effelsberg, R. Steinmetz, T. Strufe: *Benchmarking Peer-to-Peer Systems;* Springer Publisher, Berlin, Germany, Lecture Notes in Computer Science LNCS, Vol. 7874, 2013 (Chapters 1, 2, 4, 5, and 9)
- Haiying Shen, Ze Li, and Kang Chen: *Social-P2P: An Online Social Network Based P2P File Sharing System;* IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 10, October 2015
- Pradnya Eknath Gaonkar, Sachin Bojewar, Jayesh Ajit Das: A Survey: *Data Storage Technologies;* International Journal of Engineering Science and Innovative Technology (IJESIT), Vol. 2, No. 2, March 2013

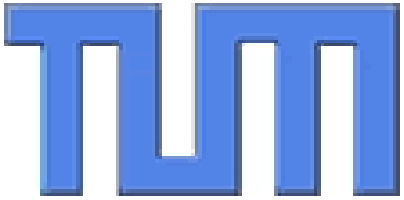
© 2022 UZH, CSG@IfI

Overlay Generations





Universität Zürich[UZH]



Slides based on Rüdiger Schollmeier as well as Jörg Eberspächer (Technische Universität München, Germany).

ToC

- General Characteristics of Overlay Systems
  - 1st and 2nd Generation Architectures, Topology
  - Routing, Bootstrapping
- Centralized P2P Networks
  - Definition, Characteristics, and Topology
  - Example and Discussion: Napster
- Decentralized P2P Networks
  - Definition, Characteristics, and Topology
  - Example and Discussion: Gnutella
- History and Past
  - From the ARPANET to Peer-to-Peer
  - From Peer-to-Peer to Overlay Networks

© 2022 UZH, CSG@IfI

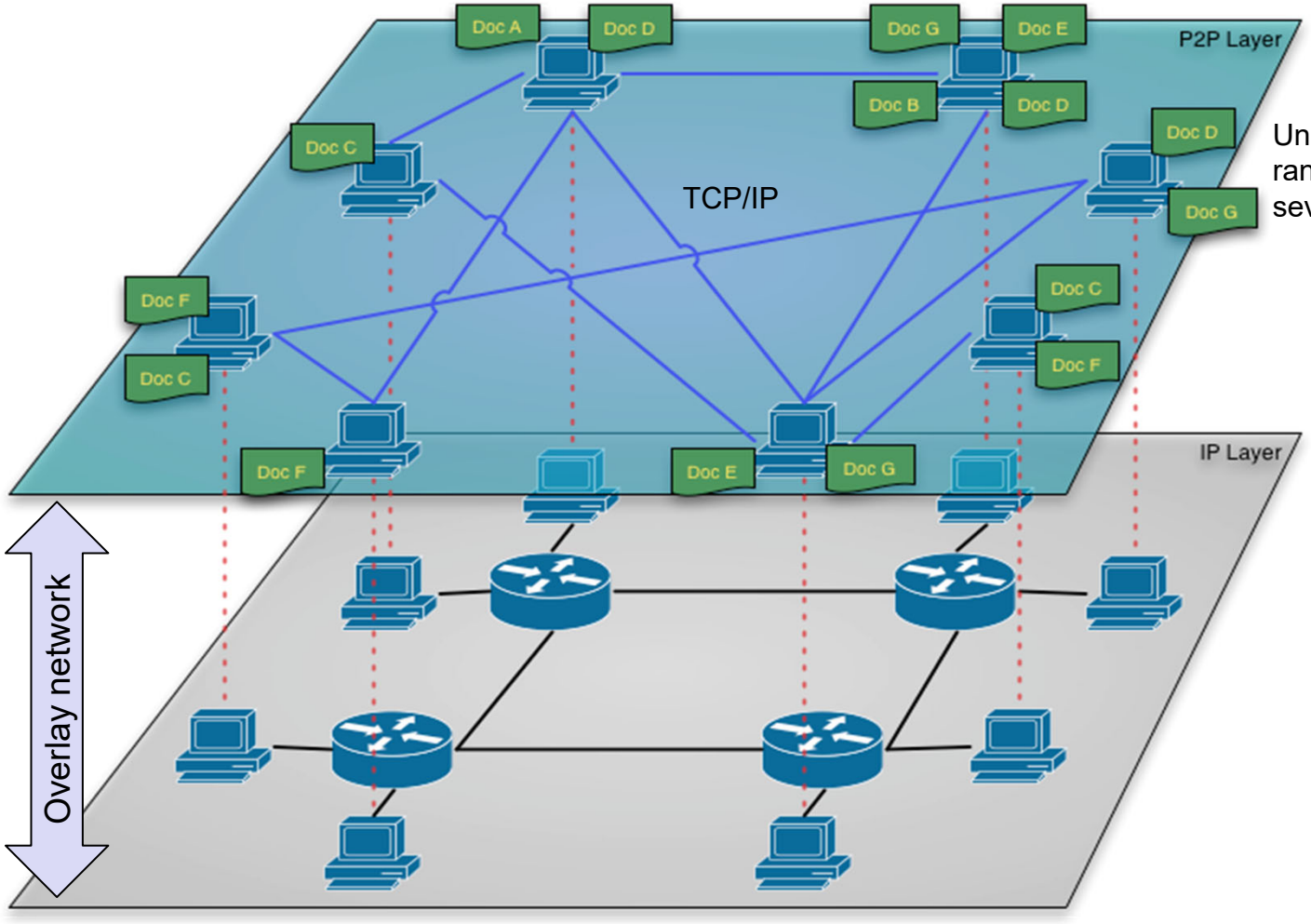
General Characteristics of Overlay Systems

Characteristics, Architecture,
Topology, Routing, and Bootstrapping

Characteristics of 1st & 2nd Generation P2P

- 1st & 2nd generation P2P system overlay architectures
  - TCP/IP-based
  - Decentralized, self-organizing (possibly centralized elements)
  - Content distributed “randomly” on network, several replicas
    - Content and its descriptions are not structured (stays at network nodes)
    - Content transfer
      - Out-of-band, i.e. on separate connections and not via signaling connections
      - Mostly via HTTP
  - Employ shared resources (computing power, storage)
  - Generally two kinds of requests
    - Content requests: to find content in the overlay
    - Keep-alive requests: stay connected in the overlay
  - Initially developed for file sharing, various implementations exist

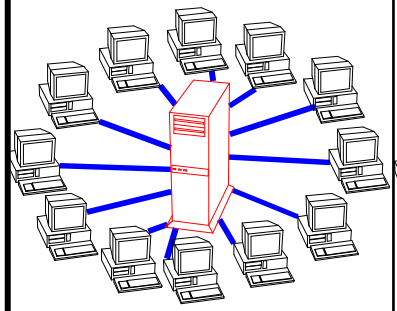
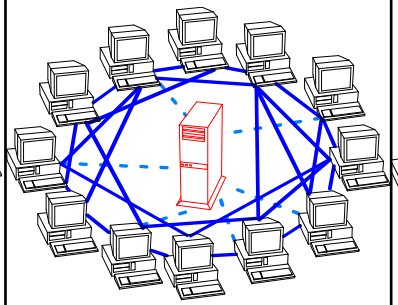
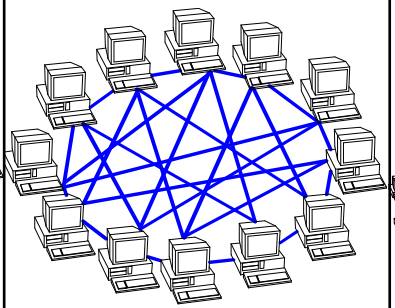
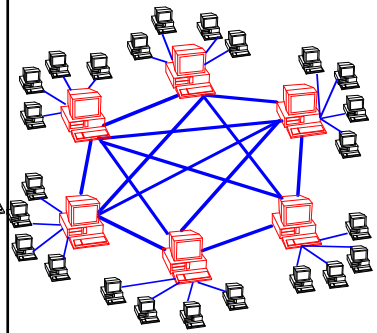
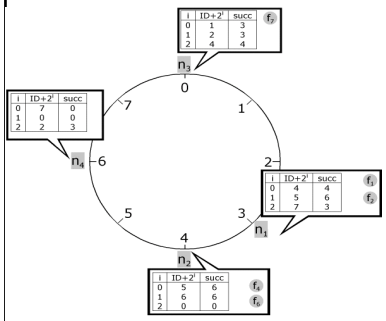
Overview on 1st and 2nd Generation P2P



Unstructured content, randomly distributed, several replicas

1st and 2nd Generation P2P Architectures

<table>
<tr><th>Client-Server</th><th colspan="4">Peer-to-Peer</th></tr>
<tr><td rowspan="3">1. Server is the central entity and only provider of service and content. → Network managed by the Server
2. Server as the higher performance system.
3. Clients as the lower performance system

Example: WWW</td><td colspan="4">1. Resources are shared between the peers
2. Resources can be accessed directly from other peers
3. Peer is provider and requestor (Servent concept)</td></tr>
<tr><td colspan="3">Unstructured P2P</td><td>Structured P2P</td></tr>
<tr><td>Centralized P2P</td><td>Pure P2P</td><td>Hybrid P2P</td><td>DHT-Based</td></tr>
<tr><td></td><td>1. All features of Peer-to-Peer included
2. Central entity is necessary to provide the service
3. Central entity is some kind of index/group database
Example: Napster</td><td>1. All features of Peer-to-Peer included
2. Any terminal entity can be removed without loss of functionality
3. → No central entities
Examples: Gnutella 0.4, Freenet</td><td>1. All features of Peer-to-Peer included
2. Any terminal entity can be removed without loss of functionality
3. → dynamic central entities
Example: Gnutella 0.6, JXTA</td><td>1. All features of Peer-to-Peer included
2. Any terminal entity can be removed without loss of functionality
3. → No central entities
4. Connections in the overlay are “fixed”
Examples: Chord, CAN</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td><td>  </td></tr>
</table>

1st Gen. 2nd Gen.

Basic Topology Characteristics

- Peer: Node actively participating in the overlay (*e.g.*, PC)
  - Content provider, content requestor, router in overlay network
  - Identified by a General Unique ID (Identifier)
    - Hash-value of “unique” or random ID
- Overlay Topology: “Virtual” signaling network established via TCP connections between peers
- Characteristics of overlay topologies
  - Independent of physical network, due to abstraction of TCP/IP
  - May include hierarchies (rendezvous peers, JXTA): hubs
  - May include centralized elements (lookup server, Napster): star
  - May be a completely randomized network (Gnutella 0.4), modeled with random graphs (randomly meshed network)
  - Separate addressing scheme

© 2022 UZH, CSG@IfI

Basic Routing Behavior – Request

- Request messages
  - Includes hop-counter, GUID (Globally Unique Identifier), TTL (Time-to-Live) in header
    - TTL determines along how many hops a message may be forwarded
  - Are flooded in the overlay network
    - Every node forwards every incoming message to all neighbors except the neighbor, it received the message from
    - Exceptions: see below
  - Request messages terminate, if
    - Same message-type with same GUID is received for more than once (loop!!)
    - Hop-counter = TTL

Basic Routing Behavior – Response

- Response messages
  - Includes hop-counter, GUID, TTL (Time-to-Live) in header
  - GUID is the same as of the initializing request message
  - Are routed back on the same way to the requestor, the request message was transmitted to the responding peer
    - Every peer has to store the GUID of each request for a certain amount of time
    - No flooding to save resources

Basic Bootstrapping

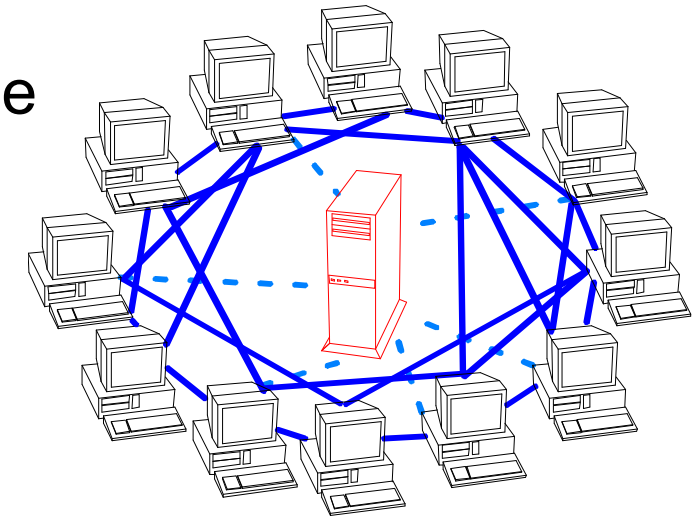
- Mostly not part of the protocol specification
- Necessary to know at least one active participant of the network
- Otherwise, no participation at the overlay possible for a new node
- Address (TCP) of an active node can be retrieved by different means
  - Bootstrap cache
    - Try to establish one after another a connection to a node seen in a previous session
  - Bootstrap server
    - Connect to a “well known host”, which almost always participates
    - Ask a bootstrap server to provide a valid address of at least one active node
    - Implementations
      - FIFO of all node-addresses which recently used this bootstrap (a node which just connected is assumed to be still active)
      - Random pick of addresses which recently connected via this server to the overlay (pros: no loops, cons: may be outdated)
  - Broadcast on the IP layer
    - Use multicast channels
    - Use IP broadcasting (cons: limited to local network)

Centralized P2P Networks

Definition, Basic Characteristics,
Signaling Characteristics, and Discussion

Definition of Centralized P2P

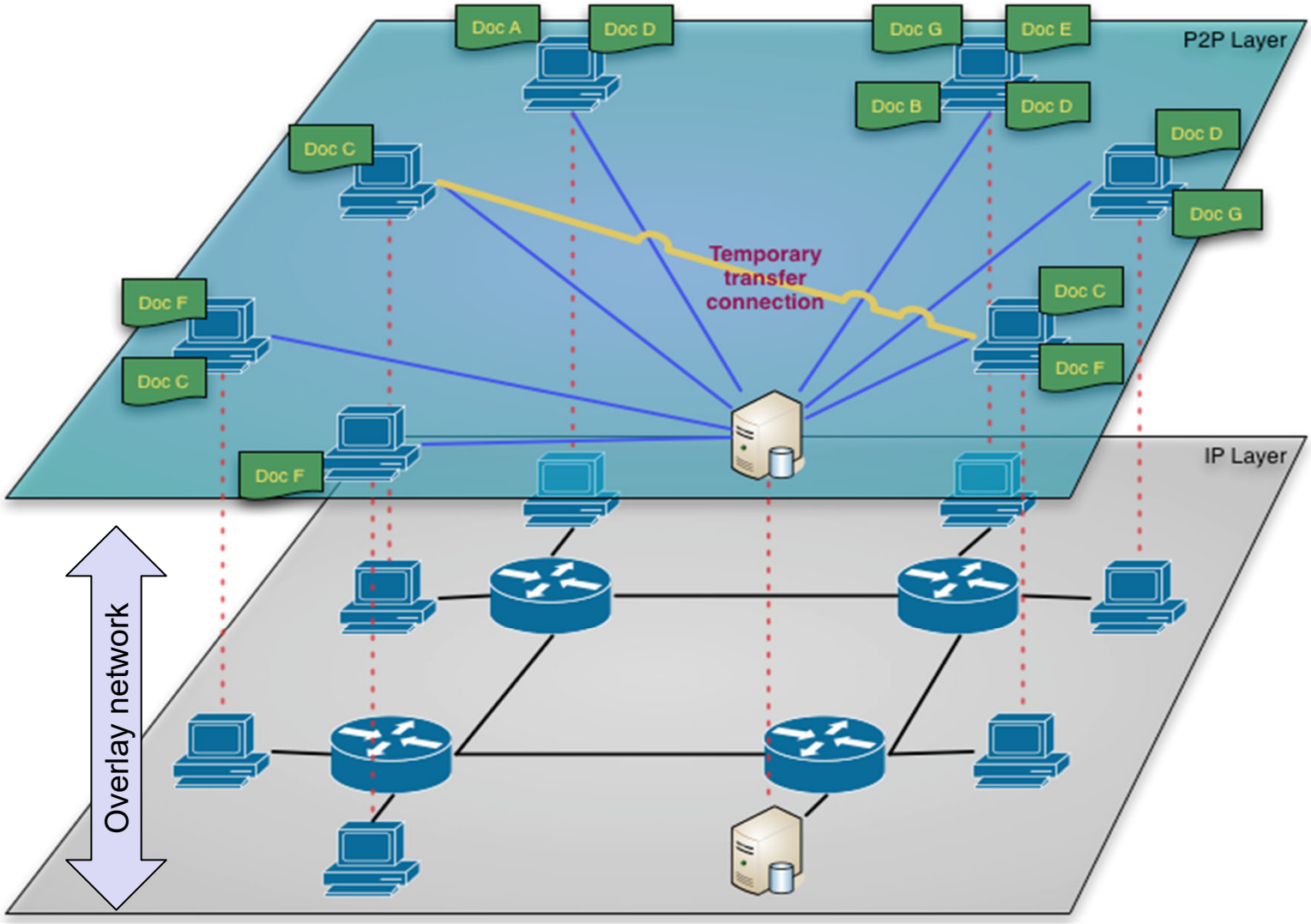
- All peers are connected to a central entity
- Peers establish connections between each other on demand to exchange user data
  - *E.g.*, mp3 compressed data
- Central entity is
  - … necessary to provide the service
  - … some kind of index/group database
  - … lookup/routing table



Basic Characteristics of Centralized P2P

- Bootstrapping: Bootstrap-server = central server
- Central entity can be established as a server farm, but one single entry point = single point of failure (SPOF)
- All signaling connections are directed to central entity
- Peer ↔ central entity:
  P2P protocol, *e.g.*, Napster protocol
  - To find content
  - To log on to the overlay
  - To register
  - To update the routing tables
  - To update shared content information
- Peer ↔ Peer: HTTP
  - To exchange content/data

Topology of Centralized P2P

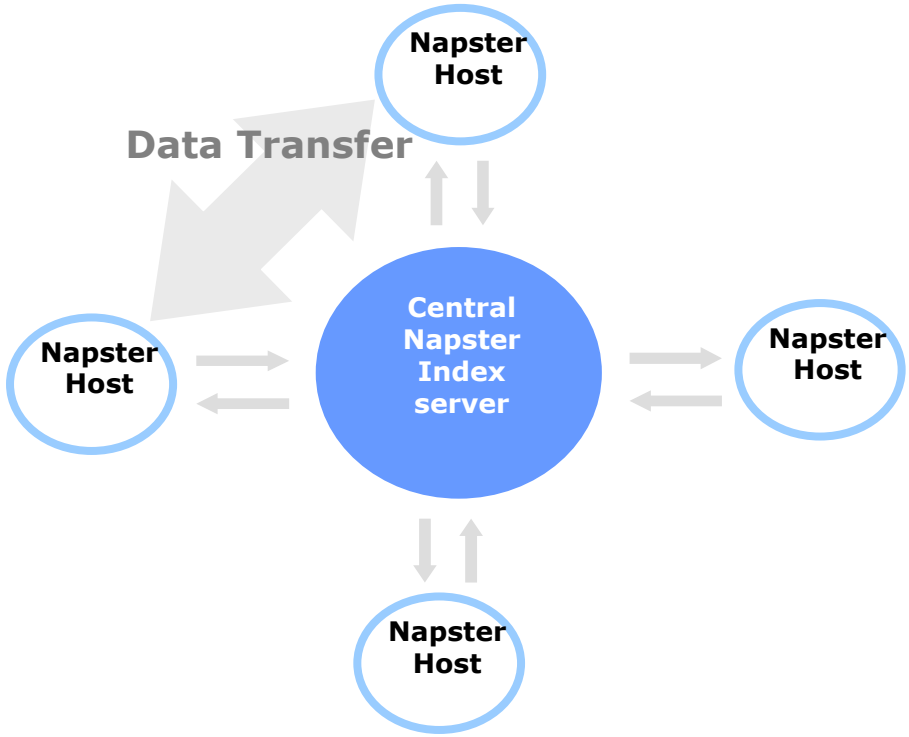


Centralized P2P — Example: Napster

- Program for sharing files over the Internet
- A “disruptive” application/technology?
- Brief History
  - **May 1999**: Shawn Fanning (freshman, Northeastern University) founds Napster Online music service
  - **December 1999**: First Lawsuit
  - **March 2000**: University of Wisconsin reports that 25% of its IP traffic is Napster traffic
  - **December 2000**: estimated 60 million users
  - **February 2001**: US Circuit Court of appeals: Napster knew users violating copyright laws
    → Shut down of the service

Napster — Basic Operations

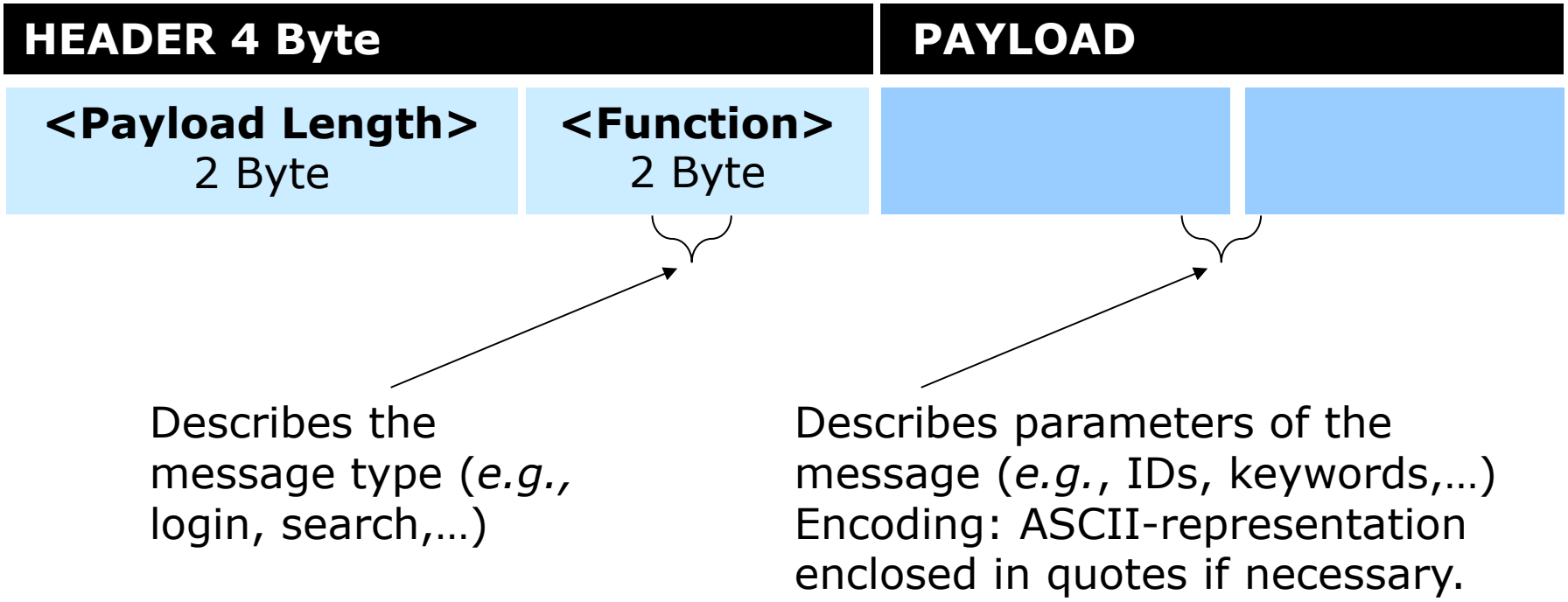
- Application-level, client-server protocol over point-to-point TCP
- Participants
  - Napster Hosts/peers
  - Client Service
    - Login
    - Data-requests
    - Download-requests
  - P2P Service
    - Data-transfer
  - Napster Index Server
    - Pure Server
- Five steps
  - Connect to Napster Server
  - Upload your list of files (push) to server
  - Query Index Server with a list of keywords to search the full list
  - Select “best” of correct answers
  - Connect to providing host/peer



© 2022 UZH, CSG@IfI

Napster Message Structure

General Header Structure:



Napster Initialization

Client/Server Service

LOGIN (Function: 0x02)

<Nick>	<Password>	<Port>	<Client-Info>	<Link-type>

LOGIN ACK (Function: 0x03)

NOTIFICATION OF SHARED FILE (Function: 0x64)

”<Filename>“	<MD5>	<Size>	<Bitrate>	<Freq>	<Time>

Napster Host IP: 001 Nick: LKN

NOTIFICATION (Function: 0x64)

„band - song.mp3“ 3f3a3... 5674544

128 44100 342

Central Napster Index server

Napster File Search Procedure

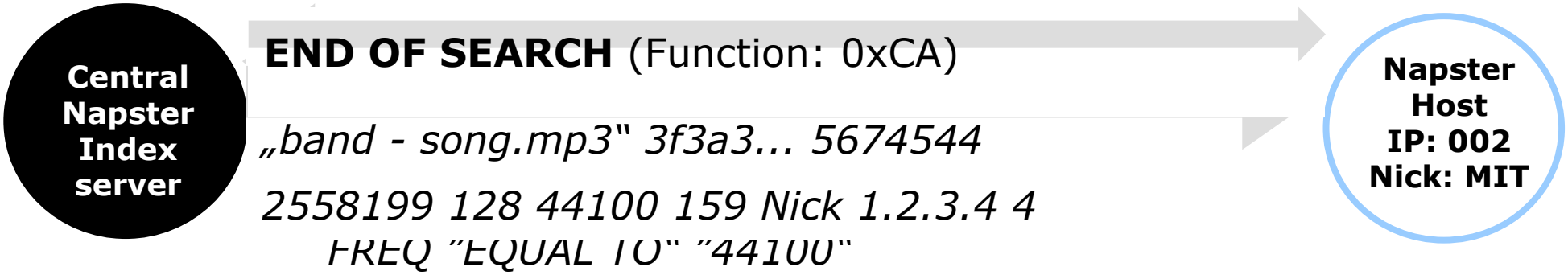
SEARCH (Function: 0xC8)

[FILENAME CONTAINS ”Search Criteria“]	[MAX_RESULT <Max>]
[LINESPEED <Compare> <Link-Type>]	
[BITRATE <Compare> “<Bitrate>”]	[FREQ <Compare> “<Freq>”]

SEARCH RESPONSE (Function: 0xC9)

”<Filename>“	<MD5>	<Size>	<Bitrate>	<Freq>
<Time>	<Nick>	<IP>	<Link-Type>	

END OF SEARCH RESPONSE (Function: 0xCA)



© 2022 UZH, CSG@IfI

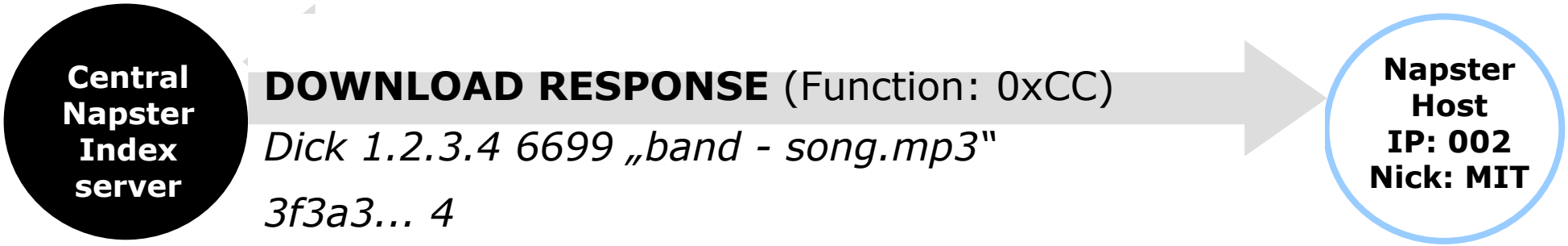
Napster File Request Procedure

DOWNLOAD REQUEST (Function: 0xCB)

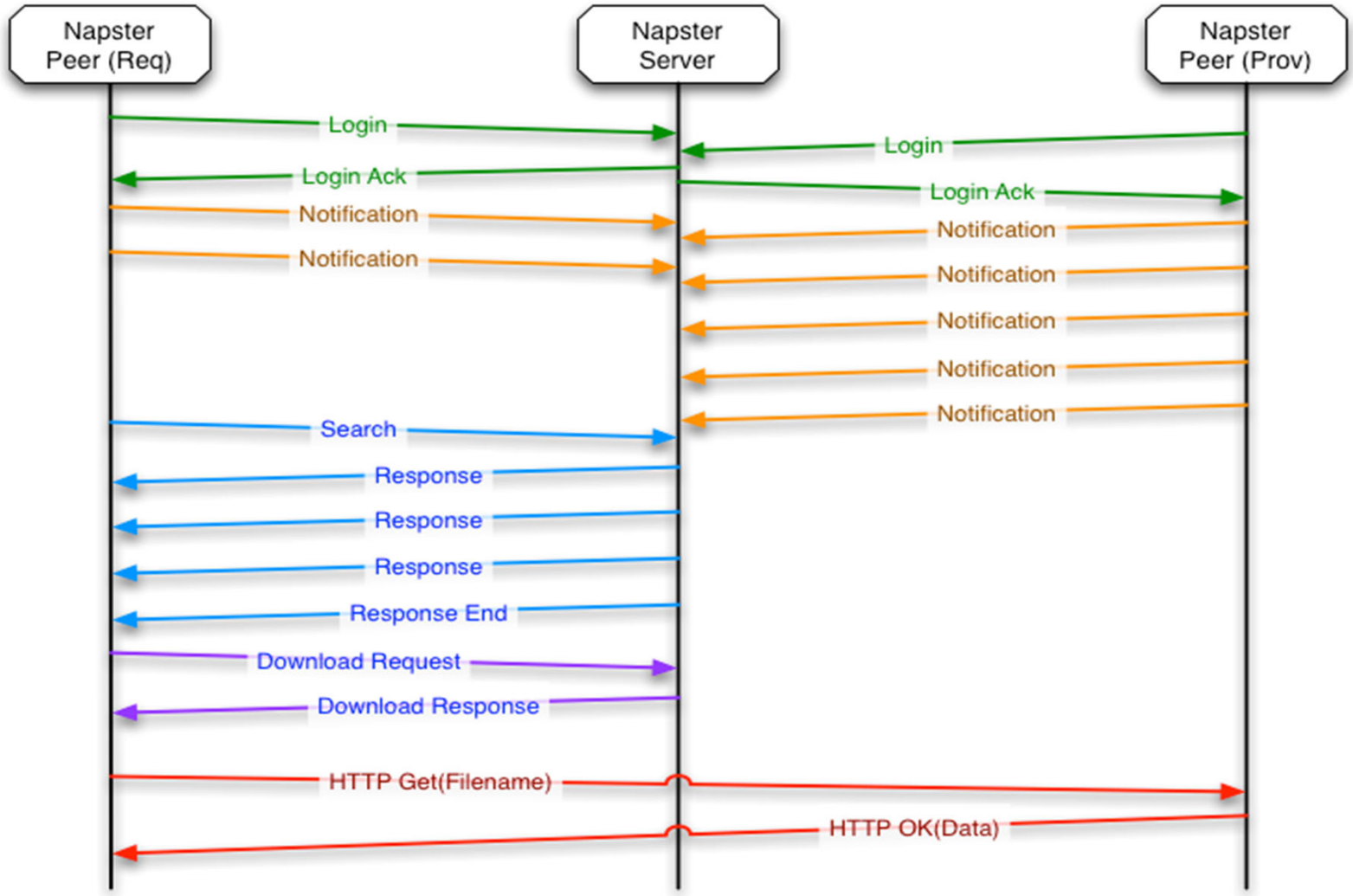
<Nick>	”<Filename>“

DOWNLOAD RESPONSE (Function: 0xCC)

<Nick>	<IP>	<Port>	<Filename>	<MD5>	<Link-type>

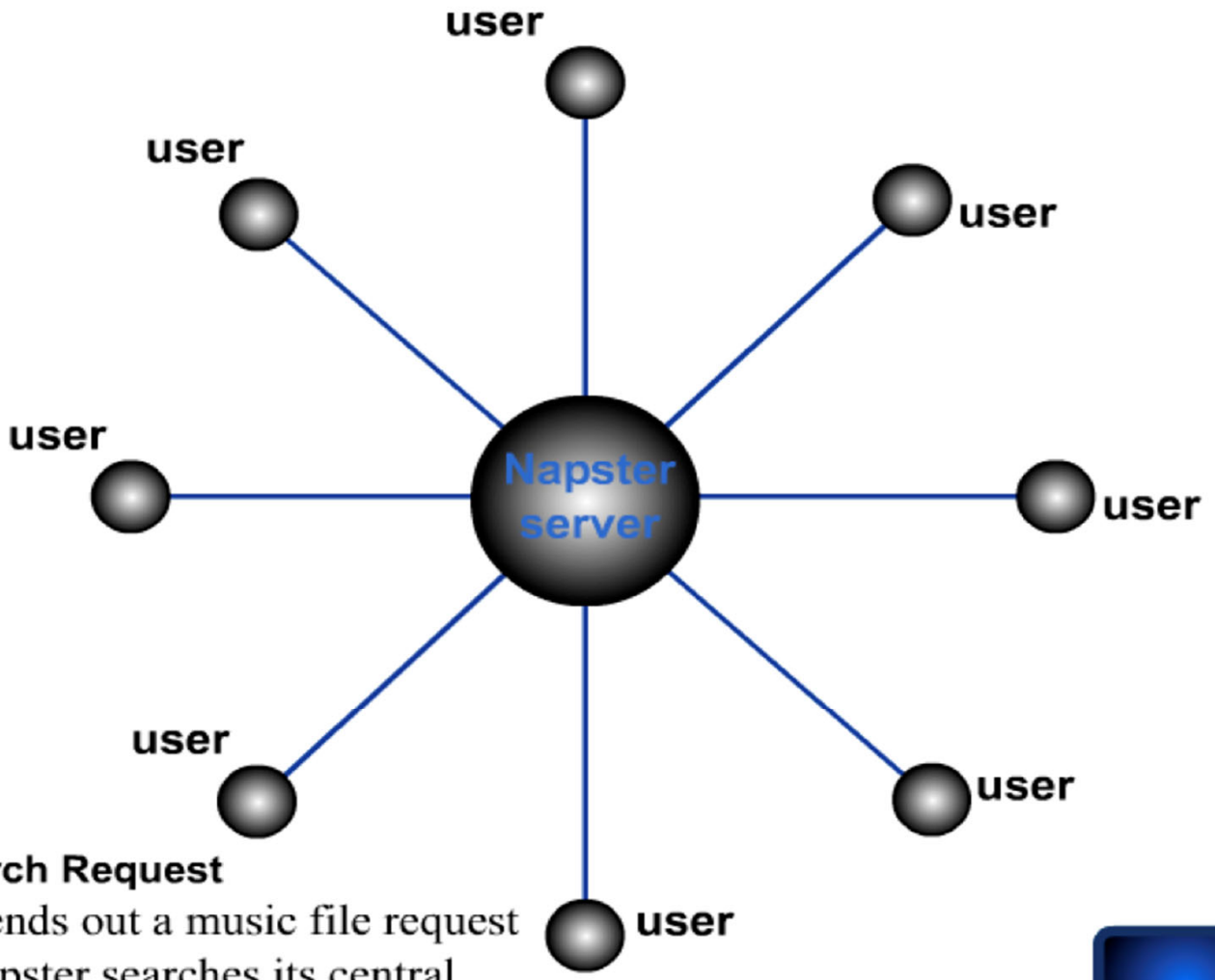


Napster Signaling (Summary)



Sample message sequence for one Napster server with one requesting and one providing peer

Napster Discussion (1)



1) Search Request

User sends out a music file request and Napster searches its central data base

Napster Discussion (2)

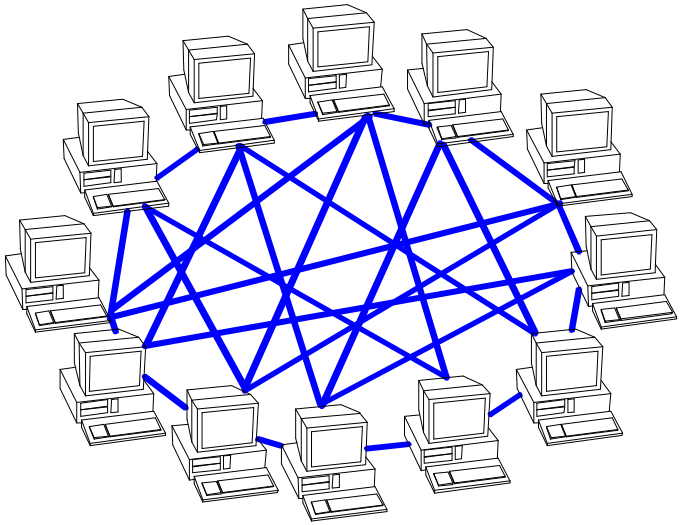
- Drawbacks
  - Single Point of Failure → Easily attackable
  - Bottleneck
  - Potential of congestion
  - Central server in control of all peers
- Advantages
  - Fast and complete lookup (one hop lookup)
  - Central managing/trust authority
  - No keep alive necessary, beyond content updates
- Application areas (examples)
  - VoIP (SIP, H.323)
  - Auctioning (Ebay)

Decentralized P2P Networks

Definition, Basic Characteristics,
Signaling Characteristics, and Discussion

Definition of Decentralized P2P (Pure P2P)

- Any terminal entity can be removed without loss of functionality
- No central entities at all employed in the overlay
- Peers establish connections between each other randomly
  - To route request and response messages
  - To insert request messages into the overlay



Basic Characteristics of Decentralized P2P (1)

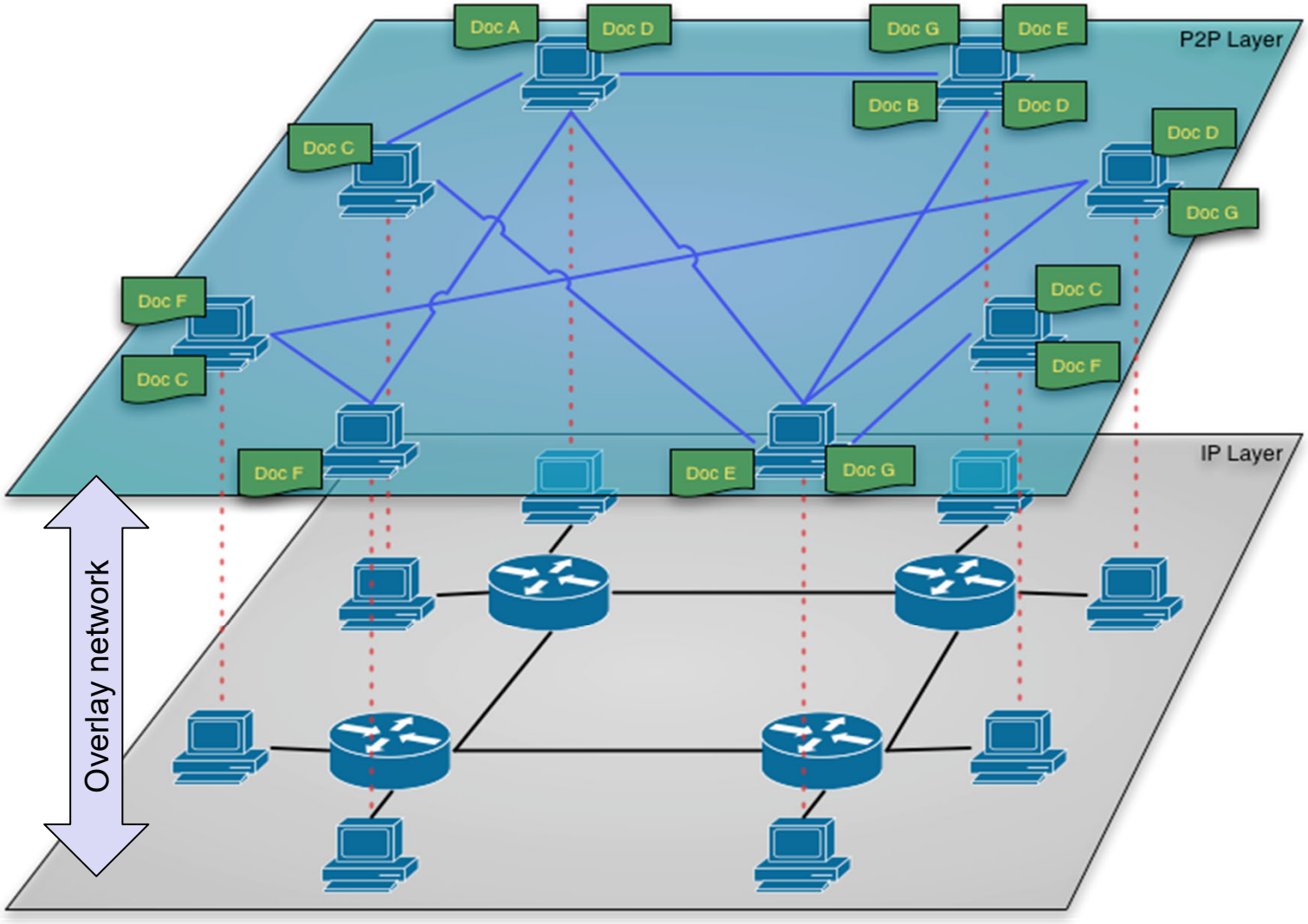
- Peer (Servent), identified by GUID
- Bootstrapping
  - Via bootstrap-server (host list from a web server)
  - Via peer-cache (from previous sessions)
  - Via well-known host
  - No registration
- Routing (Peer ↔ Peer ↔ Peer ↔ Peer ↔ Peer …)
  - Completely decentralized
  - Reactive protocol: routes to content providers are only established on demand, no content announcements
  - Requests: flooding (limited by TTL and GUID)
  - Responses: routed (Backward routing with help of GUID)

Basic Characteristics of Decentralized P2P (2)

- Signaling connections
  (stable, as long as neighbors do not change)
  - Based on TCP
  - Keep-alive
  - Content search

- Data/Content transfer connections (Peer ↔ Peer)
  - Temporary
  - Based on HTTP
  - Out of band transmission

Topology of Decentralized P2P

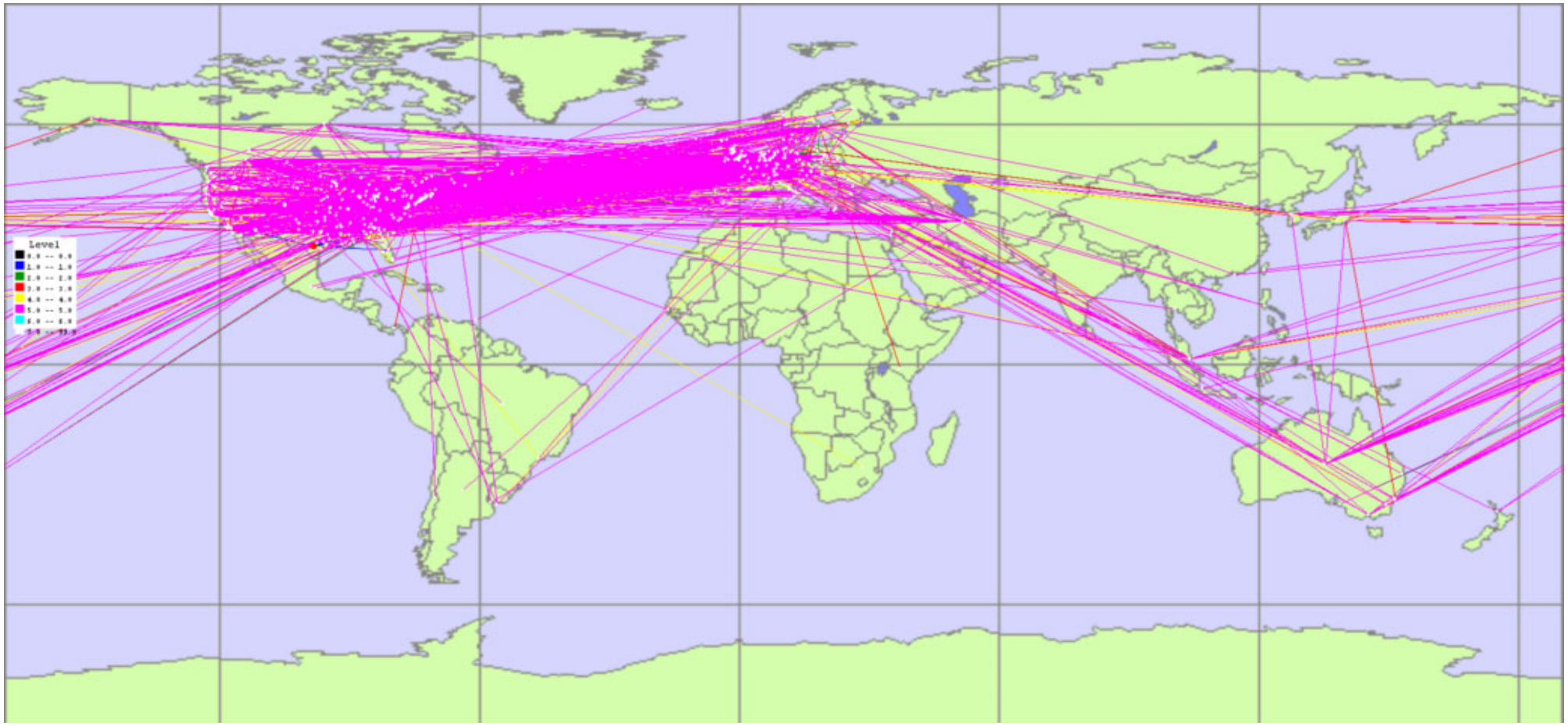


Decentralized P2P — Example: Gnutella 0.4

- Program for sharing files over the Internet
- Focus: decentralized method of searching for files
- A “disruptive” application/technology?

- Brief History
  - **March 2000**: open source release by by Justin Frankel and Tom Pepper of Nullsoft, a division of AOL, and almost immediately withdrawn
  - **Spring 2001**: further developments to improve scalability → Gnutella 0.6 (Hybrid P2P)
  - Since then:
    - Available in a lot of implementations (Limewire, bearshare,…)
    - Developed further on (privacy, scalability, performance,…)

The Gnutella Network

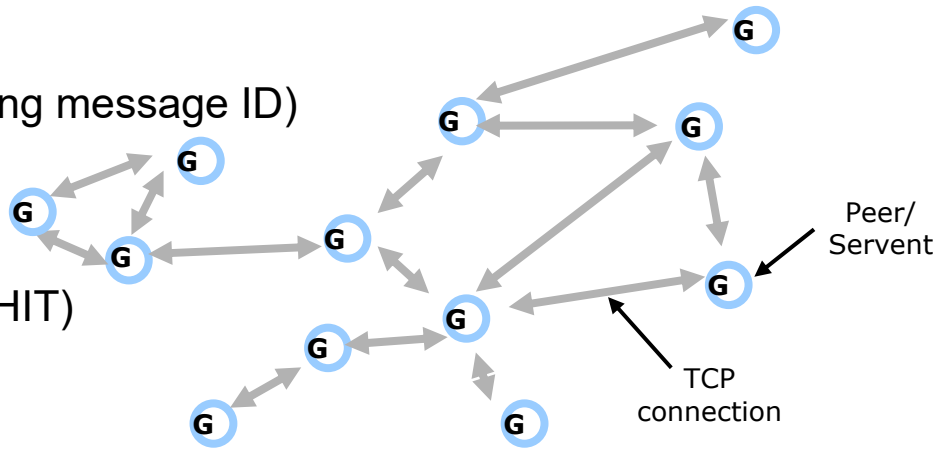


Measurements taken at LKN in May 2002

© 2022 UZH, CSG@IfI

Gnutella — Basic Operations

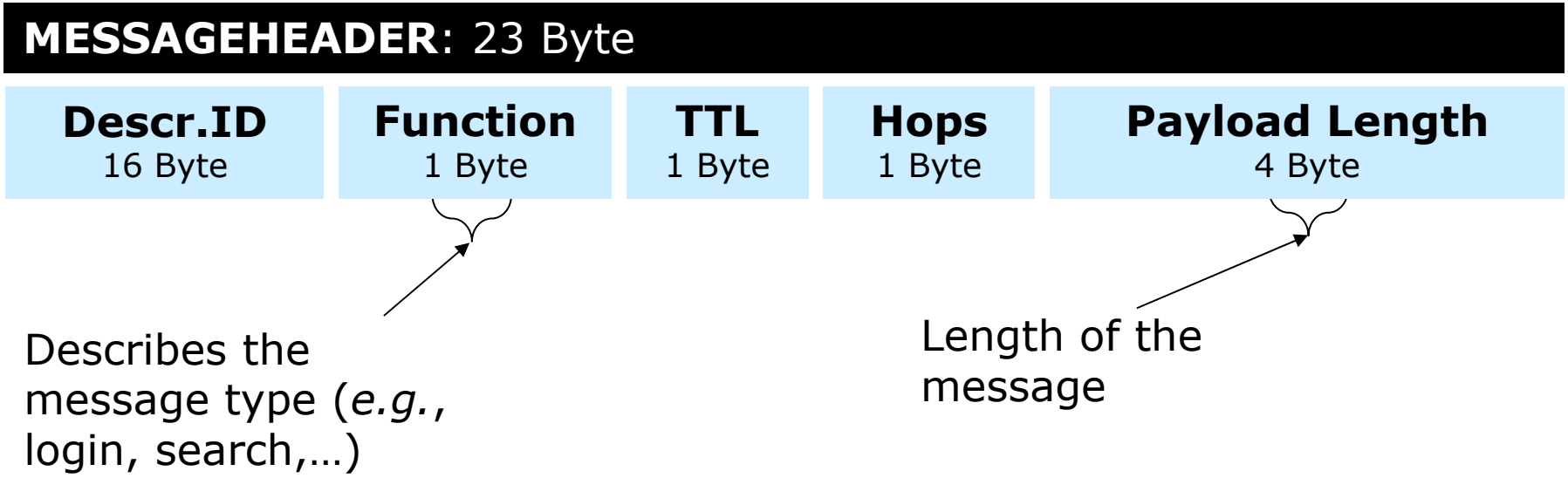
- Application-level, peer-to-peer protocol over point-to-point TCP participants:
  - Gnutella peers/servents
    - Establish overlay network by connecting to peers
    - After bootstrap, use Ping/Pong to find peers
  - Router Service
    - Flood requests (regard TTL!)
      - Keep alive (PING)
      - Content (QUERY)
    - Route responses for other peers (using message ID)
      - Keep alive (PONG)
      - Content (QUERYHIT)
  - Lookup Service
    - Answer QUERY messages (QUERYHIT)
  - "Server"-Service
    - Serve Data-requests (HTTP)



- Five steps
  - Connect to at least one active peer (address received from bootstrap)
  - Explore your neighborhood (PING/PONG)
  - Submit Query with a list of keywords to your direct peers (they forward it)
  - Select "best" of correct answers (which we receive after a while)
  - Connect to providing host/peer

Gnutella Message Structure

General Header Structure:



- **Descr.ID**: Unique 128 bit ID of the message (e.g. random!)
- **TTL** (Time-To-Live): Number of servents, a message may pass before it is killed
- **Hops**: Number of servents a message already passed

Gnutella Messages

PING (Function: 0x00) *No Payload*

PONG (Function: 0x01)

Port 2 Byte	**IP Address** 4 Bytes	**No. of shared Files** 4 Byte	**No. of Kbytes shared** 4 Byte

QUERY (Function: 0x80)

Minimum Speed 2 Byte	**Search Criteria** n Byte

QUERY HIT (Function: 0x81)

No. of Hits 1 Byte	**Port** 2 Byte	**IP Address** 4 Byte	**Speed** 1 Byte	**Result Set** n Byte	**GnodeID** 16 Byte

File Index 4 Byte	**File Name** m Byte

Gnutella Routing

- **Basic Routing Principle: „Enhanced" Flooding**

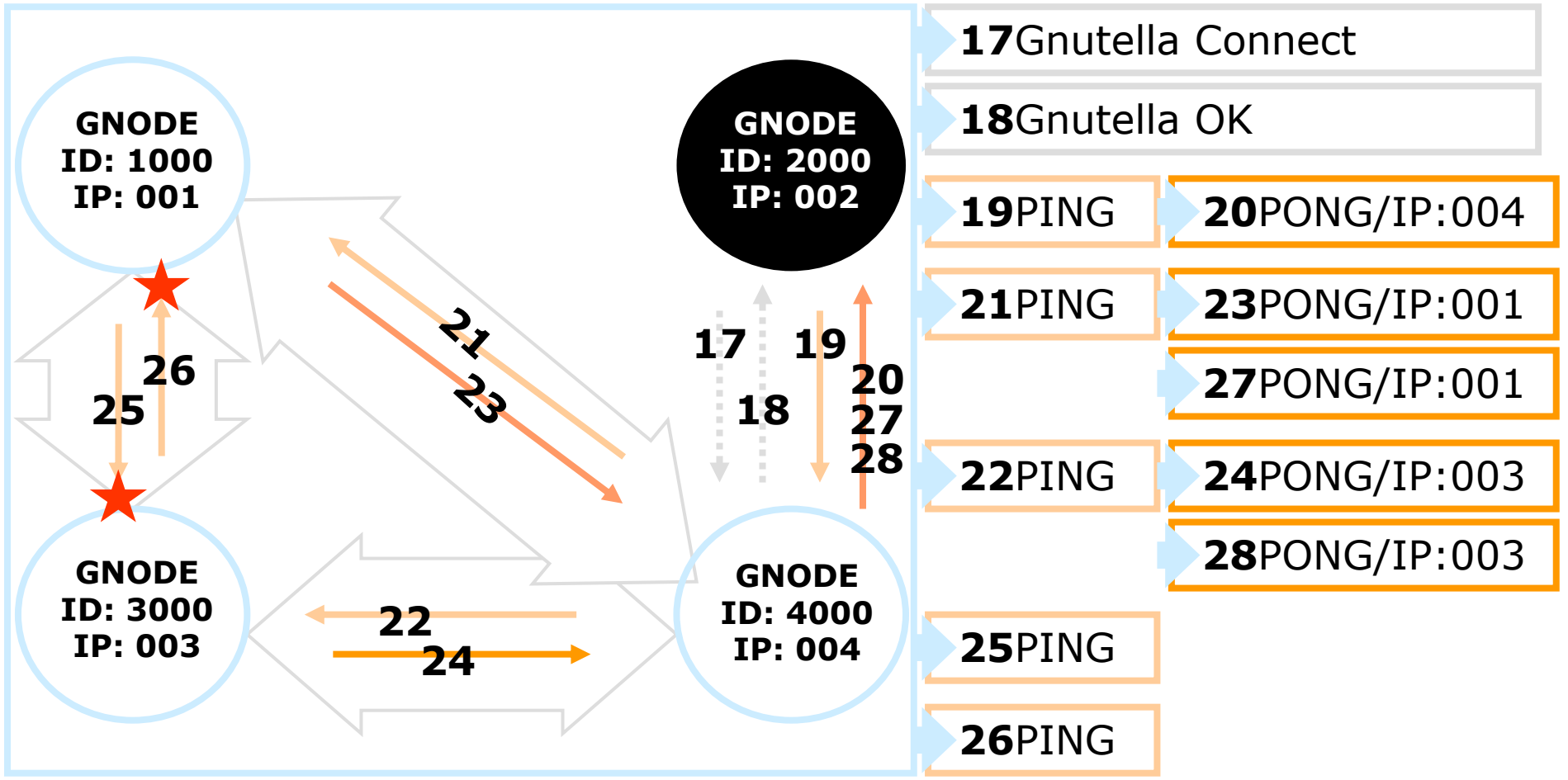
- Save Origin of received **PINGs** and **QUERIEs**
- Decrease **TTL** by 1
- If **TTL** equals 0, kill the message

- **Flooding**: Received **PINGS** and **QUERIES** must be forwarded to all connected Gnodes
- **PINGS** or **QUERYS** with the same FUNCTION ID and Descr.ID as previous messages are destroyed (avoid loops)

- **PONG** and **QUERY HIT are forwarded to the** origin of the according PING or QUERY

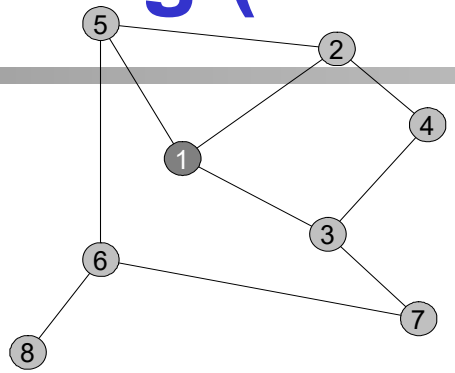
Gnutella Connection Setup

Gnode 2000 establishes a connection to 4000

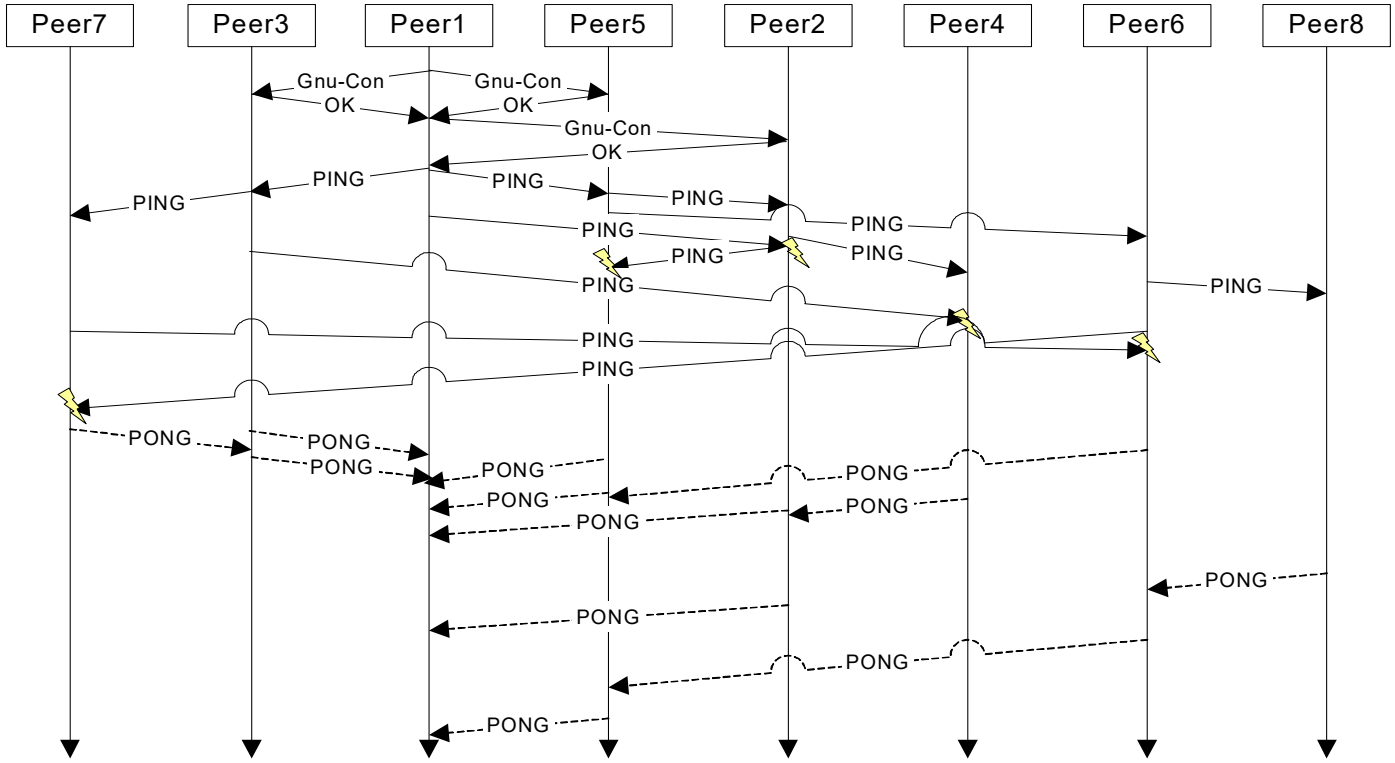


Summary of Signaling (Gnutella 0.4)

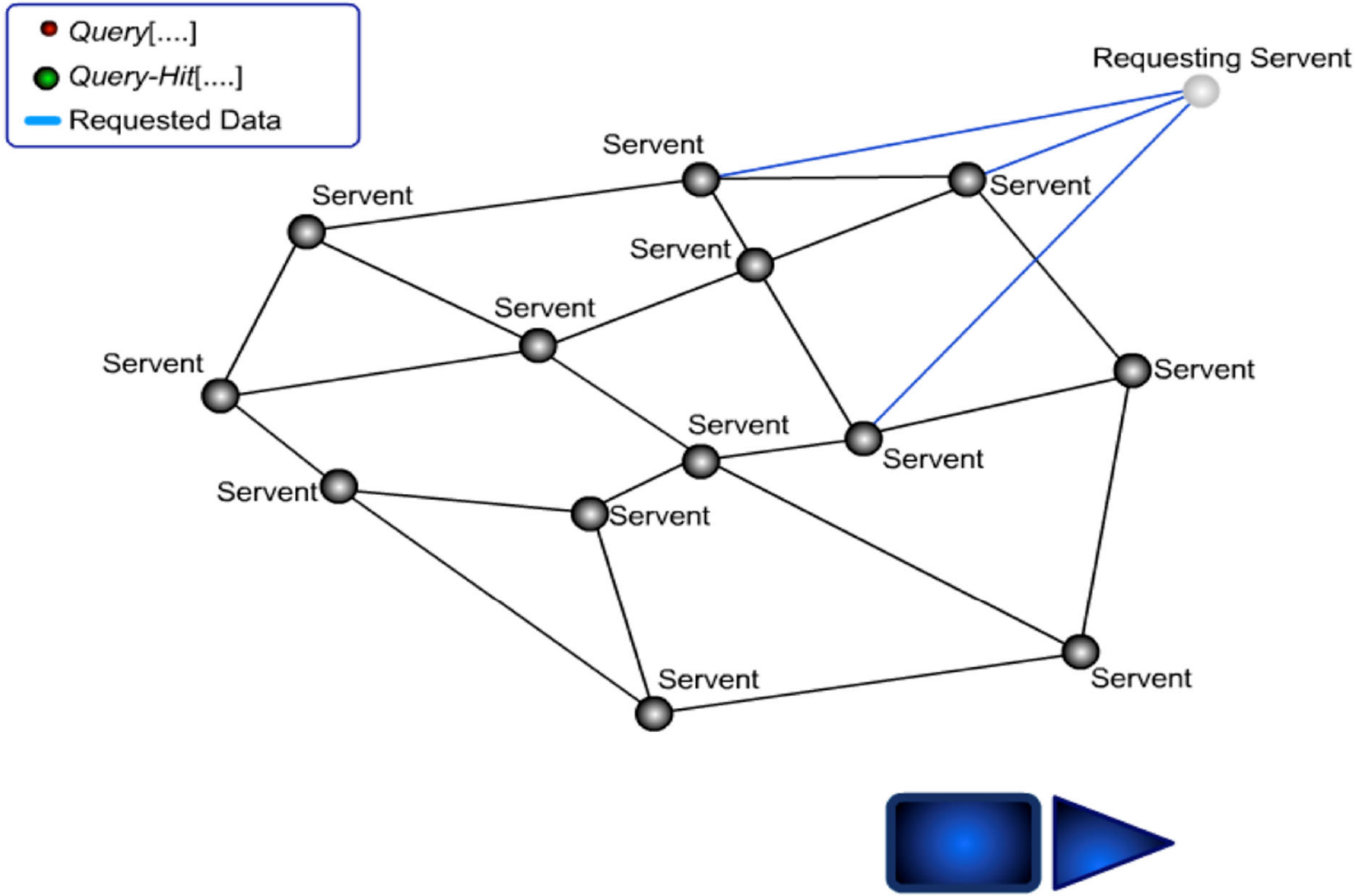
Sample Gnutella 0.4 network:



Sample message sequence chart according to the sample network:



Discussion (1)



Discussion (2)

- Drawbacks
  - High signaling traffic, because of flooding
  - Modem nodes may become bottlenecks
  - Overlay topology not optimal, since
    - No complete view available, no coordinator
  - If not adapted to physical structure delay and total network load increases: Zigzag routes, loops
- Advantages
  - No single point of failure, can provide anonymity
  - Can be adapted to physical network and special interest groups
- Application areas
  - File-sharing, context-based routing

History and Recent Past

From the ARPANET to Peer-to-Peer
and Overlay Network Developments

From the ARPANET to Peer-to-Peer

- Late 1960s: Establishment of the ARPANET
  - Goal: share computing resources, documents between US research facilities
  - Applications: FTP and TelNet
  - Client-Server mode, with no decentralized search and storage
- 1979: Development of the UseNet protocol
  - Newsgroup application to organize content
  - Self organizing approach to add and remove newsgroup servers
  - Application itself is still a client server application
- ~1990: Rush of the general public to join the Internet
  - Applications following the client server approach: WWW, email, streaming
  - Straightforward model to administrate and control the content distribution
  - Security concerns resulted in a partitioned Internet by firewalls
- ~ 2000/05: Application of P2P principles in dist. systems mgmt
  - Resource management (mgmt) approaches for grid computing environments
  - Bandwidth, channel mgmt for wireless access networks

From Peer-to-Peer to Overlay Networks

- P2P as a term is “disappearing” (due to legal concerns)
- Overlay mechanisms as a form to create/utilize overlay networks do exist and are exploited
  - Optimize the sequence of nodes a data packet traverses to its destination, but do not change the IP routing procedures
  - Akamai (commercial overlay)
  - All Content Distribution Networks (CDN) are overlay networks
  - Tradepal (student interconnects): https://www.tradepal.com
  - Blockchains/Bitcoin: Distributed Ledger for blocks and tx
- Overlay approaches under change pressure
  - Skype (moving away, mobile devices battery efficiency, Dec 2017), https://www.lifewire.com/skype-changes-from-p2p-3426522

Distributed Hash Tables (DHT)





Slides based on slides by K. Wehrle, S. Götz, S. Rieche (University of Tübingen)

ToC

1. Distributed Management and Retrieval of Data
   - Comparison of strategies for data retrieval
   - Central server
   - Flooding search
   - Distributed indexing
   - Comparison of lookup concepts
2. Fundamentals of Distributed Hash Tables
   - Distributed management of data
   - Addressing in Distributed Hash Tables
   - Routing
   - Data Storage
3. DHT Mechanisms
   - Node Arrival
   - Node Failure / Departure
4. DHT Summary
   - DHT Interfaces
   - Properties of DHTs
   - DHT Implementations
5. Kademlia
   - Routing Concept
   - Protocol
   - Routing Table Construction
   - Peer Selection Policy
   - Enhancements

© 2022 UZH, CSG@IfI

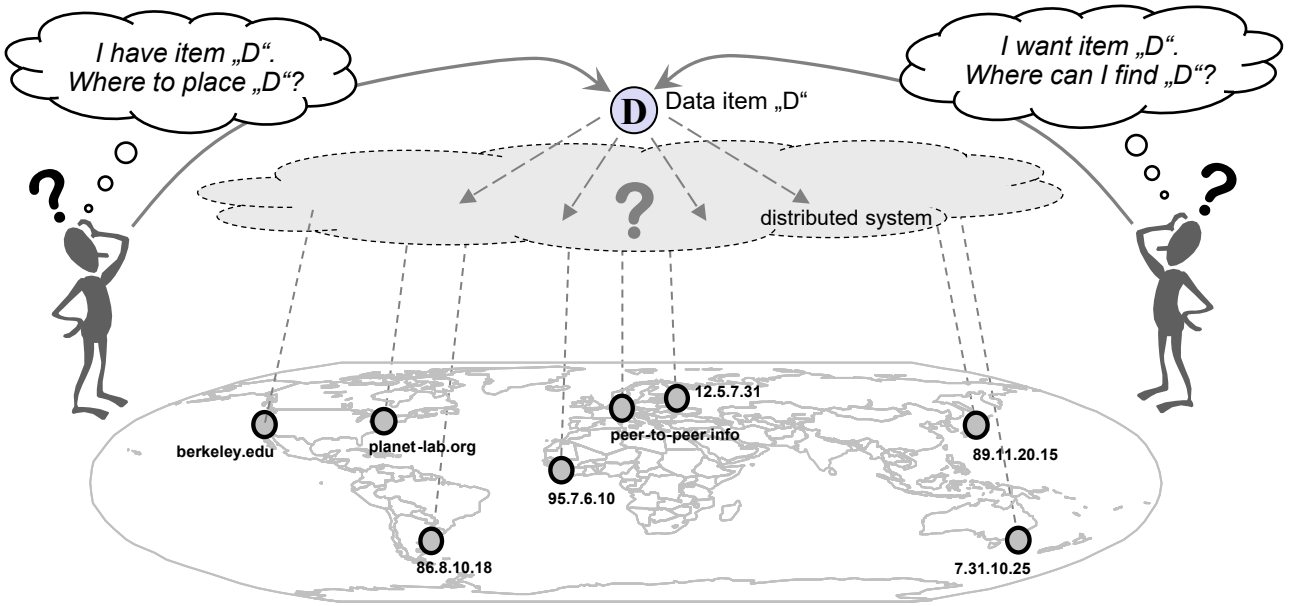
Distributed Management and Retrieval of Data

Comparison of Strategies:

Central Server, Flooding, Distributed Indexing

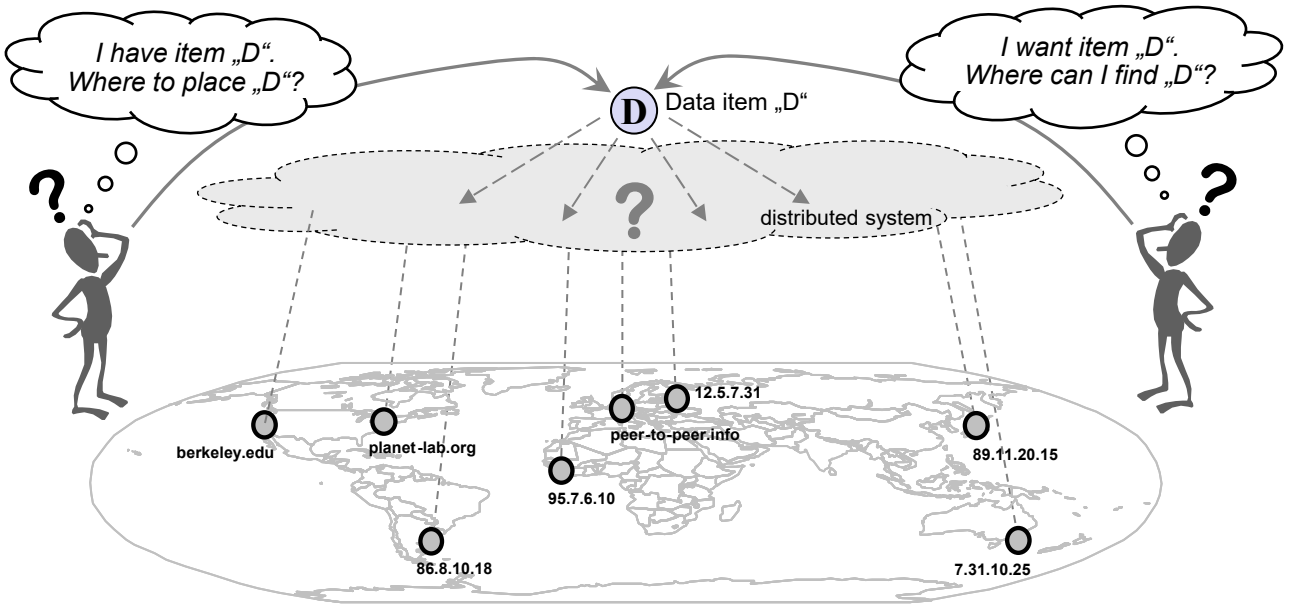
© 2022 UZH, CSG@IfI

Distributed Management and Retrieval of Data



- Essential challenge in (most) Peer-to-Peer systems?
  - Location of a data item among systems distributed
    - Where shall the item be stored by the provider?
    - How does a requester find the actual location of an item?
  - Scalability: keep the complexity for communication and storage scalable
  - Robustness and resilience in case of faults and frequent changes

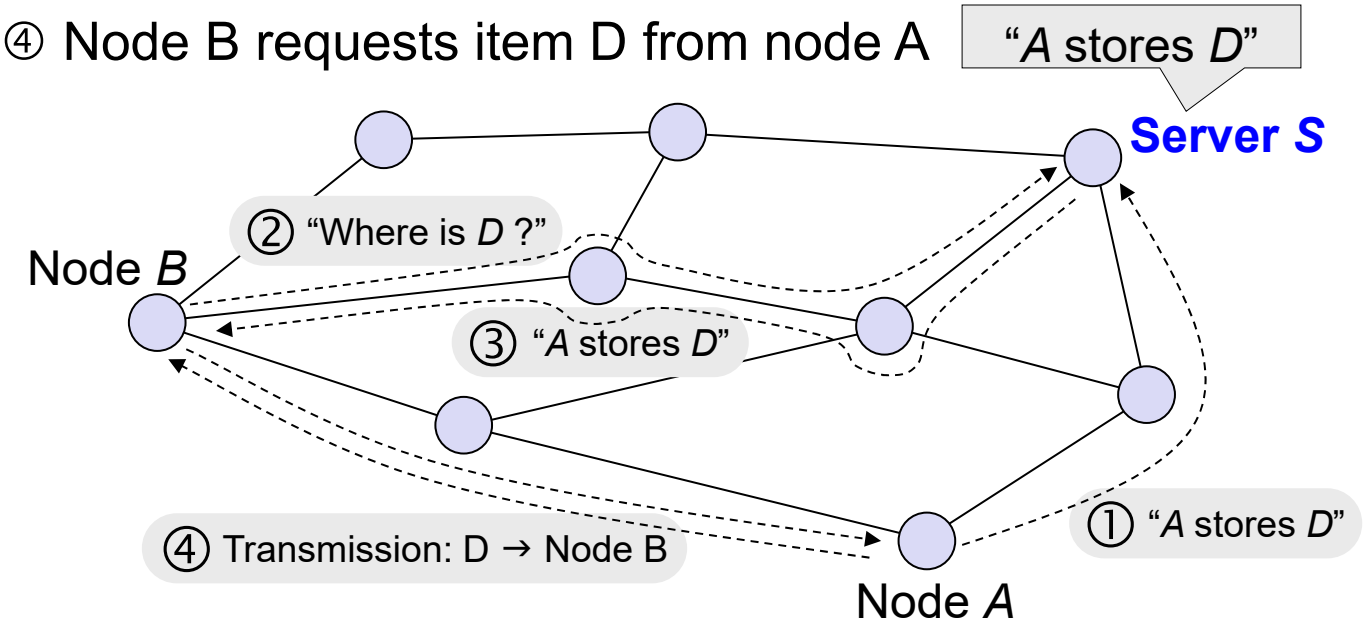
Comparison of Strategies for Data Retrieval



- Strategies to store and retrieve data items in distributed systems
  - Central server
  - Flooding search
  - Distributed indexing

Approach I: Central Server (1)

- Simple strategy: Central Server
  - Server stores information about locations
    - ① Node A (provider) tells server that it stores item D
    - ② Node B (requester) asks server S for the location of D
    - ③ Server S tells B that node A stores item D
    - ④ Node B requests item D from node A



Approach I: Central Server (2)

- Advantages
  - Search complexity of O(1) – *"just ask the server"*
  - Complex and fuzzy queries are possible (i.e., not only exact match queries)
  - Simple and fast
- Problems
  - No Scalability
    - O(N) node state in server
    - O(N) network and system load of server
  - Single point of failure or attack (also for lawsuits, or not ☺)
  - Central server not suitable for systems with massive numbers of users
- But overall, best principle for small, simple applications!

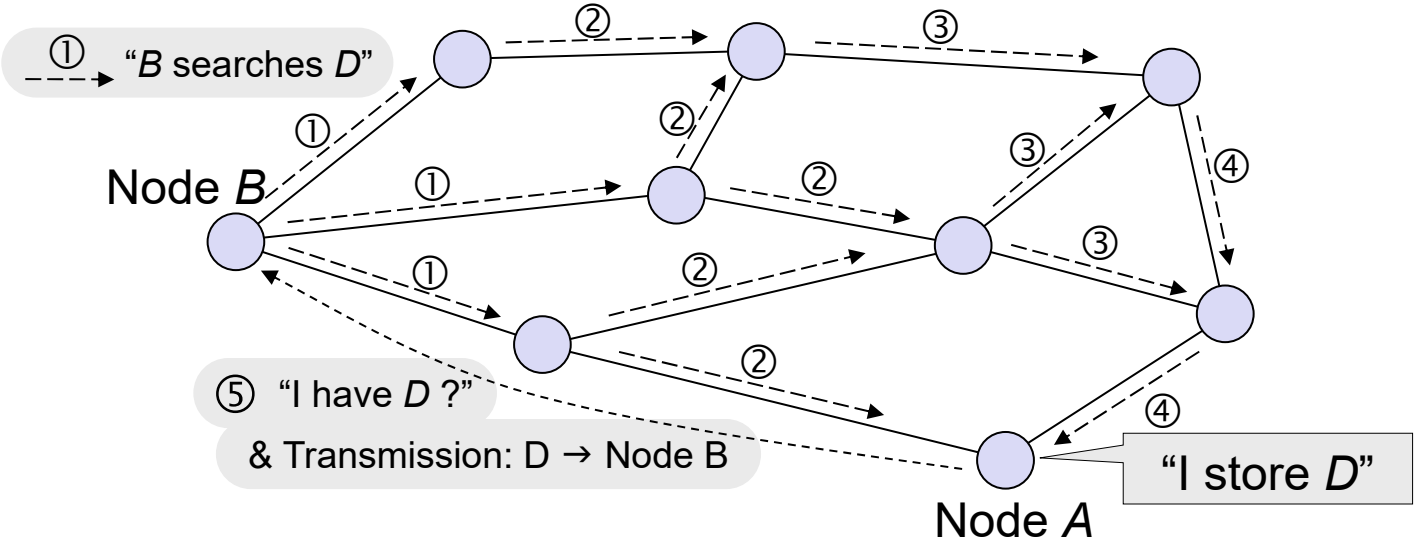
Approach II: Flooding Search (1)

- Fully-distributed approach
  - Central systems are vulnerable and do not scale
  - Unstructured Peer-to-Peer systems follow opposite approach
  - No information on location of a content

- Retrieval of data
  - No routing information for content
  - Necessity to ask as much systems as possible / necessary
  - Approaches
    - Flooding: high traffic load on network, does not scale
    - Highest degree search: quick search through large areas – large number of messages needed for unique identification

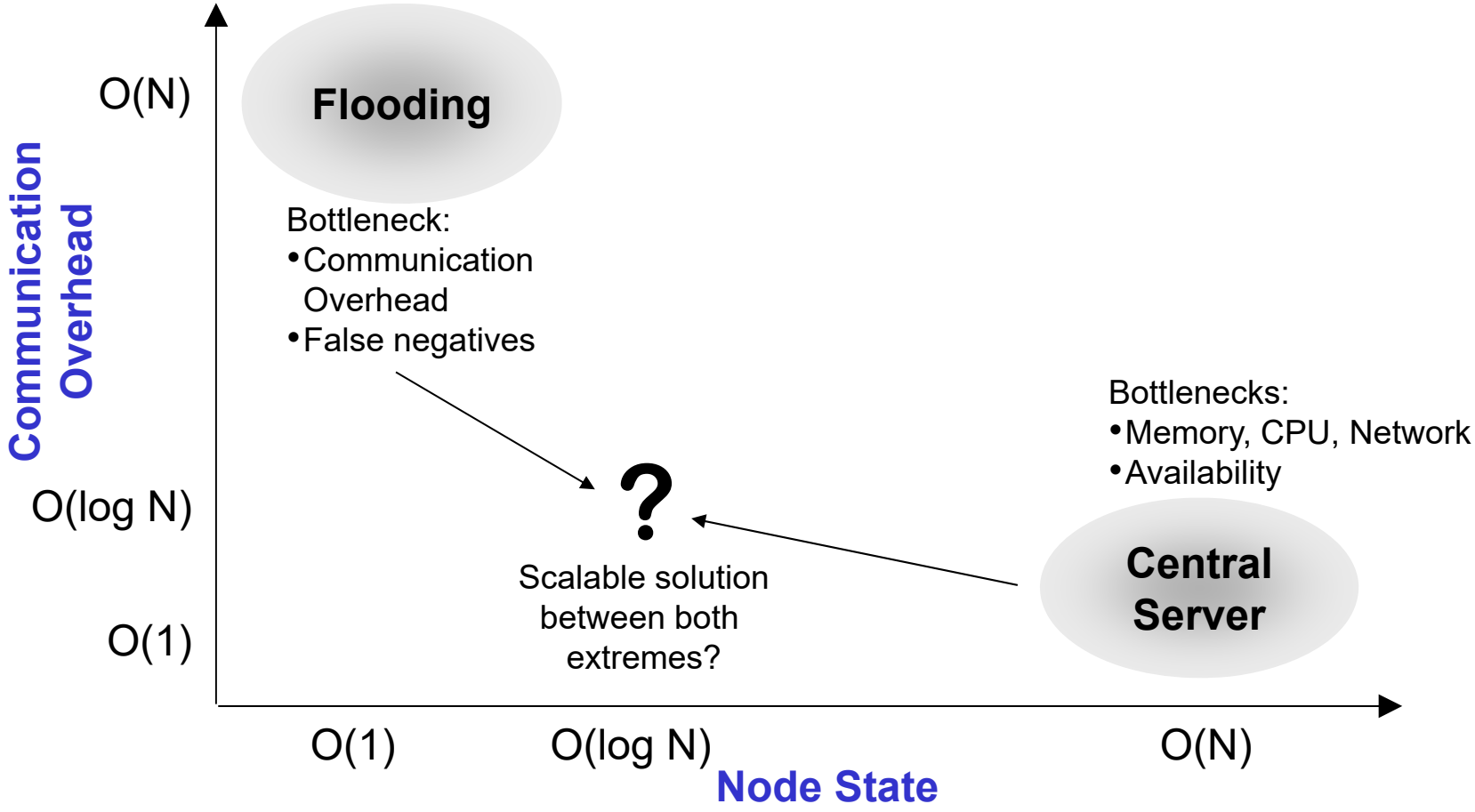
Approach II: Flooding Search (2)

- Fully Decentralized Approach: Flooding Search
  - No information about location of data in intermediate systems
  - Necessity for broad search
    - ① Node B (requester) asks neighboring nodes for item D
    - ②-④ Nodes forward request to further nodes (breadth-first search/flood)
    - ⑤ Node A (provider of item D) sends D to requesting node B



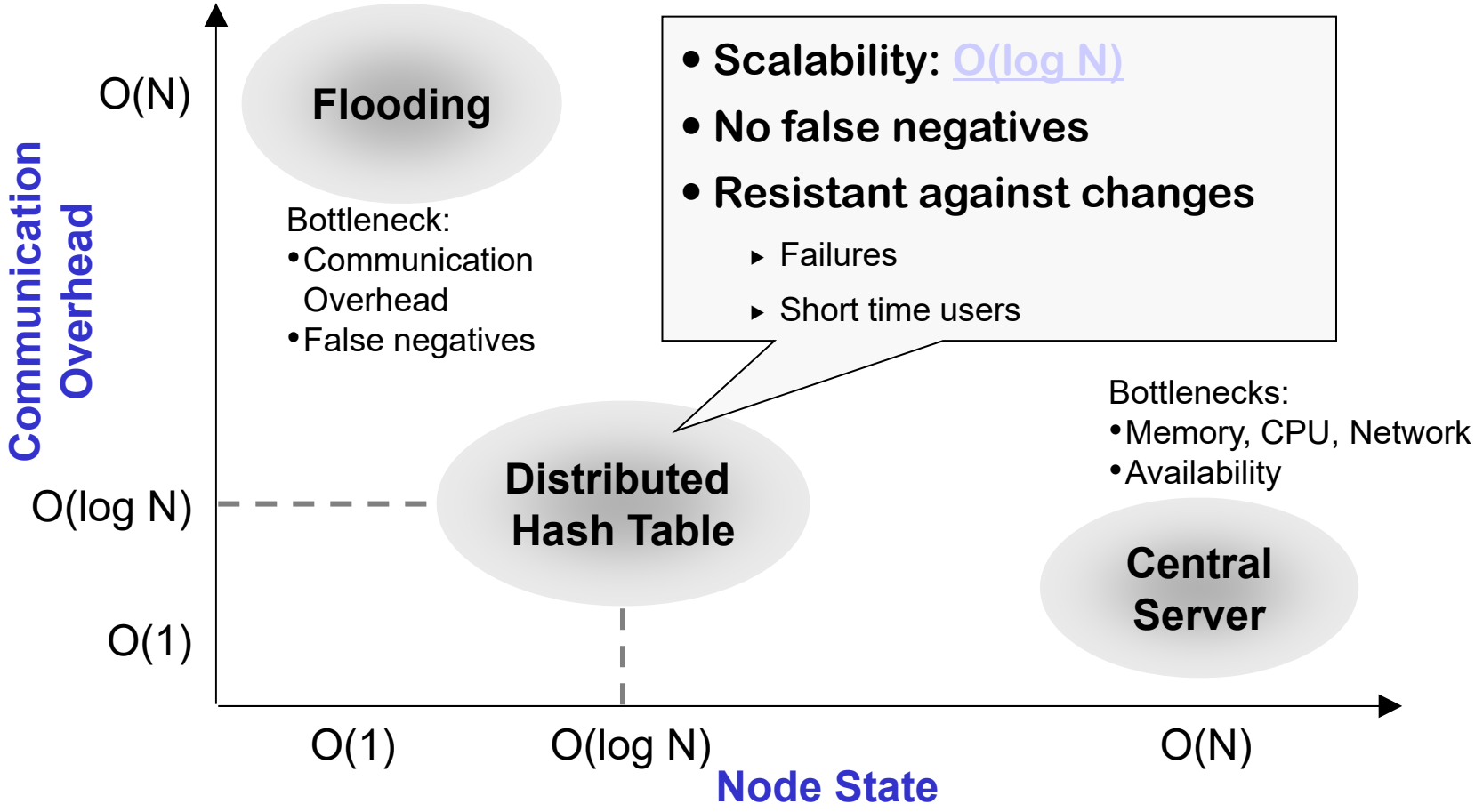
Distributed Indexing (1)

- Communication overhead vs. node state



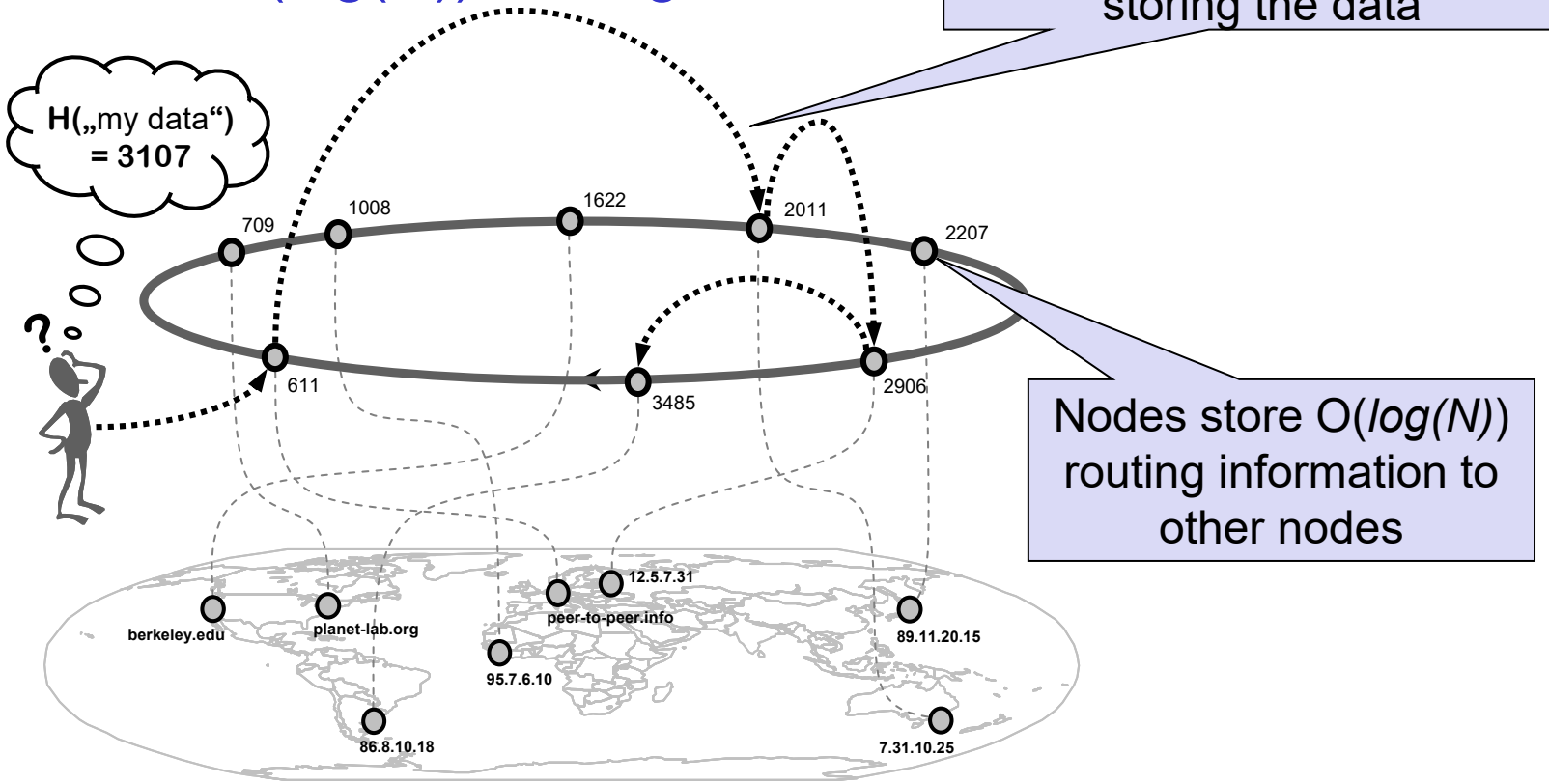
Distributed Indexing (2)

- Communication overhead vs. node state



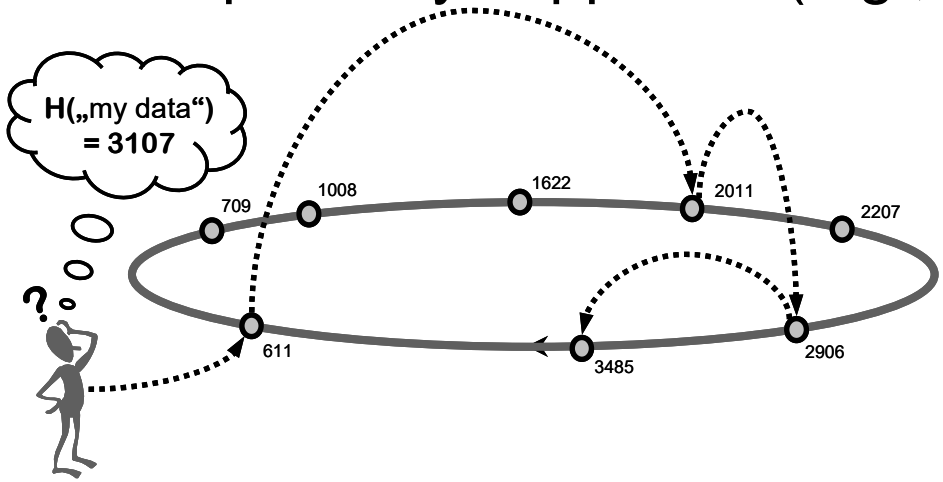
Distributed Indexing (3)

- Goal is scalable complexity for
  - Communication effort: O(log(N)) hops
  - Node state: O(log(N)) routing entries



Distributed Indexing (4)

- Approach of distributed indexing schemes
  - Data and nodes are mapped into same address space
  - Nodes maintain routing information to other nodes
    - Definitive statement of existence of content
- Problems
  - Maintenance of routing information required
  - Fuzzy queries not primarily supported (e.g., wildcard searches)



Comparison of Lookup Concepts

System	Per Node State	Communi-cation Overhead	Fuzzy Queries	No false negatives	Robust-ness
Central Server	O(N)	O(1)	✓	✓	✗
Flooding Search	O(1)	O(N)	✓	✗	✓
Distributed Hash Tables	O(log N)	O(log N)	✗	✓	✓

© 2022 UZH, CSG@IfI

Fundamentals of Distributed Hash Tables

Distributed Data Management

Addressing, Routing, Data Storage

© 2022 UZH, CSG@IfI

Fundamentals of Distributed Hash Tables

- Challenges for designing DHTs
  - Desired characteristics
    - Reliability / Scalability
  - Equal distribution of content among nodes
    - Crucial for efficient lookup of content
  - Permanent adaptation to faults, joins, and exits of nodes
    - Assignment of responsibilities to new nodes
    - Re-assignment and re-distribution of responsibilities in case of node failure or departure

Hash Table vs. Distributed Hash Table

- Hash Table
  - Bucket = hash(x) mod n
  - If n changes, remapping / bucket changes needed
  - N changes if capacity is reached
    - HashMap.class: static final float DEFAULT_LOAD_FACTOR = 0.75f;
  - Remapping is expensive in DHT!
    - Why?
      - The hash range N change leads to a rehashing of EVERY resource (node and content)!
- Distributed Hash Table
  - Consistent hashing

    → nodes responsible for hash value intervals
  - More peers = smaller responsible intervals

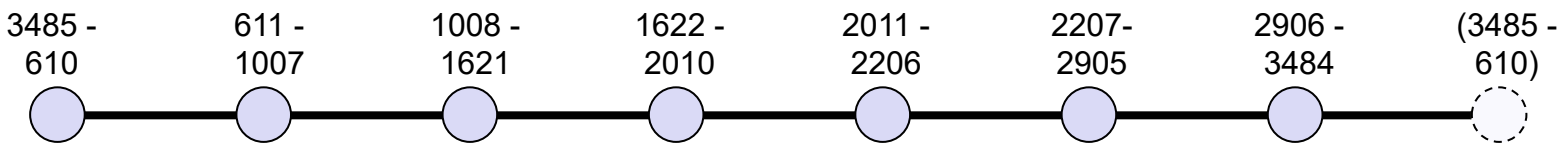
Distributed Management of Data

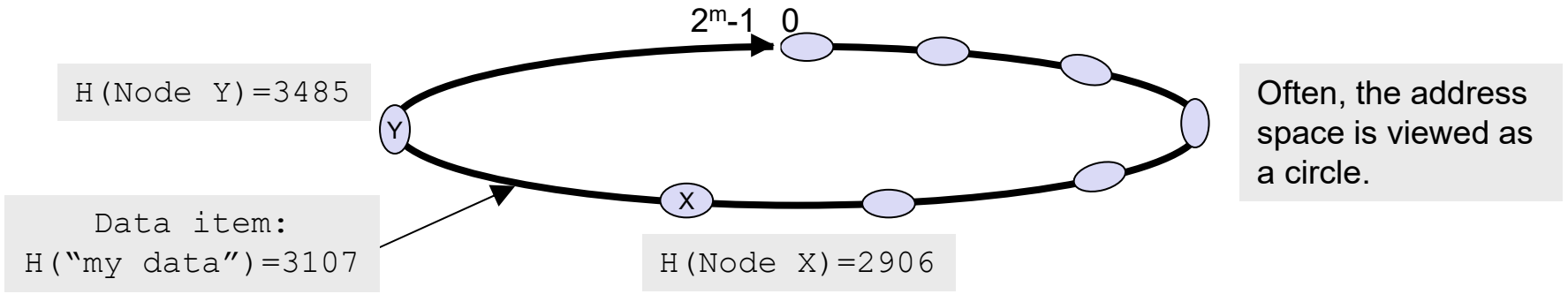
Sequence of operations

1. Mapping of nodes and data into same address space
   - Peers and content are addressed using flat identifiers (IDs)
   - Common address space for data and nodes
   - Nodes are responsible for data in certain parts of the address space
   - Association of data to nodes may change since nodes may disappear
2. Storing / Looking up data in the DHT
   - Search for data = routing to the responsible node
     - Responsible node not necessarily known in advance

Addressing in Distributed Hash Tables

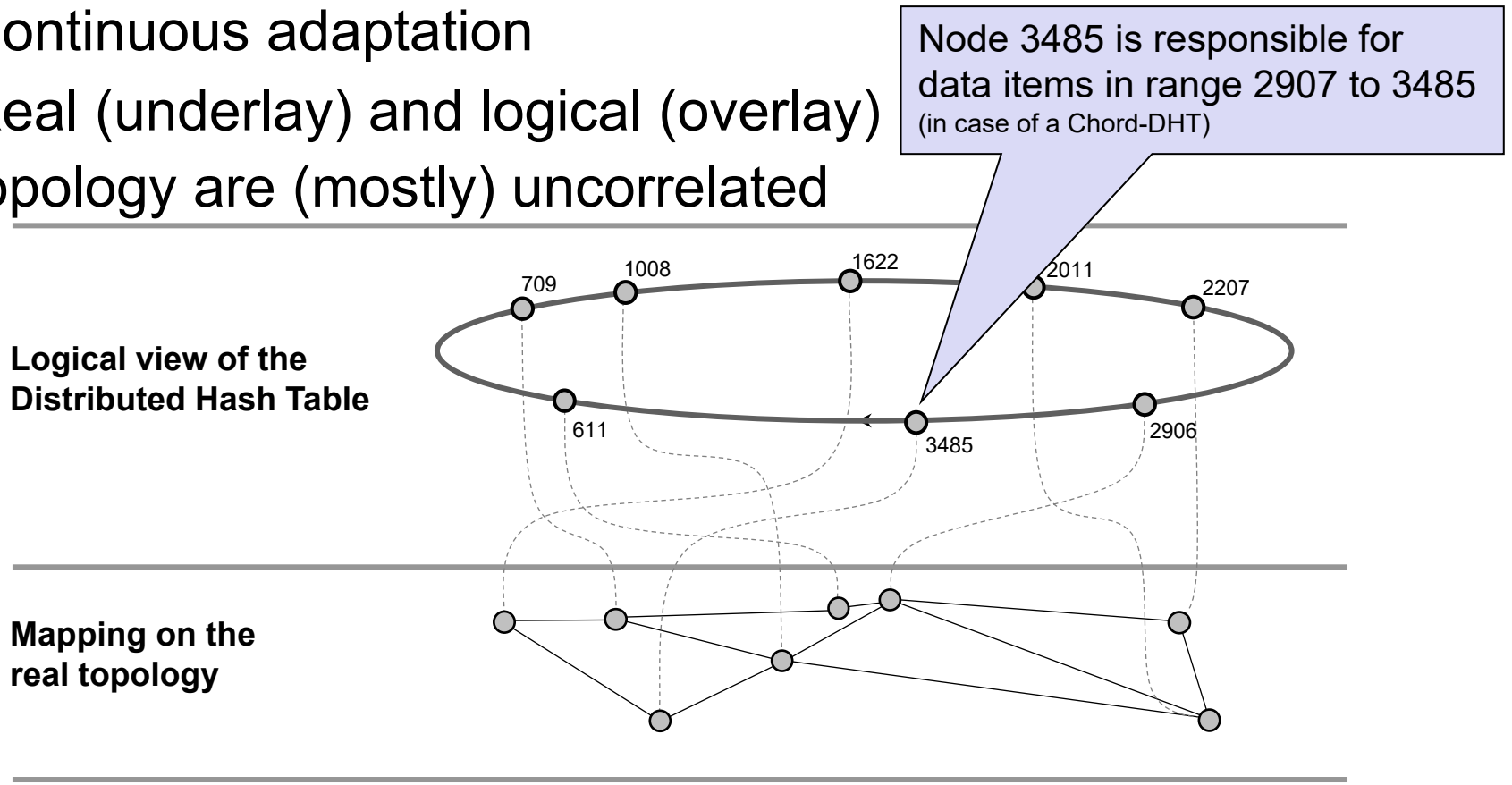
- Step 1: Mapping of content/nodes into linear space
  - Usually: 0, …, 2^m-1 >> number of objects to be stored
  - Mapping of data and nodes into an address space (with hash function)
    - E.g., Hash(*String*) mod 2^m: H(„*my data*“) → 3107





Association of Address Space with Nodes

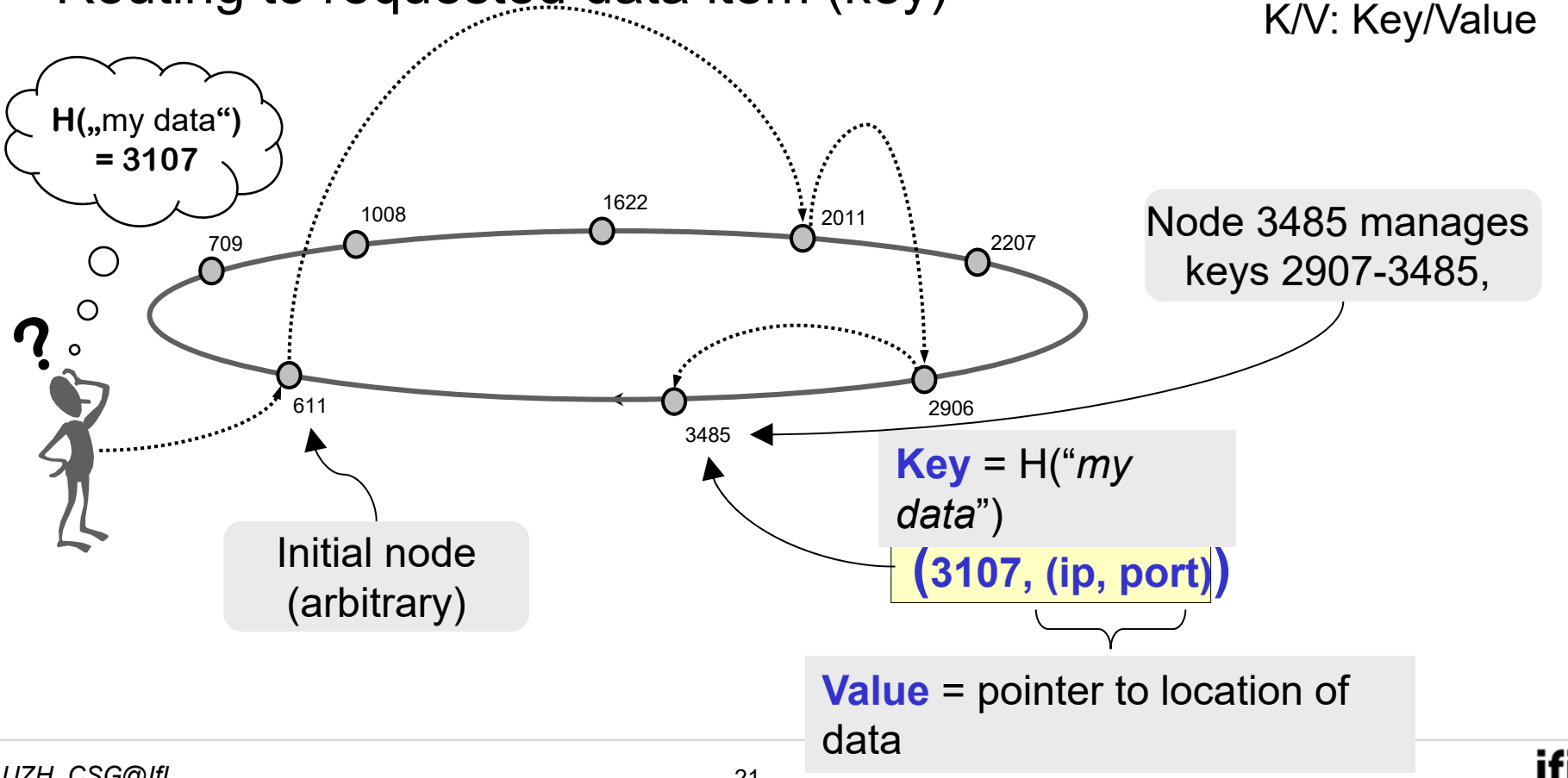
- Each node is responsible for part of the value range
  - Often with redundancy (overlapping of parts)
  - Continuous adaptation
  - Real (underlay) and logical (overlay) topology are (mostly) uncorrelated



Step 2: Routing to a Data Item (1)

- Step 2: Locating the data/routing to a K/V-pair
  - Start lookup at arbitrary node of DHT
  - Routing to requested data item (key)

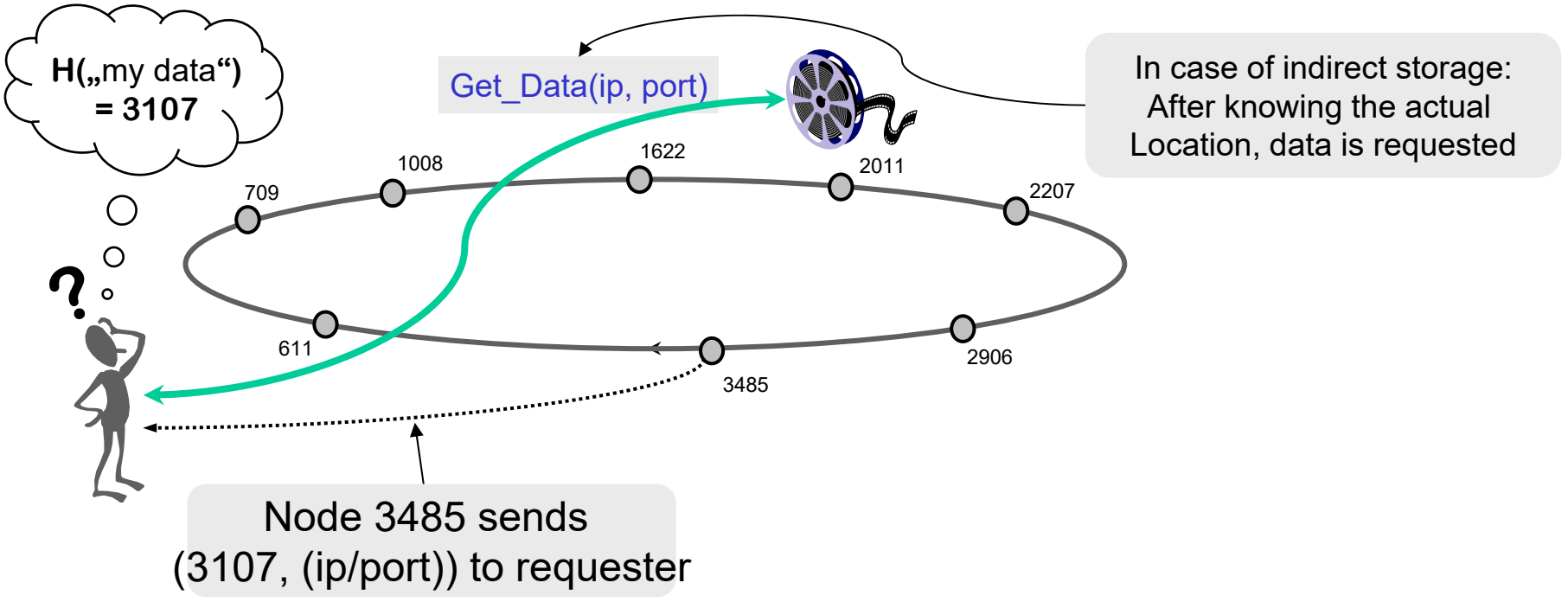
K/V: Key/Value



Step 2: Routing to a Data Item (2)

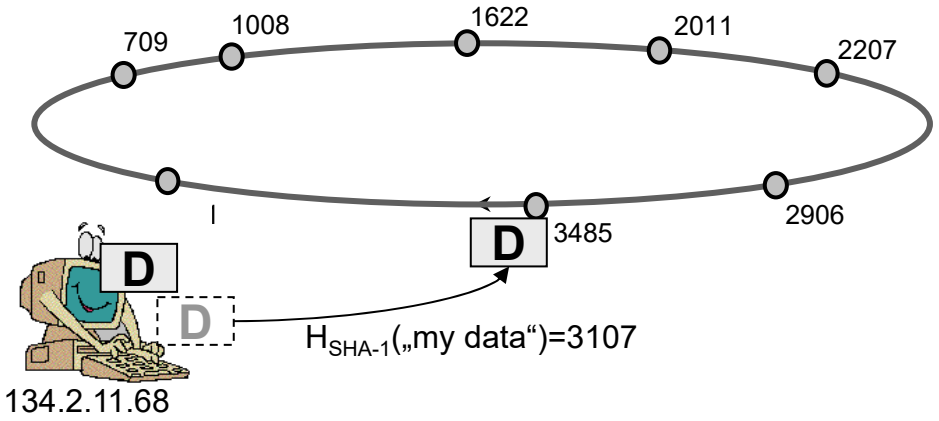
- And getting the content
  - K/V-pair is delivered to requester
  - Requester analyzes K/V-tuple

(and downloads data from actual location – in case of indirect storage)



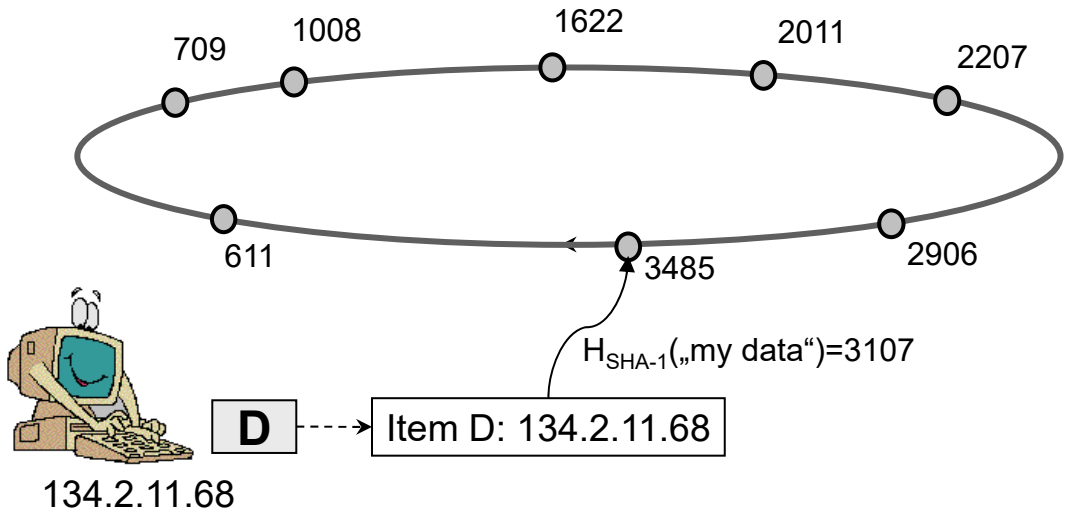
Association of Data with IDs – Direct Storage

- How is content stored on the nodes?
  - Example:
    H("*my data*") = 3107 is mapped into DHT address space
- Direct storage
  - Content is stored in responsible node for H("*my data*")
    → Inflexible for large content – o.k., if small amount data (~kB)



Association of Data with IDs – Indirect Storage

- Indirect storage
  - Nodes in a DHT store tuples like (key, value)
    - Key = Hash(„*my data*”) → 2313
    - Value is often the real storage address of content: (IP, Port) = (134.2.11.140, 4711)
  - More flexible, but one step more to reach content



DHT Mechanisms

Node Arrival

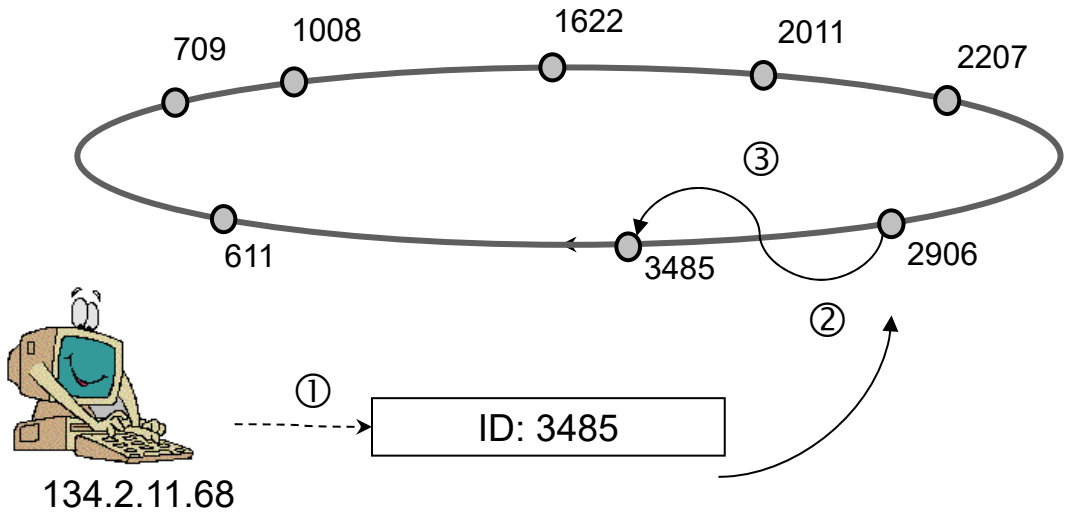
Node Failure and/or Departure

Node Arrival

- Joining of a new node

PK: Public Key

1. Calculation of node ID (normally random / or based on PK)
2. New node contacts DHT via arbitrary node (bootstrap node)
3. Lookup of its node ID (routing)
4. Copying of K/V-pairs of hash range (in case of replication)
5. Notify neighbors



Node Failure and/or Departure

- Failure of a node
  - Use of redundant K/V pairs (if a node fails)
  - Use of redundant / alternative routing paths
  - Key-value usually still retrievable if at least one copy remains

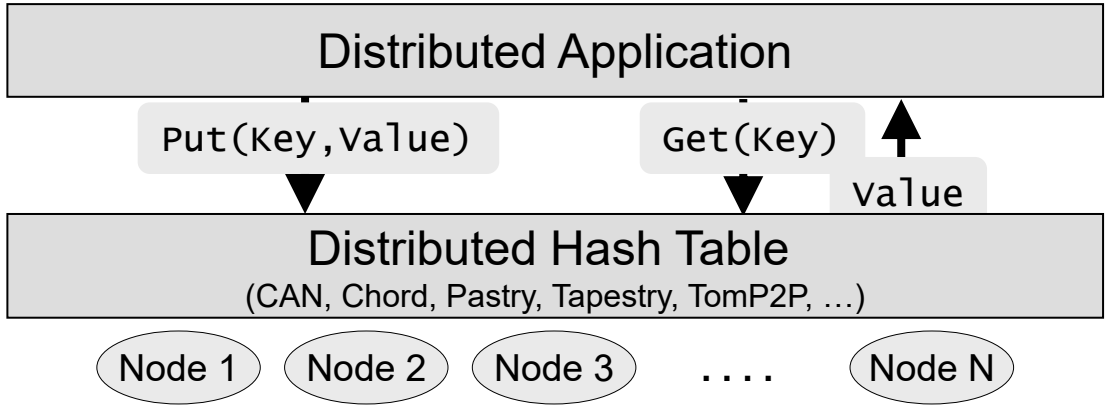
- Departure of a node
  - Copying of K/V pairs to corresponding nodes
    - Can be before or after unbinding
  - Friendly unbinding from routing environment
    - If unbinding is unfriendly, need for keep-alive messages

DHT Summary

DHT Interfaces, Properties of DHTs,
DHT Implementations

DHT Interfaces

- Generic interface of distributed hash tables
  - Provisioning of information: Put(key, value)
  - Requesting of information (search for content): Get(key)
  - Reply: Value
- DHT approaches are interchangeable (with respect to interface)



Properties of DHTs

- Use of routing information for efficient search for content
- Keys are evenly distributed across nodes of DHT
  - No bottlenecks
  - Failure of nodes can be tolerated
  - Survival of some attacks possible
- Self-organizing system
- Simple and efficient prototyping and implementation
- Supporting a wide spectrum of applications
  - Flat (hash) key without semantic meaning
  - Value depends on application

DHT Implementations

- Examples of generic Distributed Hash Tables
  - Pastry (freepastry, v2.1, 13.3.2009)
    Microsoft Research, Rice University
  - Chord (v1.0.5, 11.4.2008)
    UC Berkeley, MIT
  - Tapestry (chimera 1.20 ~2007)
    UC Berkeley
  - CAN (~2001)
    UC Berkeley, ICSI
  - P-Grid (v3.2.0_822, 25.11.2008)
    EPFL Lausanne
  - BEP 44 Storing arbitrary data in the DHT 1KB, many implementations
  - TomP2P (v5b8, 27.5.2015)
- … and there are plenty of others …

Kademlia – DHT Example

Routing Concept, Protocol, Routing Table Construction,

Peer Selection Policy, Enhancements

© 2022 UZH, CSG@IfI

Kademlia (1)

- Use α parallel iterative lookups to locate data
  - Retrieves data faster
  - Overcomes faulty nodes *"concurrency"*
  - Usually, α = 3
- DHT-based overlay networks use XOR distance metric
  - Simple operation
  - Symmetrical routing paths
    (A → B == B → A)
    - due to $d_{XOR}(A,B) == d_{XOR}(B,A)$
- Store data with key X on k nodes closest to X according to XOR
  - "built in" replication ensuring data availability
  - Usually, k = 20

Kademlia (2)

- Use lookup messages to maintain the overlay network
  - Learn useful routing information from received lookup requests

XOR Distance Calculation

ID Node A: 110101
ID Node B: 010001

$d_{XOR}(A,B) = d(110101,010001)$

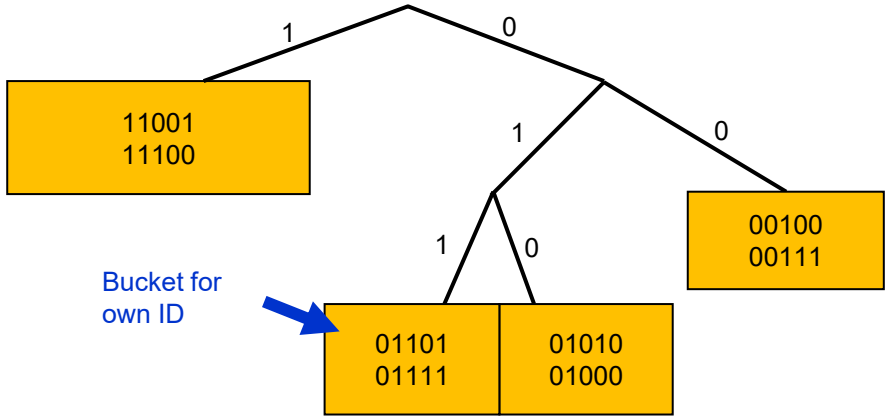
1 1 0 1 0 1
XOR
0 1 0 0 0 1
↓
1 0 0 1 0 0

$d_{XOR}(A,B) = 1\,0\,0\,1\,0\,0_2 = 36_{10}$

Concept of the Kademlia Routing

- Structure of routing table
  - Every node maintains a binary tree like routing table
  - Tree branches along the local node ID
  - Every leave is a bucket with k entries

Example routing table for node 01110 with bucket factor k = 2



Lookup procedure for a key X

Step 1: Traverse routing table tree and pick bucket with node IDs closest to key X

Step 2: Put node IDs in a node list

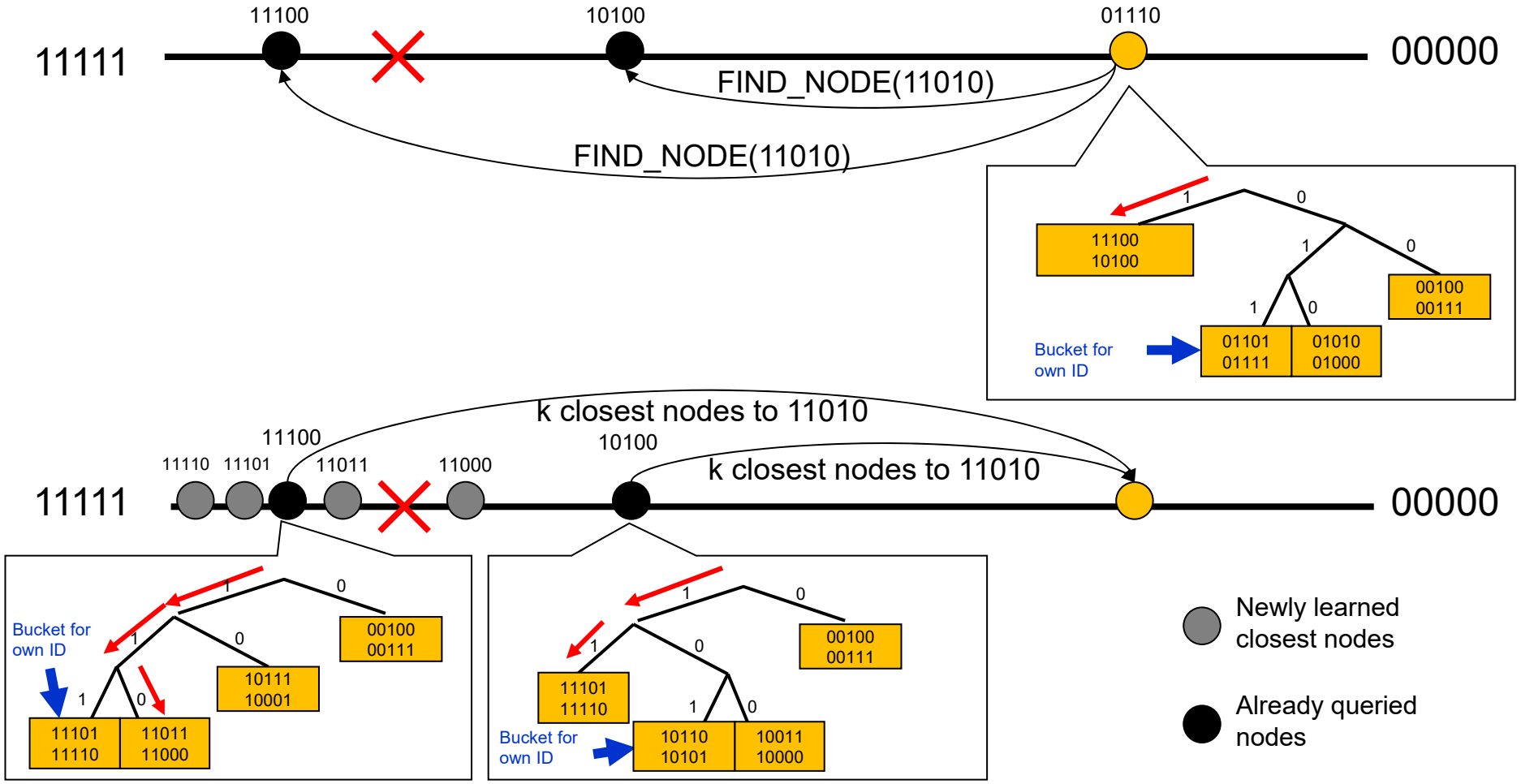
Step 3: Send α closest nodes request to nodes in the list closest to X

Step 4: Put received node IDs in node list

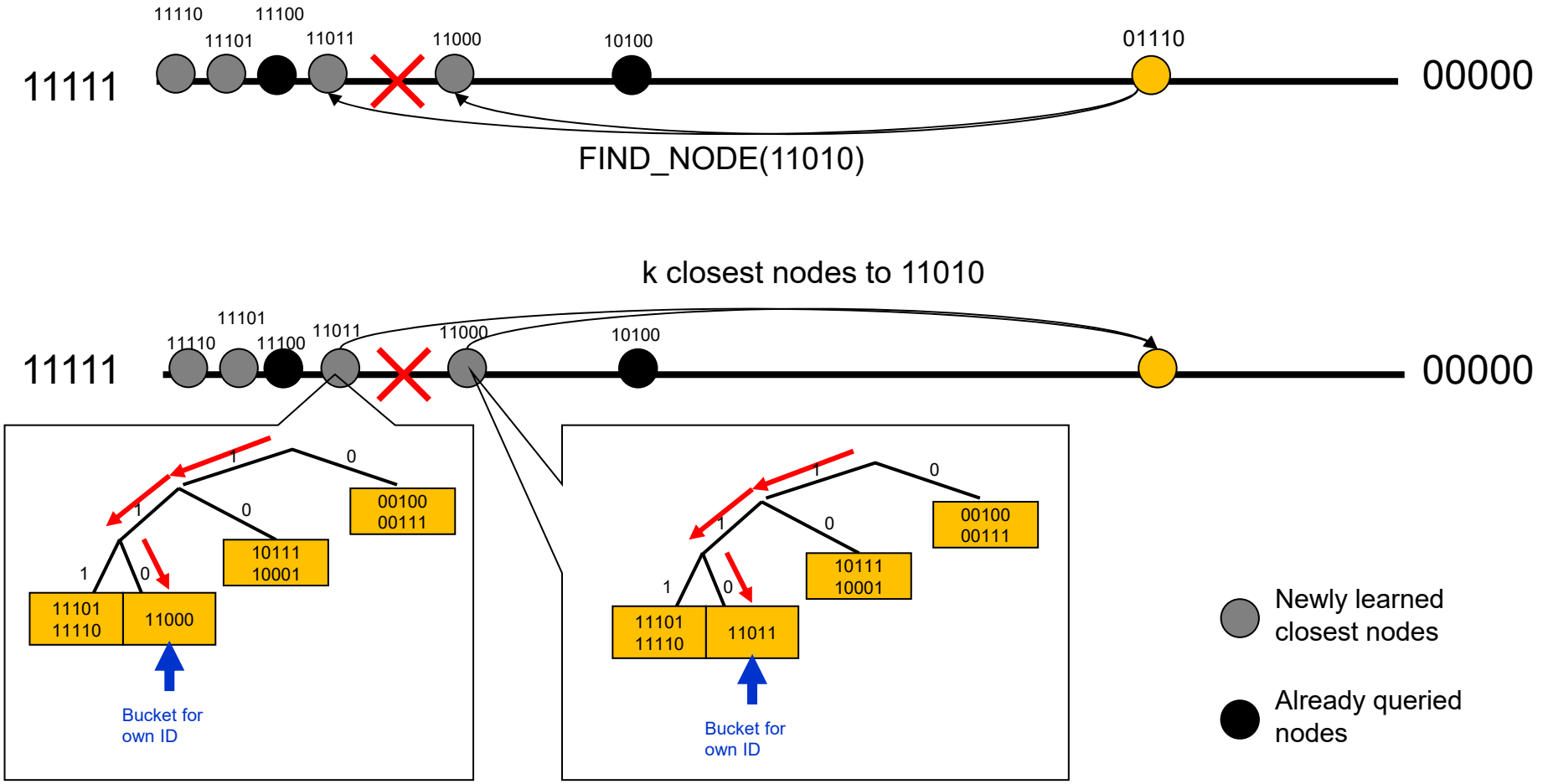
Step 5: Repeat until Step 3 and 4 until set of k closest node do not change anymore

Step 6: Pick k-closest nodes from node list and send store or get data request to them

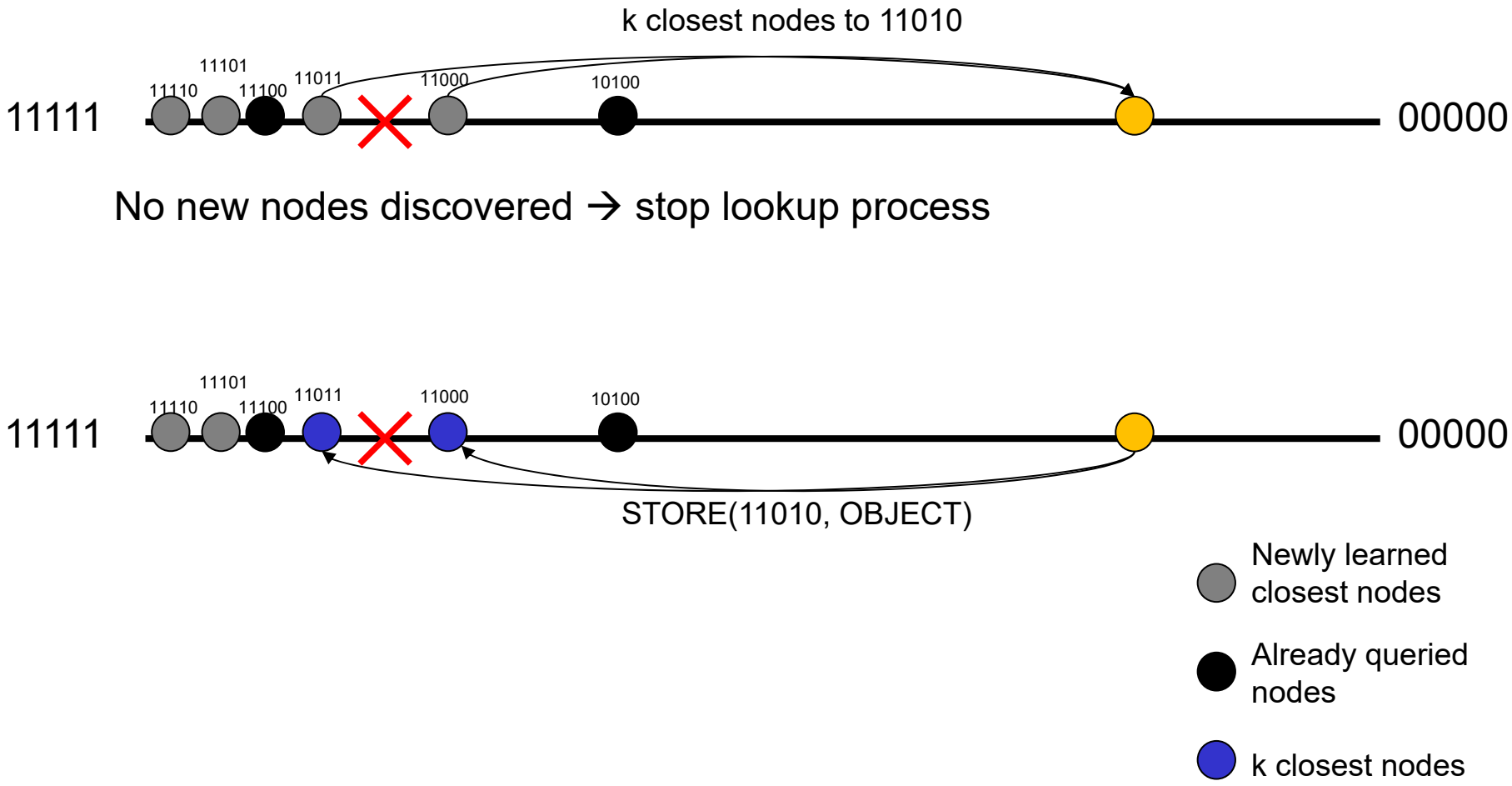
Routing from 01110 to 11010, α = 2, k = 2



Routing from 01110 to 11010, α = 2, k = 2



Routing from 01110 to 11010, α = 2, k = 2



Kademlia Protocol

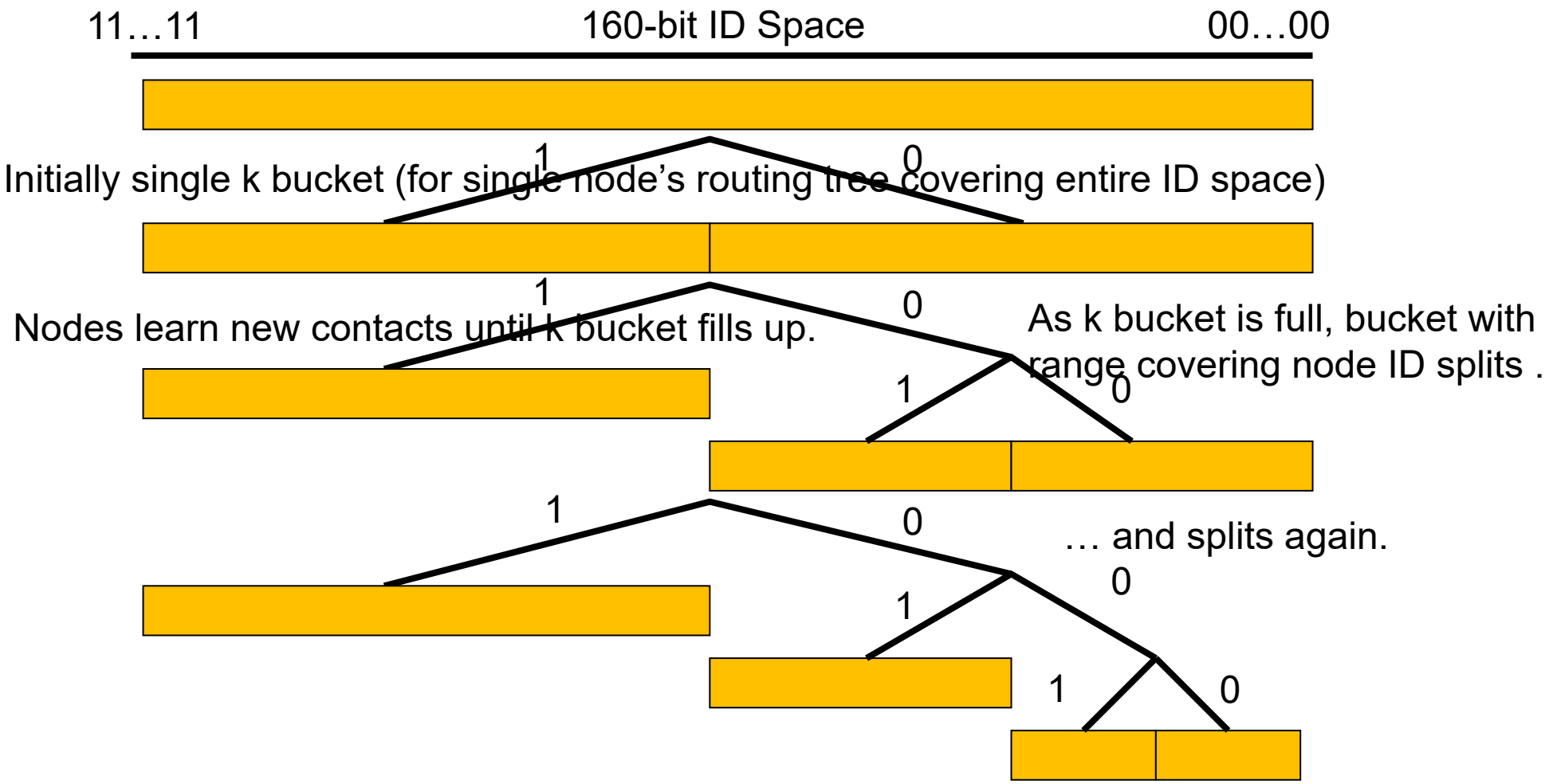
- The Kademlia protocol consists of 4 RPCs
  - FIND_NODE(KEY)
    - Recipient returns <IP Address, UDP Port, Node ID> triples for k closest nodes he knows about
  - FIND_VALUE(KEY)
    - Like FIND_NODE
    - With exception:
      - If recipient already stores the value, the value is returned instead of k closest nodes
  - PING(IP ADDRESS)
    - Probes a node to see if it is online
  - STORE(KEY, VALUE)
    - Instructs a node to store a <key,value> pair

RPC: Remote Procedure Call

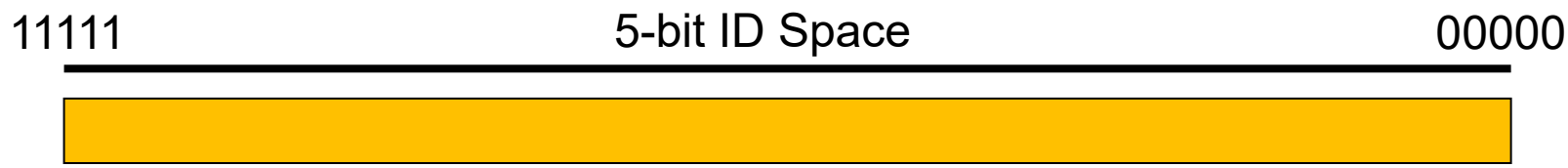
Construction of Routing Table

- Each node maintains routing table (k buckets)
  - Routing tables of different peers may be different
- For each 0 <= i < 160 every node (160 bit long IDs)
  - keeps a list of <IP Address, UDP Port, Node ID> triples
  - for k nodes within range [2 ^i ; 2 ^(i+1)[
  - in total k * 160 contacts
- Nodes learn from
  - messages they receive or using the FIND_NODE method
- Preference toward old contacts
  - Study has shown that the longer a node has been up, the more likely it is to remain up another hour
  - Resistance against DoS attacks by flooding network w/ nodes

Evolution of the k Buckets (Dynamic Allocation)



Example for Node ID 01110, k=2



New node 11001:

11001

New node 01101:

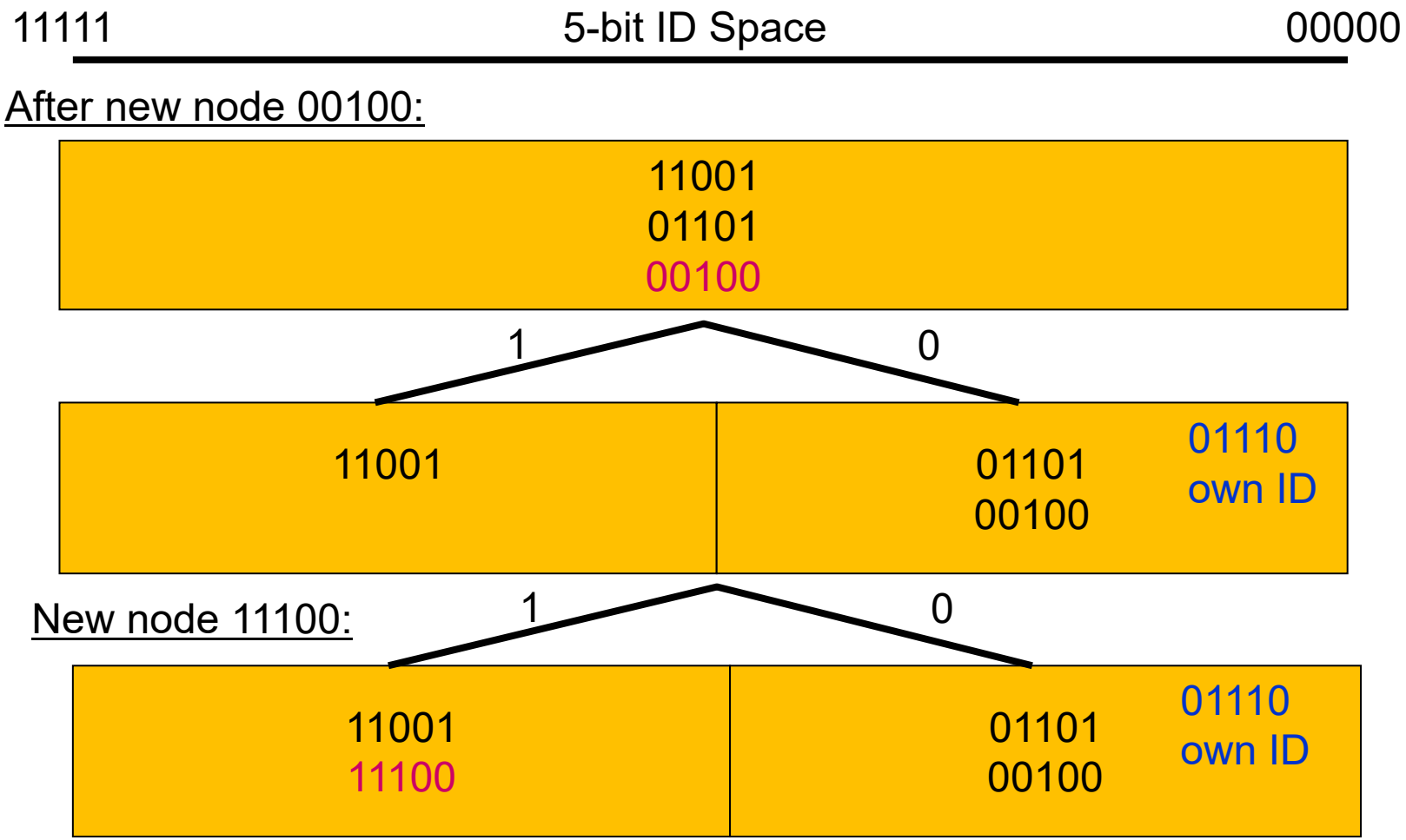
11001
01101

New node 00100:

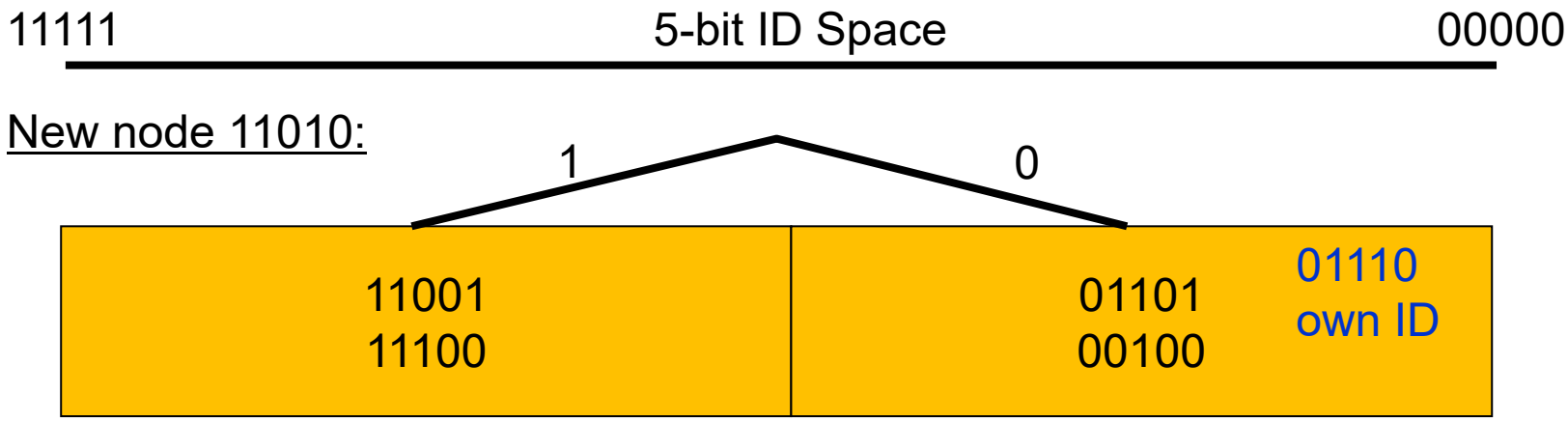
11001
01101
00100 but … no because 3rd, and k=2 is max.

k bucket is full → split necessary

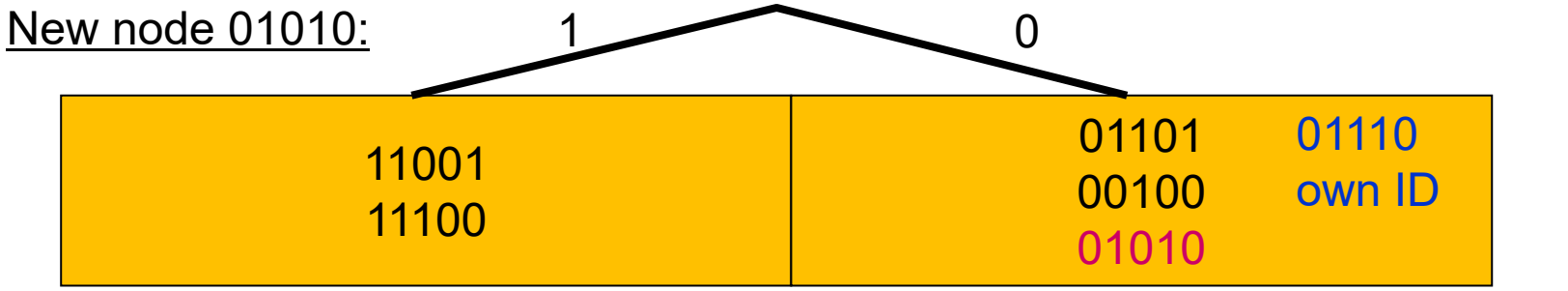
Example for Node ID 01110, k=2



Example for Node ID 01110, k=2

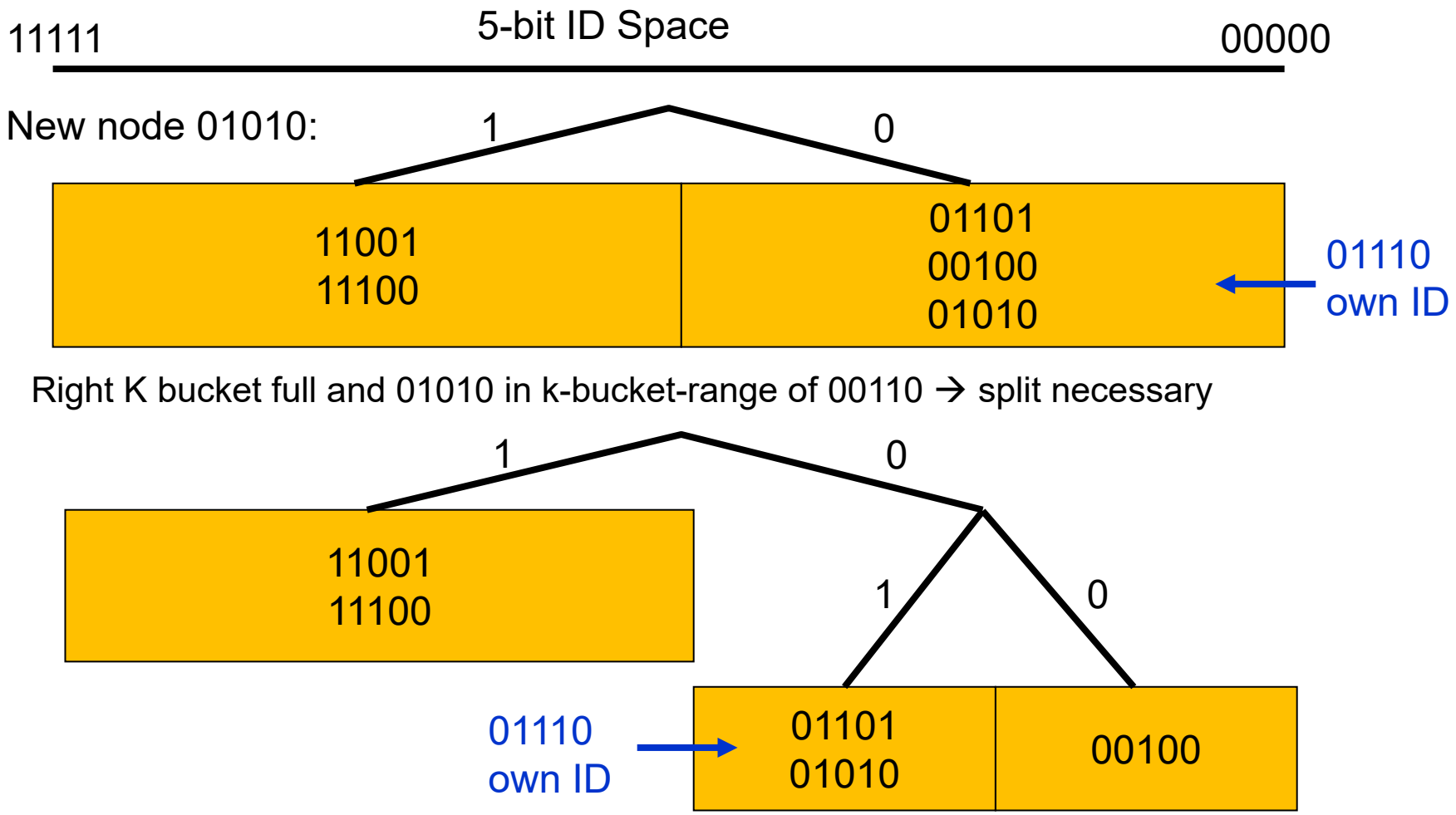


Left k bucket full and 11010 NOT in k-bucket-range of 01110 → node is dropped

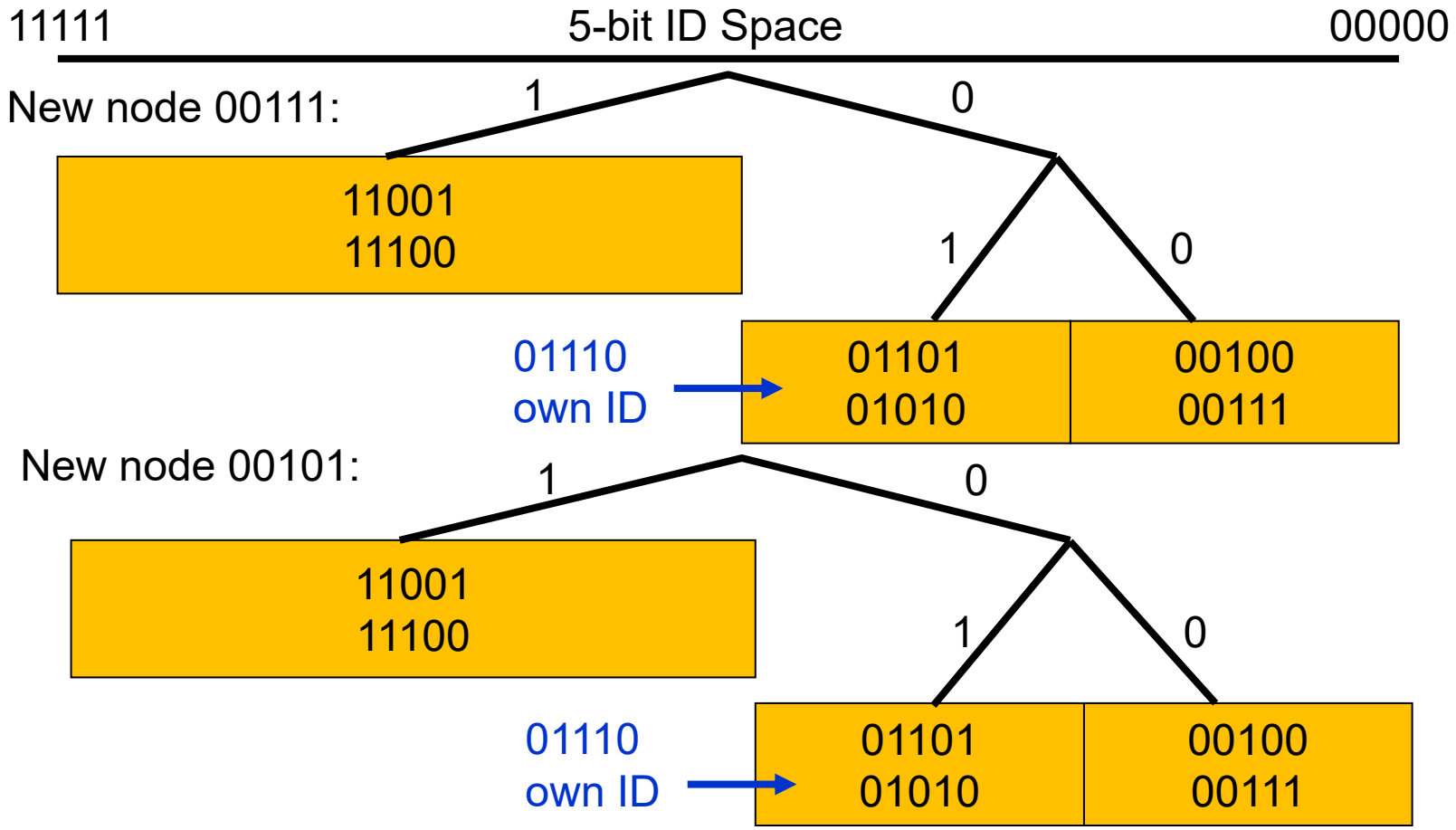


Right k bucket full and 01010 in k-bucket-range of 01110 → split necessary

Example for Node ID 01110, k=2

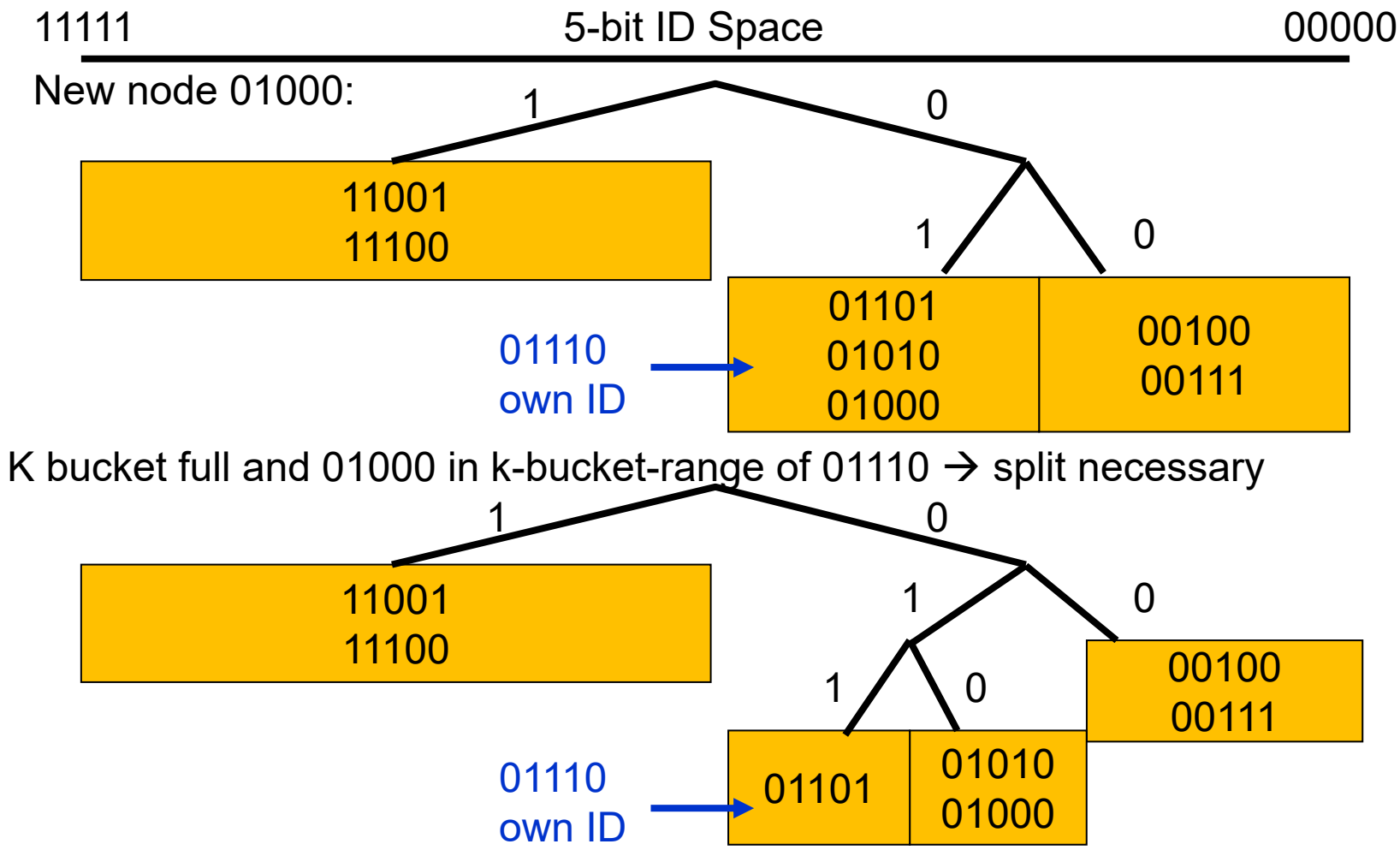


Example for Node ID 01110, k=2



Left K bucket full and 00101 NOT in k-bucket-range of 01110 → node is dropped

Example for Node ID 01110, k=2



k Buckets: Peer Selection Policy

- If node u receives a message from node v then it adds node v to its k-bucket according to the following rules:
  - IF v is already in a k-bucket
    THEN move v to the tail of the bucket
  - IF v is not in the k-bucket and the bucket has fewer than k entries
    THEN insert recipient to the tail of list
  - IF the appropriate k bucket is full AND least recently seen node is alive
    THEN move least recently seen node to tail of bucket and discard node v
  - IF the appropriate k bucket is full AND least recently seen node is not alive
    THEN remove least recently seen node from bucket and add node v at the tail
- Note: approx. k = 20 in the real world

Kademlia: Enhancements

- Parallel queries
  - For one query, α (alpha) concurrent lookups are sent
  - More traffic load, but lower response times
- Network maintenance
  - In Chord: active fixing of fingers
  - In Kademlia: learning for bypassing queries
  - Check if peer IDs fit better in routing table
- Large routing tables
  - In Chord: 1 finger per distance 2^i to 2^(i+1)
  - In Kademlia: k contacts per distance 2^i to 2^(i+1)
  - Increased robustness

Kademlia: Key Characteristics

- XOR-based metric topology reaches
  - Consistency of nodes and content identification
  - Reasonable performance based on k buckets scheme
  - Latency-minimized routing (search)
  - Symmetric and unidirectional topology
- Kademlia's concurrency parameter α
  - Enables nodes to "trade" constant bandwidth factor with asynchronous lowest-latency hop selection and delay-free failt recovery
- Kademlia has proven that
  - Node failures are inversely related to uptime!

References

- P. Maymounkov, D. Mazieres: *Kademlia: A Peer-to-peer Information System Based on the XOR Metric;* New York University (at best found via Google), 2002
  - https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf
- Klaus Wehrle, Stefan Götz, Simon Rieche: *Distributed Hash Tables (Section 7);* R. Steinmetz, K. Wehrle (Edts.) Peer-to-peer Systems and Applications, LNCS Vol. 3485, Springer, Heidelberg, Germany, 2005
  - https://pdfs.semanticscholar.org/87df/ba92ddcb21f9372ad9ac4f7763f3e257cd06.pdf

Blockchain Basics



Universität Zürich[UZH]

CSG
Communication Systems Group

ToC

- Blockchain Motivation and Basics
  - Linked Lists, Hashes and Cryptography, Data Bases
  - Definition, Who-owns-what Records, Token Types
- Blockchain Operations I
  - Ingredients, Protocol, Data Structure, Transactions (tx)
  - Hashing and Consensus
  - Smart Contracts
- Blockchain Operations II
  - UTXO and Account Model, tx Fees, Addresses
- Blockchain Functionality and Terminology
  - Time-stamping, Segregated Witness
  - Key Terms

© 2022 UZH, CSG@IfI

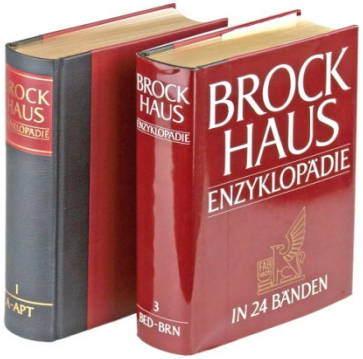
Blockchain Motivation and Basics

The Digitization of Society

Physical Objects



Telegram



Encyclopedia



Money

Ended Dec 29, 2017 in Belgium

Since mid 70's, RFC 524

Since 2001

Since 2009

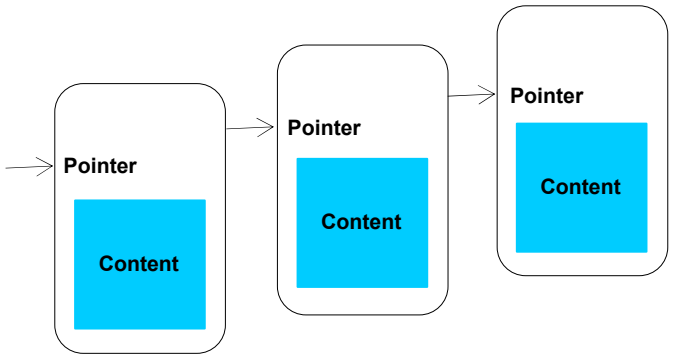
Bitcoins

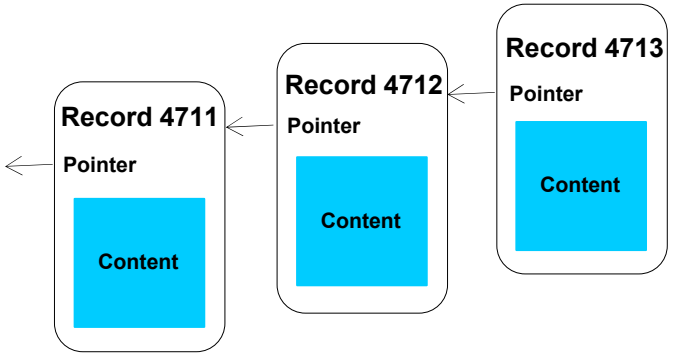
Digitized Representations

Systems operated as decentralized and distributed systems!

Excursus 1 – Data Type “Linked List”

- Linear collection of data elements (records)
  - Linear order of records is given by pointers to the next record
  - Data structure as a group of nodes represents an implicit sequence
  - Example: backward linked list
  - Data structure as a group of nodes represents an explicit sequence due to record identifiers added



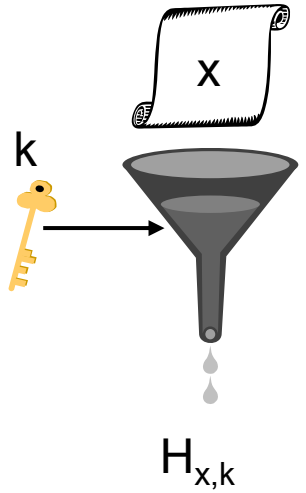


© 2022 UZH, CSG@IfI

Excursus 2 – Hashes and Cryptography

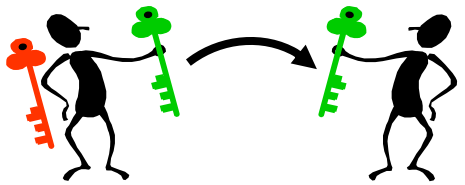
- Hash function
  - "One-way" function
    - Efficient generation: g(x) = H, result is termed "hash"
    - Impossible to calculate $g^{-1}(x)$
    - Very low collision possibility: x, x' with H(x)=H(x')
      - Examples: (MD5), RIPEMD-160, SHA-256

x: Plain text
k: Key



- Asymmetric cryptography (public key cryptography)
  - Encryption defines a function $f(x)$, x ➔ y
  - Decryption defines the inverse f^{-1}, y ➔ x
  - Key pair of
    - public part: (e, pq) – published to anyone
    - private part: (p, q, d) – the "secret"
    - Example: RSA

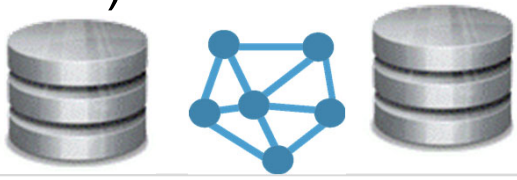
$y = x^e \bmod pq$: cypher text
$d = e^{-1} \bmod (p-1)(q-1)$
p, q: (large) prime numbers



Excursus 3 – Databases

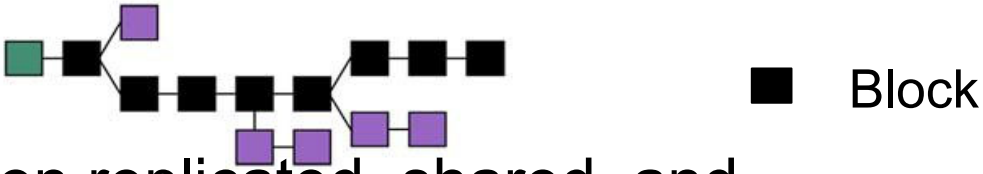
- Databases (DB)
  - Access control for users and the “SysAdmin” (trusted root)
    - Centralized, physical, and trusted servers are maintained
  - Potentially central point of failure, loss, or misuse
  - Exchange of records between clients/database via commands
    - SQL statements include INSERT, SELECT, UPDATE, DELETE, …
  - Databases typically maintain the ACID principle on transactions
    - Atomicity, Consistency, Isolation, Durability (ACID)
    - However, records updated are updated and records deleted are deleted → No storage of any history (standard case)
- Distributed DBs
  - Every copy runs on a trusted node





Distributed Ledgers

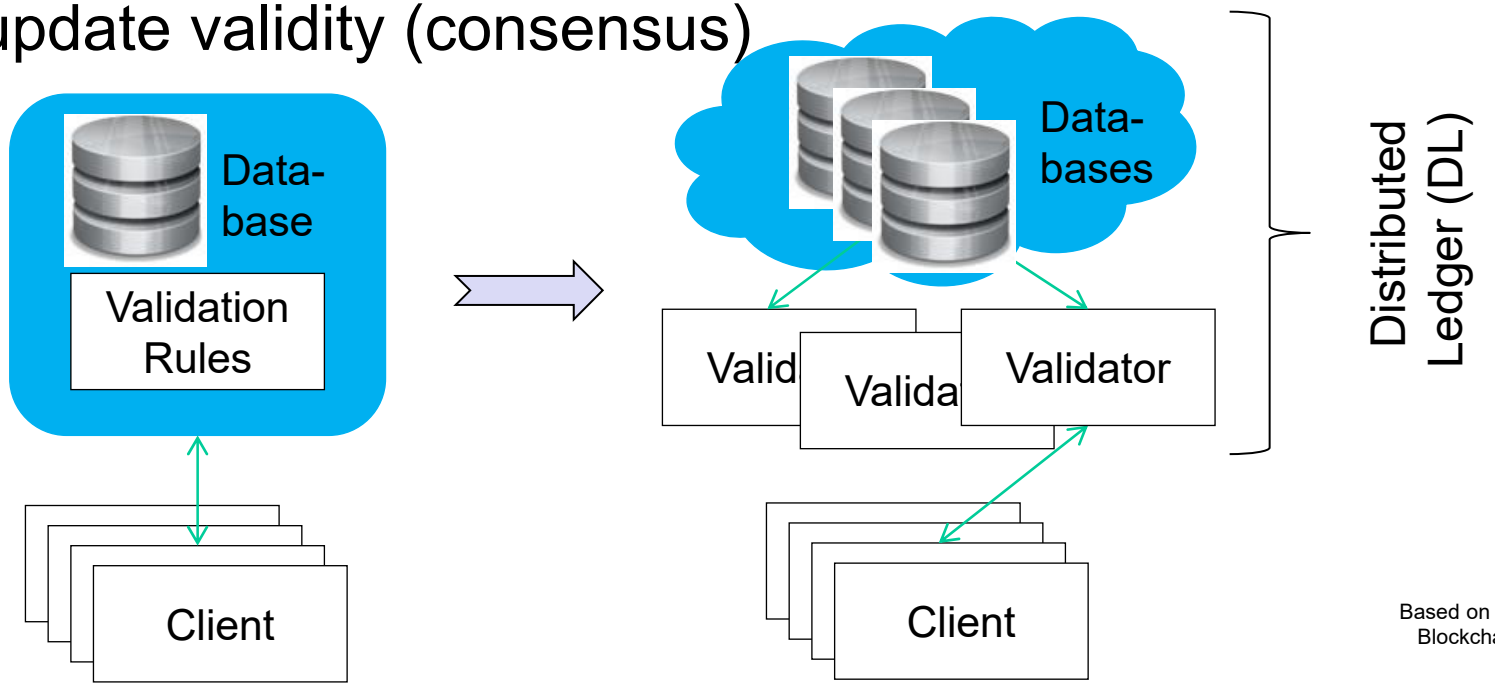
- Can data be stored fully decentralized and handled reliably between non-trusted stakeholders?
  - Unstructured/structured data stored across the world by anyone
  - Access control by “all” w/o a central root
  - No central point, redundant copies, non-trusted participants, and detectable misuse → Distributed (Shared) Ledger (DL)



  - DL defines a “consensus” on replicated, shared, and synchronized digital data (blocks), geographically spread across many sites on earth
    - Every node participating may hold a copy of the full DL, thus, no copy is trusted differently and no formal main “no. 1” instance exists

Key Idea: “Replacing” (Central) Databases

- Distributed Ledgers replace clients’ access-protected writes to an authoritative database via validation rules by a distributed consensus of decentralized validators
  - where the database’s state depends on majority agreement of update validity (consensus)



Based on Terence Spies:
Blockchain Mechanics

Definition

- [Distributed Ledgers (DL) or] Blockchains (BC)
  - Digital records of who-owns-what w/o a central storage



"Private" BC "Public" BC

  - Consensus Mechanism (CM) ensures that each node's copy of the ledger is identical to every other node's copy



  - Write access to BCs by miners or validators (with data from any asset owner) for transactions via CM and cryptographic signatures, read access at no "costs"



- Key advantages of (public) BCs
  - Immutable, traceable, and preventing "double spending"



“Who-owns-what” Records

- A digital asset = an electronic representation, *e.g.*, file
  - Inherently bears the exclusive right of use of this file
- A token (digital token) = digital asset
  - Issued by a stakeholder, giving right to participate within that network of stakeholders
  - It may allow for “payments” inside that network
- A coin (digital coin) = electronic representation of value
  - Specifically designed to represent digital “money” within a network of stakeholders, typically the BC, and beyond
  - Counterfeiting and double-spending prevented by cryptography

⇨ One can buy a token with a coin, but generally not a coin with a token.

Token Types

- Utility tokens provide access digitally to an application or service by means of a blockchain
- Asset tokens represent assets such as

  (a) a debt or

  (b) an equity claim on the issuer

  [Finma Guidelines for ICOs, February 2018]

  – *E.g.*, they promise a share in future company earnings or future capital flows – analog to equities, bonds, or derivatives
- Payment tokens are used

  (a) as a means of payment for acquiring goods or services or

  (b) as a means of money or value transfer

  – Synonymous to "cryptocurrencies" (thus, a dedicated "coin")

    • Note: Cryptocurrencies give rise to no claims on their issuer

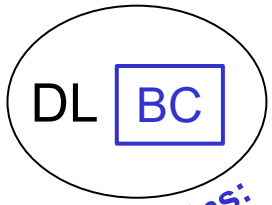
DL/BC Types and Terminology – Simplified

Private permissioned



- Read/write/consensus restricted to authorized nodes (pre-defined stakeholders)

– "Enterprise-grade" DL



No real blockchains: "restricted" stakeholders capabilities!

Public permissioned



- Write/consensus restricted to authorized nodes (pre-defined stakeholders)
- Read open to everyone

– "Controlled collaborative" DL

Private permissionless

- Write/consensus restricted to authorized nodes (pre-defined stakeholders)
- Read partially open



– "Consortium-grade" DL

Public permissionless



- Read/write/consensus open to everyone
- No restrictions and full transparency

– "Public" BC, **the** BC

The real and only blockchain (BC)!

Blockchain Operations I

Blockchain Ingredients (1)

- Public key cryptography and hashes
  - Asymmetric approach for arbitrary users
    - Ensures validation and authentication (in turn authorization)



- Internet
  - Networked infrastructure for everyone
  - Distributed system with arbitrary users and devices (nodes)
    - Peer-to-peer (overlay network) communication paradigms: protocol
    - Storage capabilities for “any”-sized data volumes



- Incentives
  - Supporting rewards for participants’ tasks performed within an overlay network by a “protocol” enabling communications
    - Ensures participation of anyone (potentially non-trusted stakeholders)



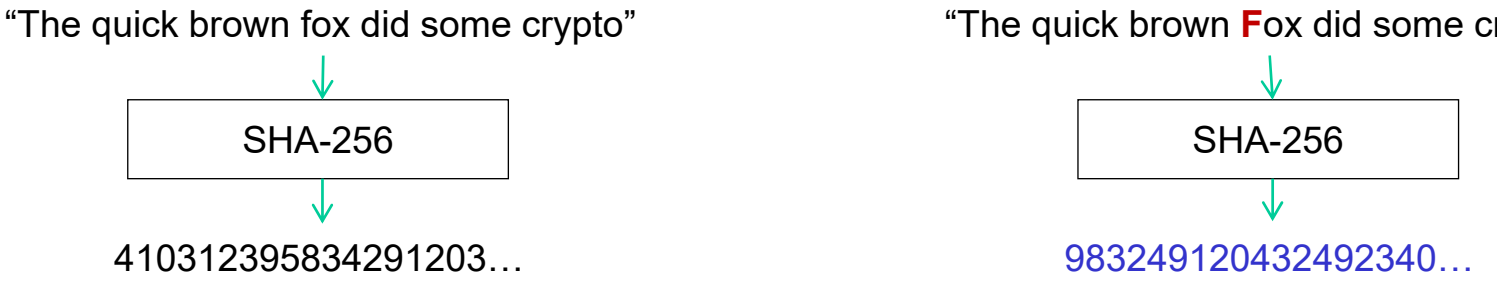
© 2022 UZH, CSG@IfI

Blockchain Ingredients (2)

- Current cryptography determines the basis for BCs
- Hashes

Based on Terence Spies: Blockchain Mechanics

  - A hash function (like SHA-256) takes a block of data and produces effectively a random fixed size integer, *e.g.*, 256 bit
  - Any change to the input randomizes the output
    - "Simple" input changes, *e.g.*, one char, lead to "dramatic" result changes



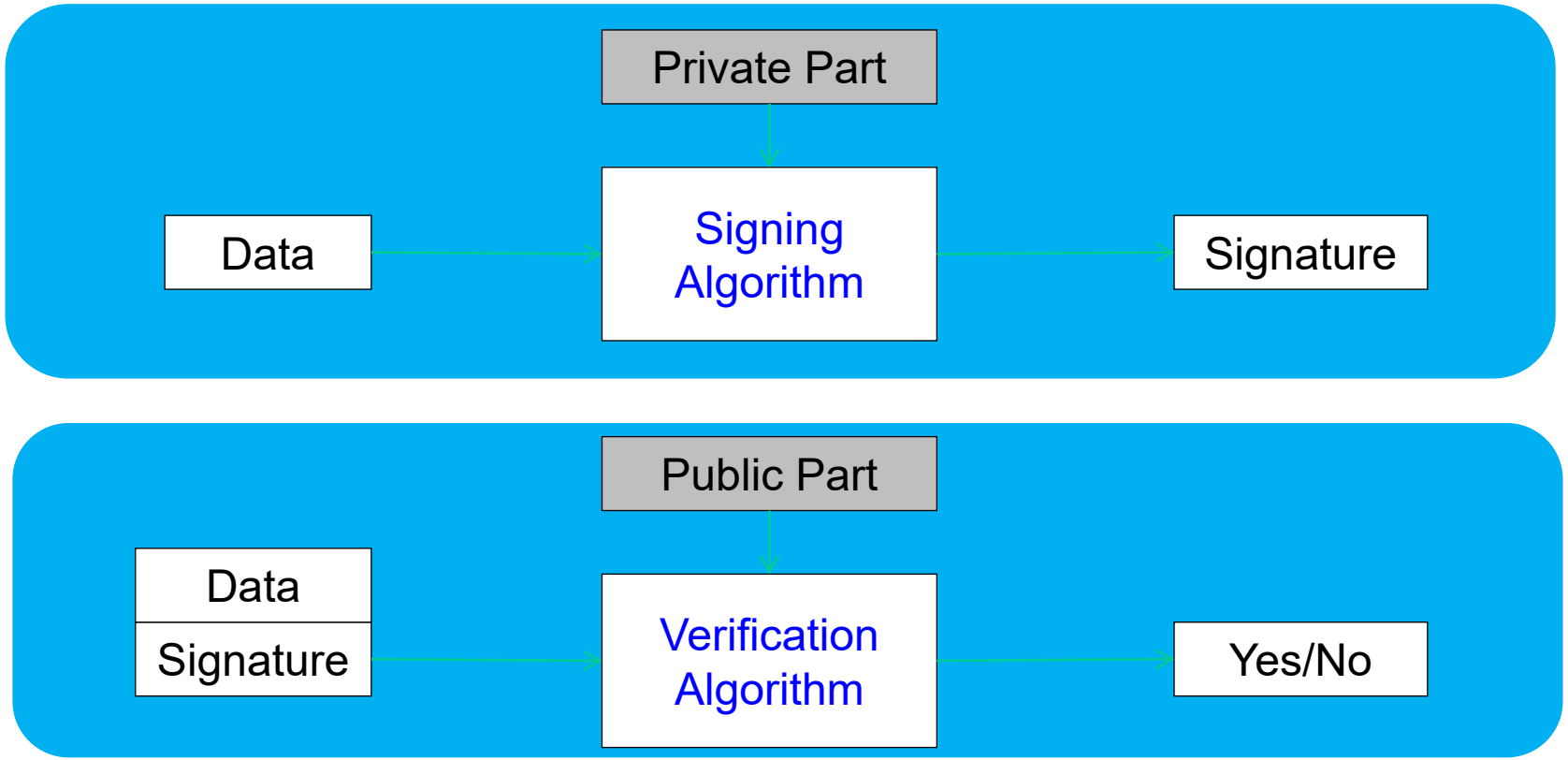
  - Merkle tree "links" pairs of hashes hierachically up to root
    - Any leaf change changes all hashes based on that (incl. the root)

Blockchain Ingredients (3)

- Signatures
  - *E.g.*, RSA

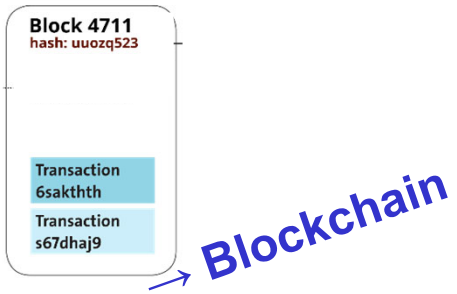
Signing Key	
Public Part	454F4D3E..
Private Part	56F23F2D..

Based on Terence Spies: Blockchain Mechanics



Blockchain Operations

- Transactions (content) collected in blocks
  - New blocks created regularly
- A block contains a hash of and a pointer to the previous block …



- Consensus mechanism required to determine the block to be integrated into this blockchain

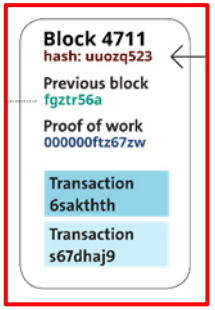


  - Public blocks contain, *e.g.*, solved crypto puzzles (PoW)
    - *E.g.*, a form of partial hash collisions (SHA-256)
- Creation of valid blocks performed by anyone (incentive)
  - Solving crypto puzzles ≡ confirmation of blocks ≡ Mining
  - Computational expensive → Avoids double spending



Blockchain Data Structure in Detail (1)

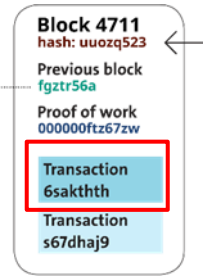
- BCs are a backward-ordered, linear list of blocks
  - Chain starts with genesis block to which others are back-linked
- Blocks contain (at least)
  - Transaction (tx) data (content, payload)
  - Pointer to and a hash of the previous block
  - Cryptographically hashed value of crypto puzzle (result of PoW)
  - Time stamp



- BC's structural and technical characteristics
  - Chain may show side chains, but only one valid branch finally
    - Chronological order guaranteed by previous block's hashes
  - A BC network is organized as a peer-to-peer network
    - Overlay topology may change, replicas of BC are hold on multiple nodes, exchanges of new blocks performed within that overlay, anyone to join

Blockchain Data Structure in Detail (2)

- Transactions (tx/Tx)
  - Data structure encoding the transfer of “value”
    - From a source (“input”) to a destination (“output”)
  - Structures are typically not related to accounts or identities
    - Chunk of “value” locked with a specific secret known by the owner
  - Example tx content
    - Version of tx
    - Input counter – The number of separate input values
    - Inputs – The values
    - Output counter – The number of separate input values
    - Outputs – The values
    - Locktime – Earliest time at which tx will be valid (to be relayed on the BC network or added to the BC), typically a Unix timestamp



UTXO or account

Blockchain Operations in Detail

For illustrative reasons, the Bitcoin BC (BTC) is used here.

- Overall operations consists out of 4 steps (in principle)
  1. User A wants to pay user B the amount of 1 BTC (bitcoin)
     - This intent is broadcast as a transaction via the P2P protocol into the BC network
  2. Any miner interested, adds this transaction to his/her respective current block
  3. Any of these minors wins a lottery and "mines" the block
     - That miner solved the crypto puzzle at first
     - Such a block may contain multiple transactions and is mined in well-defined periods of time
  4. The "definitive block "– including the transactions and the new hash – is broadcast to the BC network again as well and added to everyone's BC copy

Communications – The Flood Protocol

- New transactions and mined blocks are broadcast to the BC network
  - Minimum of 8 neighbors maintained in Bitcoin P2P network
- Temporary BC splits possible
  - 2 miners arrive at 2 different, but valid hashes at the same time
  - P2P network resolves that split in short time to 1 valid branch
  - Clients only accept “longest chain of blocks” as valid (pruning)
    - Length is determined by the most combined difficulty, not #blocks
- As soon as block gets “solved” as valid, every miner in the BC network must stop the current mining process on that block and can start with the next one

Hash Computation and Checks (1)

- Once a set of transactions becomes available, a block is created by utilizing the following data
  - Transaction(s)
  - Hash of previous block
  - Nonce (arbitrary number, used only once)
  - Other information (depending on the BC)
- The hash of new block is calculated meeting the “target”
  - Target hash is a number (absolute value) that a hashed block header must be less than or equal to in order for a new block to be awarded
    - Value used to determine the “difficulty of the input”
    - Value adjusted to ensure that blocks are processed “in time”
- Once the hash was computed, the hash is broadcast to the BC network and checks are performed

Hash Computation and Checks (2)

- Accepted blocks require the miners to solve a crypto puzzle (PoW): hash
  - Difficult to produce (time-consuming, energy), but easy to verify
  - Validity is ensured by checking if a block's hash value is less than "current target"
    - Hash is above the target value → Another miner may have found a suitable hash (not known yet to others), block attached to local BC only, but miner lost the distributed lottery, nonce will be incremented and a retry starts
    - Hash is below the target value → This miner won the distributed lottery and the new block's hash determines the crypto puzzles' result (PoW result ≡ hash)
- Since each block contains the preceding block's hash, a sequence of those determines a larger amount of work
  - Changing any block would require the regeneration of all successors and redoing the work on the data they contain
  - Thus, at least 6 successors are required to consider a block being valid
- The PoW (hash, result of the crypto puzzle) is validated by other members of the BC network by confirming the new hash before adding the block to their local copy, at 51% of the network hashing power' ok, the block is "definitive"

Target Value and Adaptation Algorithm

- Mining is rewarded → Likely more miners join
  - Higher processing capacity increases likelihood on finding hash earlier
- Block creation rate in-, average mining time decreases
  - To maintain the ideal goal of 10 min mining per block: change difficulty, which effects the "target value"
- BC network decreases target value to increase difficulty
  - Decreasing target values increases difficulty to find the hash
- Block creation rate decreases and average mining time increases again
  - System stabilizes itself again, and continue as of above



  – Reverse principle of cycle applies, if block creation rate decreases

Consensus: Hash-based Proof-of-Work (PoW) (1)

Based on Terence Spies: Blockchain Mechanics

- Key: One cannot compute an input from an output
  - To find a hash with *N* zeros at input start, requires 2**N* computations, which proves computational work performed
  - Hashing an incrementing "nonce" as hash input, leads to zeros

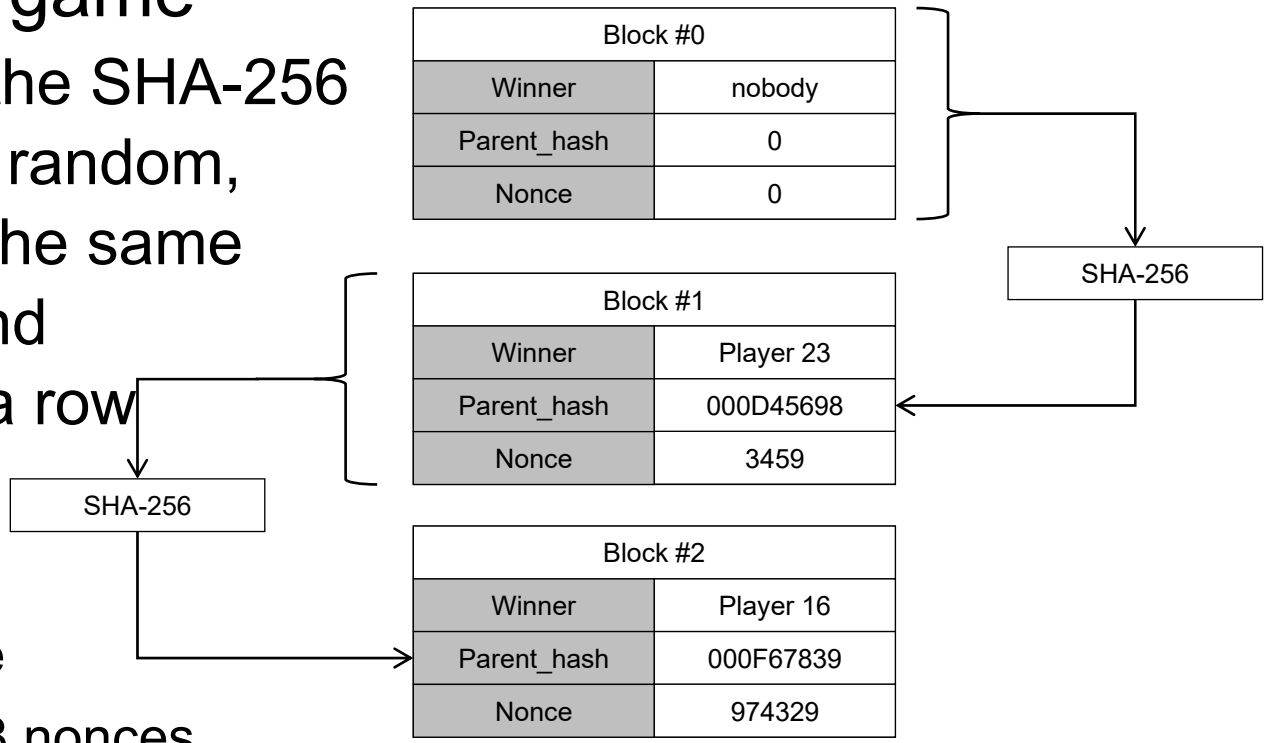
```
in 3e-05 seconds, nonce = 0 yielded 0 zeros.  value = 4c8f1205f49e70248939df9c7b704ace62c2245aba9e81641edf...
in 0.000138 seconds, nonce = 12 yielded 1 zeros.  value = 05017256be77ad2985b36e75e486af325a620a9f29c54...
in 0.000482 seconds, nonce = 112 yielded 2 zeros.  value = 00ae7e0956382f55567d0ed9311cfd41dd2cf5f0a7137...
in 0.014505 seconds, nonce = 3728 yielded 3 zeros.  value = 000b5a6cfc0f076cd81ed3a60682063887cf055e47b...
in 0.595024 seconds, nonce = 181747 yielded 4 zeros.  value = 0000af058b74703b55e27437b89b1ebcc46f45ce55d6....
in 3.491151 seconds, nonce = 1037701 yielded 5 zeros.  value = 00000e55bd0d2027f3024c378e0cc511548c94fbeed0e....
in 32.006105 seconds, nonce = 9913520 yielded 6 zeros.  value = 000000 77a77854ee39dc0dc996dea72dad8852afbde6....
in 590.89462 seconds, nonce = 186867248 yielded 7 zeros.  value = 0000000225060b16117b23dbea9ce6be86ac439d....
in 4686.171007 seconds, nonce = 1424462909 yielded 8 zeros.  value = 00000000 2dd743724609a9f57260e2492908d....
```

- The distributed game sets the difficulty *N* of the game
  - Players get a list of blocks and their content (see slide before)
- Players accumulate "points" by creating blocks
  - Hashing the previous block, finding a hash of the new block with enough zeros, and transmitting this block to everyone

Consensus: Hash-based Proof-of-Work (PoW) (2)

Based on Terence Spies: Blockchain Mechanics

- The "chain race" (probability theory)
- The difficulty of the game
  - For N zeros, since the SHA-256 output is effectively random, getting zero bits is the same as flipping a coin and getting N heads in a row
  - For N zeros, have to try 2*N/2 nonces
    - N=1 …. Try 1 nonce
    - N = 16 … Try 32768 nonces
    - N = 32 … Try 2 billion nonces
  - Winning a block proves the player did work!



Smart Contracts (1)

- A Smart Contract (SC) resides inside transactions
  - Executed and validated on every node
  - In Bitcoin (a blockchain-based cryptocurrency), SCs specify how to withdraw, escrow, refund, or transfer BTC from A to B
- Smart contracts first mentioned in 1994/96

> A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives of [a] smart contract['s] design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs. [N. Szabo]
>
> http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html
> http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

- Smart contracts alone are not "smart"
  - They need an infrastructure
  - A blockchain is the ideal infrastructure

Smart Contracts (2)

- SCs can be exchanged (not only transactions)
  - This is the code or program to be executed
    - "Programs and APIs" on the BC
  - Thus, SC are programs that encode conditions and outcomes

- *E.g.*, SCs can define secure escrow services
  - Ready to use program working on pre-determined conditions between any two users (*e.g.*, a provider and a customer)
  - "Real-time" operations
  - Near zero marginal cost

- Note: Smart Contracts are neither "smart" nor "legal"!

Blockchain Operations II

Unspent Transaction Output (UTXO)

- Concept based on physical money, *e.g.*, bills
  - Balance is calculated by total of unspent outputs
  - Each transaction has inputs (unspent outputs) and outputs

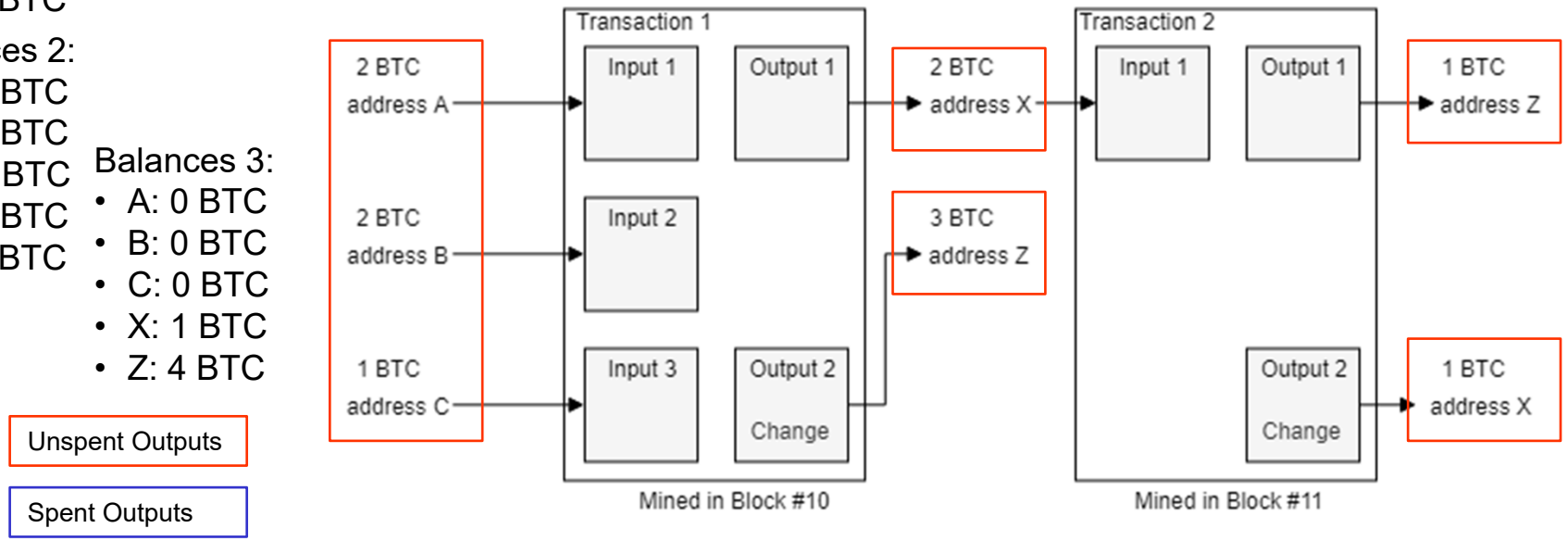


Balances 1:
- A: 2 BTC
- B: 2 BTC
- C: 1 BTC

Balances 2:
- A: 0 BTC
- B: 0 BTC
- C: 0 BTC
- X: 2 BTC
- Z: 3 BTC

Balances 3:
- A: 0 BTC
- B: 0 BTC
- C: 0 BTC
- X: 1 BTC
- Z: 4 BTC

- Possible to combine one or more outputs as inputs to form a transaction



Account-based Model

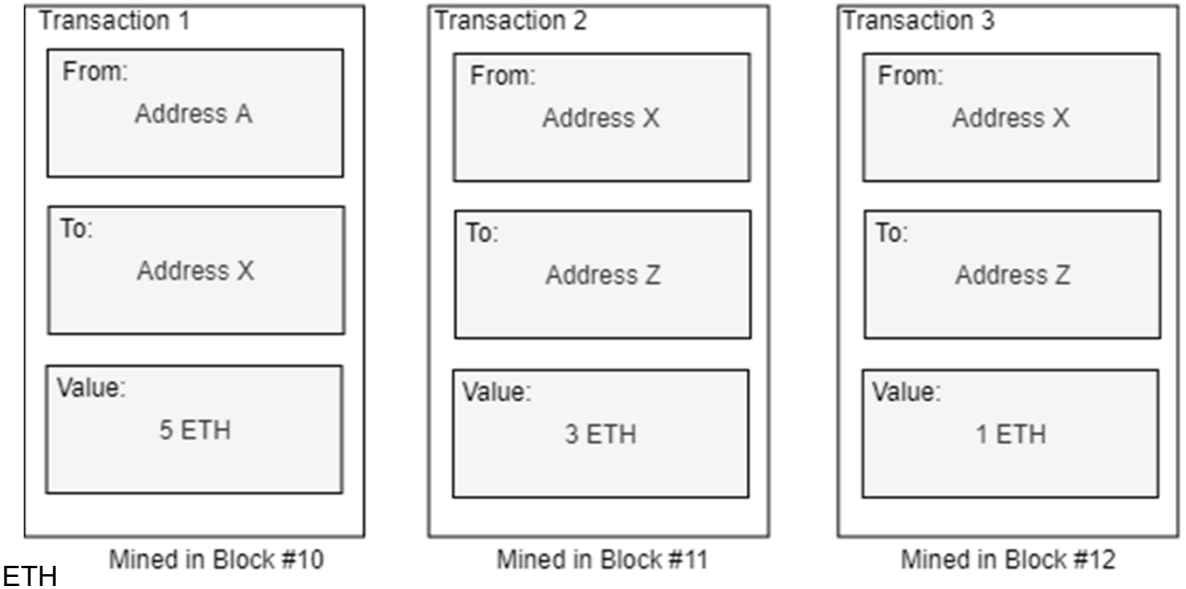
- Account maintained by a global state
- Similar concept to a debit card
  - Each transaction modifies the state of accounts
    - Balance should be larger or equal than spending amount



Balances 1:
- Account A: 0 ETH
- Account X: 5 ETH

Balances 2:
- Account A: 0 ETH
- Account X: 2 ETH
- Account Z: 3 ETH

Balances 3:
- Account A: 0 ETH
- Account X: 1 ETH
- Account Z: 4 ETH



Account A's initial balance: 5 ETH

Transaction (tx/Tx) Fees

- Mining and consensus finding require resources
  - Specifics depend on the dedicated approach used
- Generally, tx fees are part of the tx
  - Compensation for miner to mine
    - Tx fees are collected by that miner
    - Tx fees are set by the BTC's market forces
  - Tx fees incentivize a miner to include a tx into a block
    - High fees: early selection; low fees: delayed selection
      - No requirement for tx fees, but mining may be delayed "forever"
  - Tx fees are countermeasures against "spam" as providing no tx fees inside a tx may not lead to its selection
  - Tx fees are typically calculated by the size of the tx (in Byte)
    - A calculation of the tx fees based on the tx values is not applied

Blockchain Transaction Types

- On-chain tx
  - Available on the blockchain visible to all nodes on the BC
  - Tx valid when a communicated that tx across the network
    - Tx times may vary depending on the network load or tx queued
- Off-chain tx
  - Value resides outside of the blockchain
  - Tx executed instantly
- On-chain tx with off-chain storage of data
  - The same is valid as for on-chain tx
  - Additionally, the tx does not contain the full data, but only a hash of the data, which is stored off-chain
    - Reduces storage size of BC for many tx or large volume

Blockchain Addresses (1)

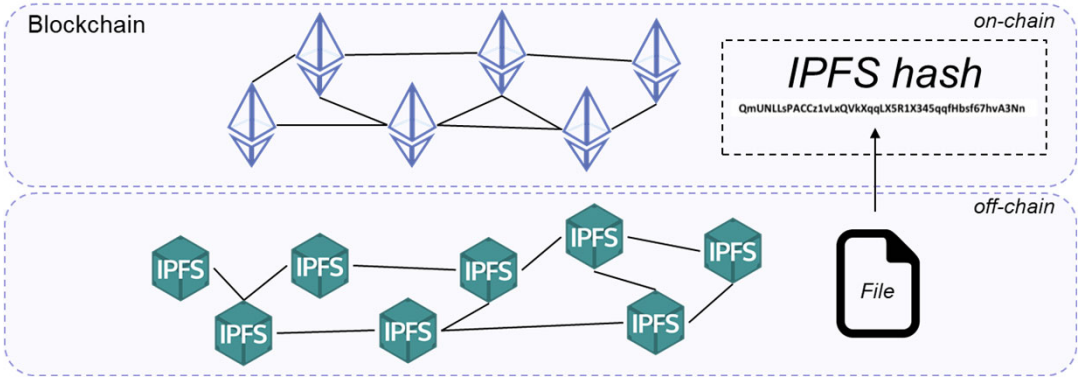
- Transactions on a blockchain require the knowledge of the blockchain address of the sender and recipient
  - Address as 27-34 digit code consists of letters and numbers
    - Generated by the wallet, where public-private key pairs are stored, from public key via cryptographic hash function (“finger print”)
    - Usually currency-specific, addresses belong to a certain blockchain
  - Example Bitcoin
    - A = RIPEMED160(SHA256(K))
      with A Bitcoin address and K public key
    - Results in a 160 bit (20 Byte) number
  - Note: A public key **IS NOT** the same as a Bitcoin address, but this is derived from a key with a one-way hash function!

Blockchain Addresses (2)

- Address representations
  - Base58 encoding (subset of Base64)
    - Text-based binary-encoding format, using upper- and lowercase letters, but omitting “0” and “O”, “l” and “I”, and “\”, “/”, and “+”
    - Balancing compact representation, readability, and error detection
  - Base58Check encoding
    - Adds 4 Byte checksum (error-checking over encoded data) to the add
      - Used for BTC, prevents transcription and typing errors
    - Example:
      - Add a *prefix* to *data* (version byte), $0x00_h$ in the Bitcoin address case $0x80_h$ in case of a private key to be encoded
      - Compute double SHA checksum (*CS*):
        CS = SHA256(SHA256(*prefix*+*data*)), with “+” concatenation,
        and use of 4 first bytes of this 32 Byte hash

Off-chain Signaling of Addresses

- Problem: BCs typically show limited storage capacity
- Solution
  - Digital representation of a file (“hash reference” != “PoW result”) is stored on-chain and the file itself is stored off-chain
- Practice
  - Peer-to-peer network stores and shares hypermedia (*e.g.*, graphics, audio, video, plain text) in a distributed file system



IPFS: Inter-Planetary File System
https://ipfs.io/

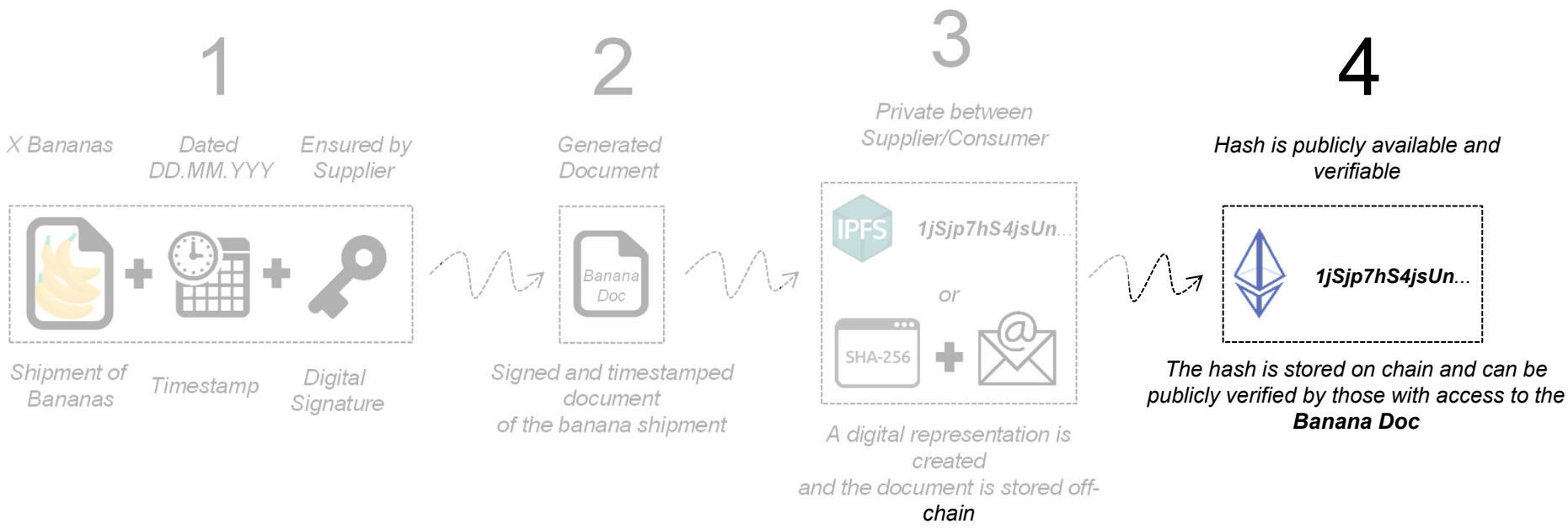
Wallets

- Container for private keys
  - Structured files, databases
  - Wallets contain keys (as keychains of private/public keypairs) not coins (which are stored on the BC as tx outputs), thus, keys prove the ownership of a coin!
- Wallet types
  - Non-deterministic (random): just a collection (hard to maintain)
  - Deterministic (seeded): keys derived from a common seed (random number+index) via hash-functions
    - Hierarchical deterministic (HD) – BIP0032/BIP0044 standard – with keys in a tree structure: parent ⇨ children ⇨ grandchildren
    - Structure organizes inputs, branches with categories (*e.g.*, departments)
  - Paper: character strings, 2D codes, BIP0038 encrypted print

Blockchain Functionality and Terminology

Blockchain‘s Immutability Exploited

- Time stamping proves existence, integrity, and creation/exchange of digital assets
  - Keeping track of the creation and modification time





Supplier can use a blockchain-based time stamping solution to ensure the quality of his products

© 2022 UZH, CSG@IfI

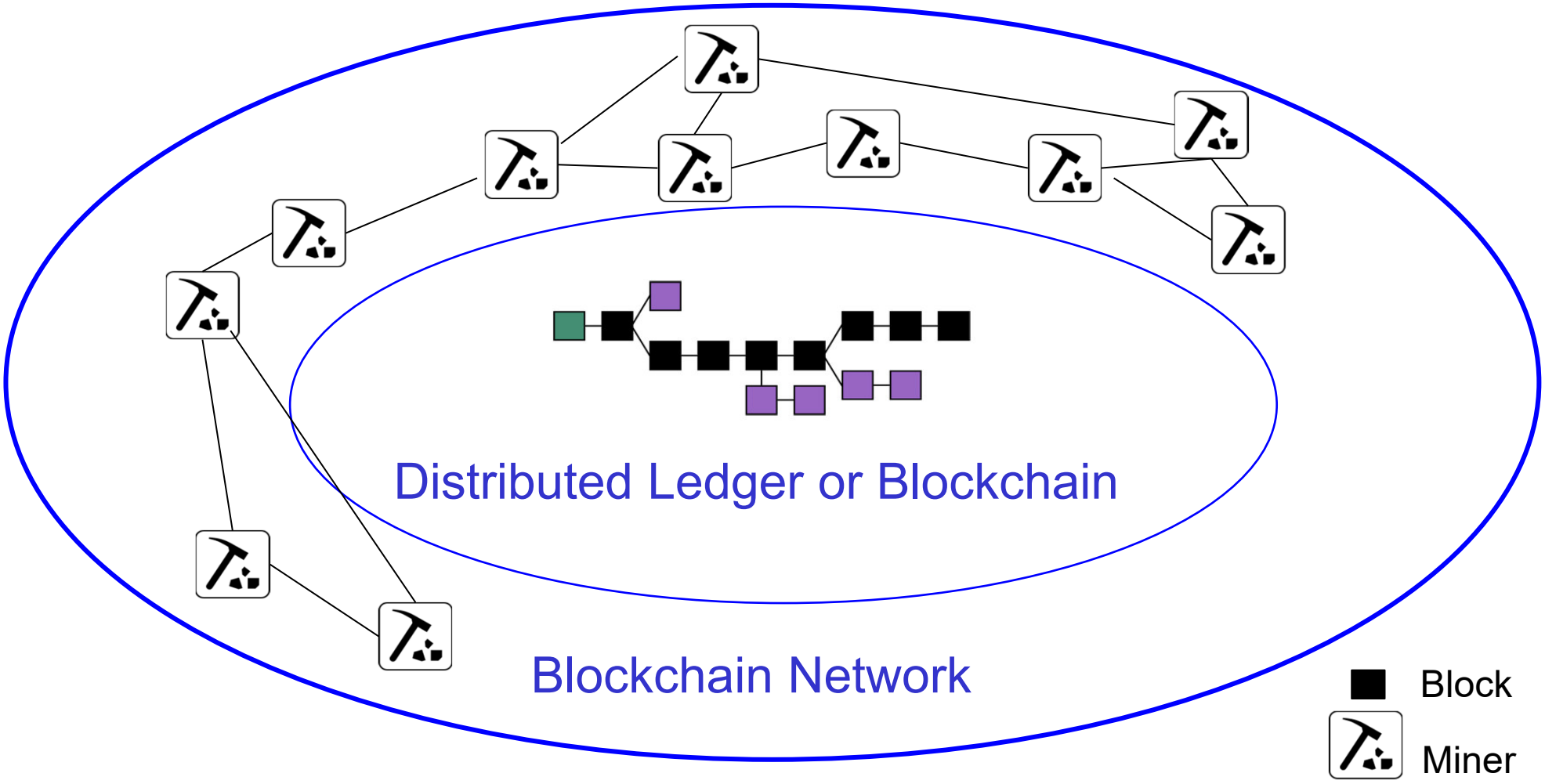
Segregated Witness (SegWit)

- BC protocol “upgrade” to provide interception protection on transaction malleability and increase block capacity

  Transaction malleability is an attack that lets a person change a Bitcoin transaction's unique ID before confirmation on the Bitcoin network. This change makes it possible for the person to pretend that a transaction didn't happen.

  - SegWit separates *witness* from the list of inputs
  - Witness contains data required to check a transaction validity, but is not required to determine tx effects https://en.bitcoin.it/wiki/Segregated_Witness
  - *weight* parameter defined
  - Blocks are allowed to have at most 4 million weight units (WU)
    - Non-witness and pre-segwit witness bytes weigh 4 WU, but each byte of Segwit witness data only weighs 1 WU, allowing blocks that are larger than 1 MB without a hardforking change
- SegWit was the protocol change needed to make the Lightning Network safe to deploy on the Bitcoin network

Blockchain Terminology (1)



Chain: Backward-linked list of cryptography-based (hash-secured) pointers to previous blocks

Blockchain Terminology (2)

- Mining (Process)
  - The process of BC members trying to solve the crypto puzzle and adding the respective new block onto the BC
- Miners
  - Those BC members, who run machines to solve crypto puzzles
  - Their reward in case of a successful inclusion are tokens of BC
    - In case of the bitcoin BC the reward are BTC (incentives to participate)
- Checks
  - Verification of hashes broadcast to the BC network

Blockchain Terminology (3)

- Difficulty
  - Measure of how difficult it is to find a new block compared to the easiest it can ever be
  - Recalculated every 2016 blocks in the BTC BC such that the last 2016 blocks would have been calculated within two weeks
    - On average, one block is mined every 10 min
- Consensus
  - State reached where the majority of members of the same P2P network agrees on the same mining output (51% agree)
  - This state of the consensus is secure and tamper-resistant, immutable with respect to the blocks
  - Respective block data is persisted to the BC network's nodes

Blockchain Eras, Application Domains, and Consensus Mechanisms



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

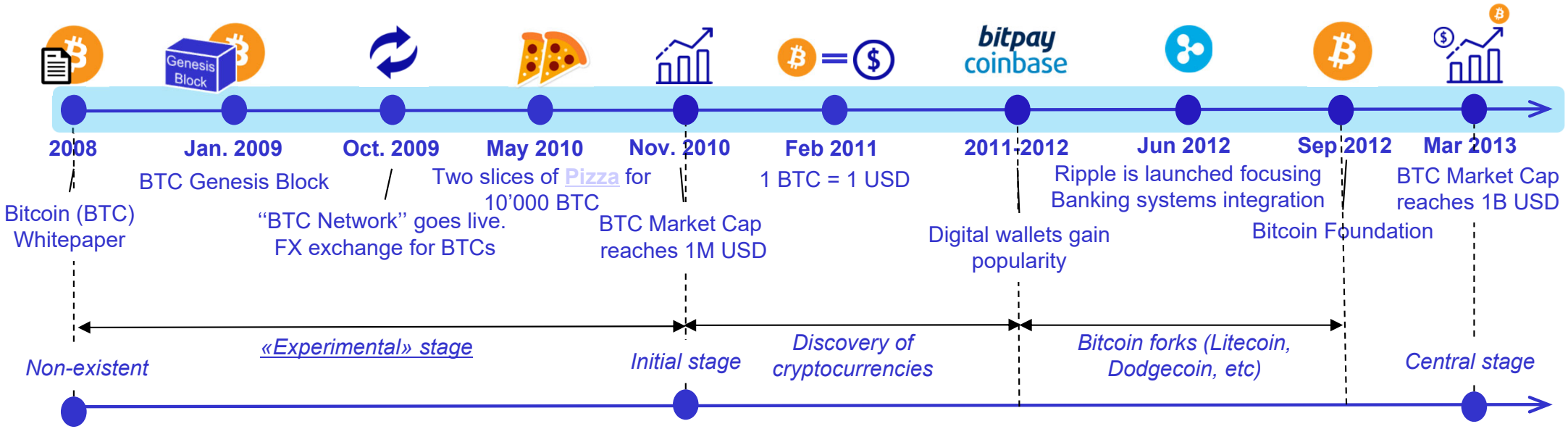
- Blockchain Eras
  - 1.0 to 4.0
  - Evolution
- Blockchain Application Domains
  - Domains and Characteristics
  - DeFi (Decentralized Finance)
  - Effects
- Consensus Mechanisms
  - Distributed Agreement
  - Mechanism Types' Overview
  - BFT, dBFT, PoW, PoS, PoA
  - Alternatives, Hybrids, Comparison

© 2022 UZH, CSG@IfI

Blockchain Eras

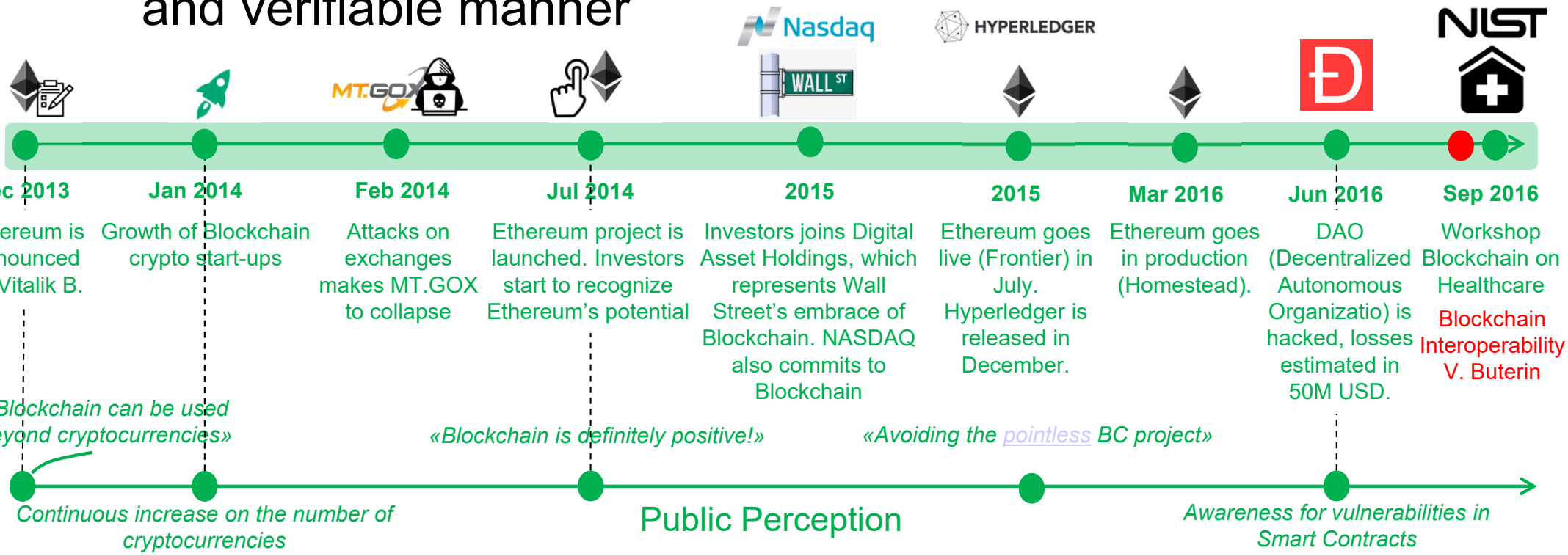
Blockchain 1.0

- Digital Currency or Cryptocurrencies
  - Decentralized payment system
  - Bitcoin as the father of digital currencies
    - Still, not much awareness of (other) blockchain capabilities
  - Proof-of-Work (PoW)



Blockchain 2.0

- Smart Contracts
  - Ethereum unlocks the blockchain potential beyond cryptocurrencies
  - Blockchain is able to run computer programs in a transparent and verifiable manner

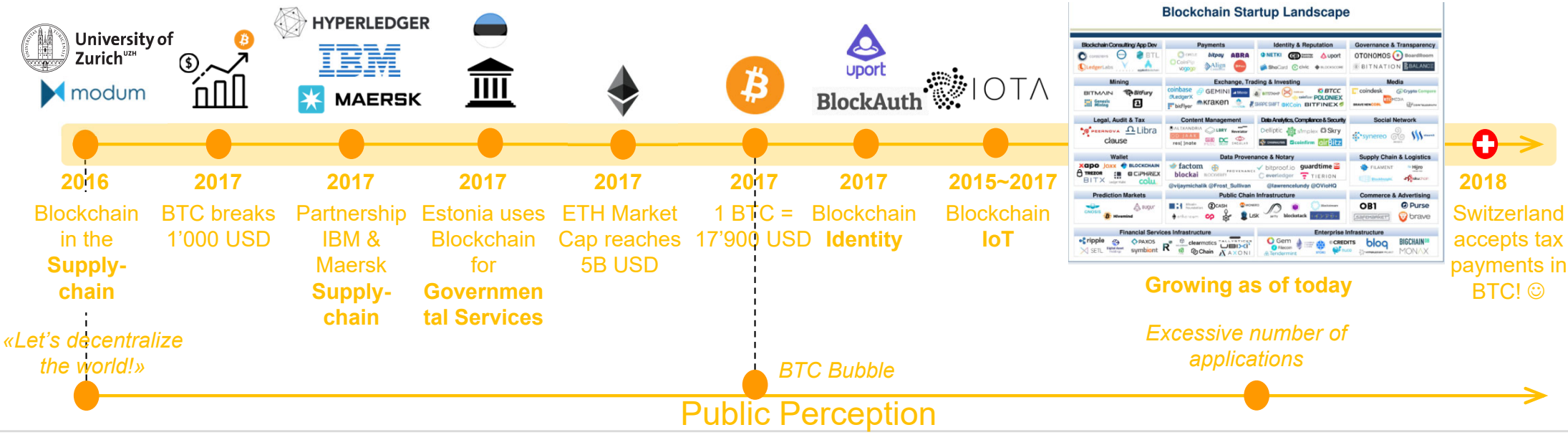


© 2022 UZH, CSG@IfI

Blockchain 3.0

- Decentralized Applications (DApps)
  - Production stage:
    - Large number of applications
  - Scalability/Performance issues:
    - Need for performance → new consensus protocols
    - Need for storage → off-chain storage tools





Blockchain 4.0

- Ecosystem and Industry Integration
  - Making blockchain effective in industry
  - Decentralized and disconnected blockchain networks
    - Vendor-specific blockchain technology, interoperable chains
  - Need for standardization

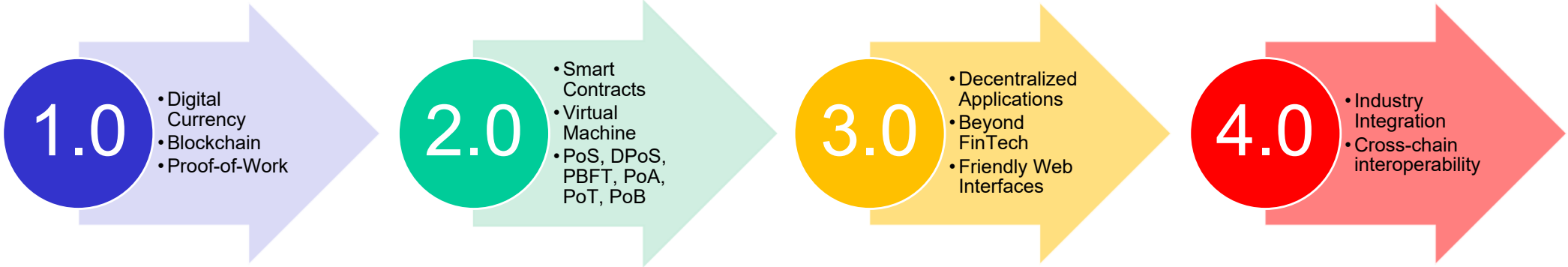
As of today



origintrail
wanchain
COSMOS
INTERNET OF BLOCKCHAINS
ARK.io
Ambrosus
BLOCKONOMI
aelf.
HERDIUS
Seele
blockstack
AION
The Third Generation Blockchain Network
INTERLEDGER
Payments across payment networks
MULTIVERSUM
HERE TO STAY
Polkadot.
CROWD MACHINE

Blockchain Eras and Evolution

- 4 different BC eras are running in parallel today



  - 1.0 – December 08/January 09: Bitcoins
    - More than 17.390+ cryptocurrencies available today



  - 2.0 – 2012-14: Ethereum, Smart Contracts, Solidity, …
  - 3.0 – April 2012: Decentralized Apps (dApps) – "Satoshi Dice"

https://hackernoon.com/dapp-and-things-you-need-to-know-4f50853a4cb7

    - Running on peer-to-peer network, all data transparent and tamper-proof
  - 4.0 – App. 2015: BC ecosystems and industrial integration
    - Countless Blockchain projects in many fields
      - FinTech ⇨ DeFi, supply-chain, governmental, identity, …

DeFi: Decentralized Finance

Blockchain Application Domains

Number of BC Application Domains

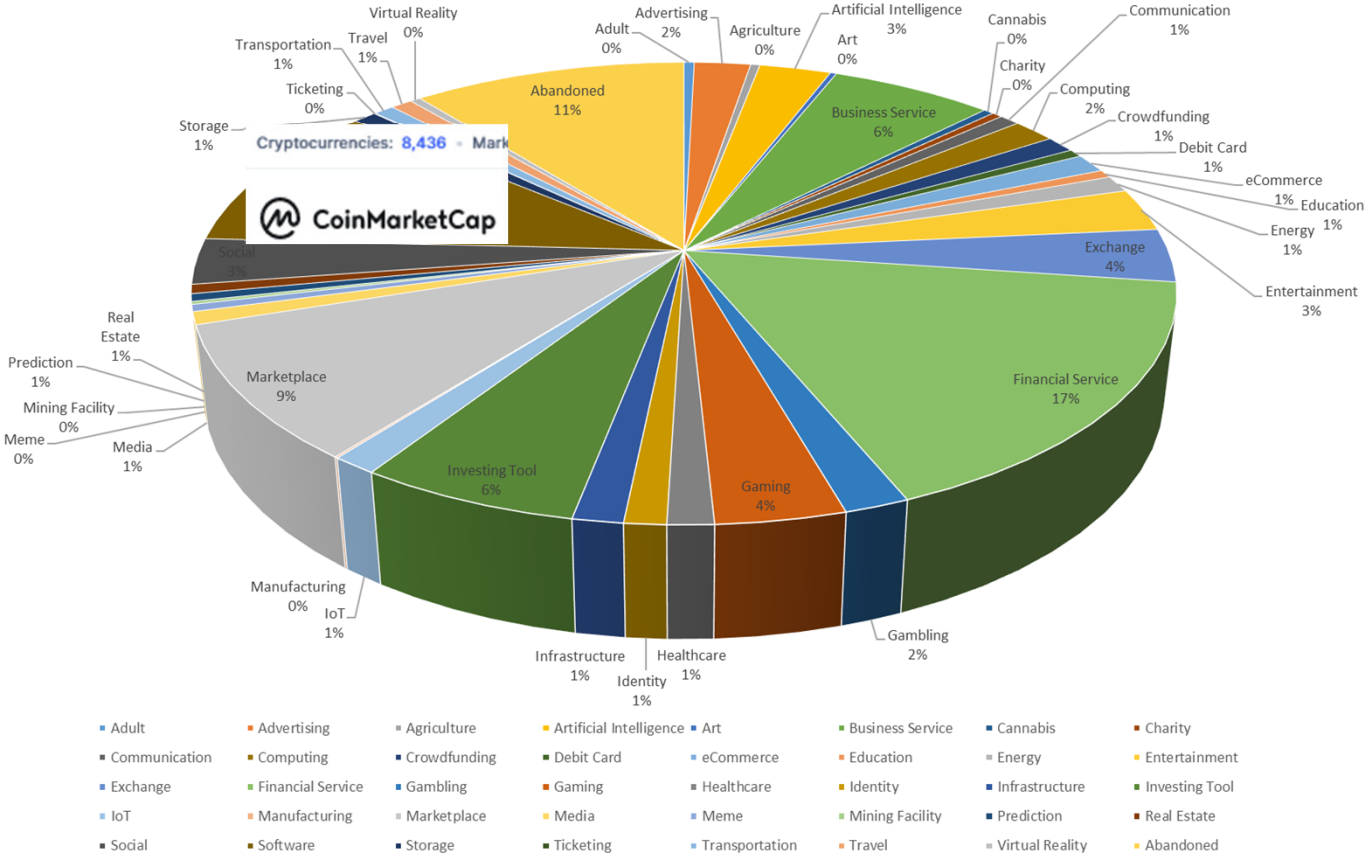
- Based on that time 8,430+ cryptocurrencies in 2020
  - Not all cryptocurrencies are implemented on its own Blockchain

Cryptocurrencies: 8,436 · Mark

CoinMarketCap

https://coinmarketcap.com/

- Finance
  - 17%
- Abandoned
  - 11%
- Software
  - 10%
- Marketplaces
  - 9%

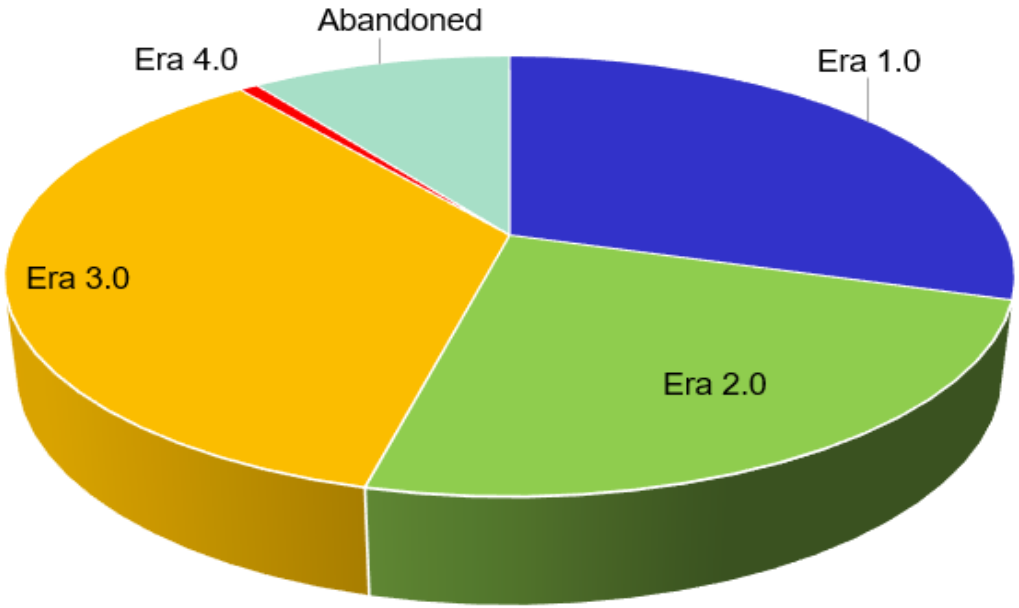


© 2022 UZH, CSG@IfI

Application Domains Grouped by BC Eras

- Similarly, BC projects/domains grouped in Eras
  - Cryptocurrencies and digital finance sector is still dominant
  - However, dApps represents the major number of projects

- Era 1.0 (Finance)
  - 738 projects, 30%
- Era 2.0 (Smart Contracts)
  - 602 projects, 24%
- Era 3.0 (dApps)
  - 884 projects, 35%
- Era 4.0 (Integration)
  - 18 projects, 1%
- Abandoned
  - 248 projects, **10%**



Missing more integration!!

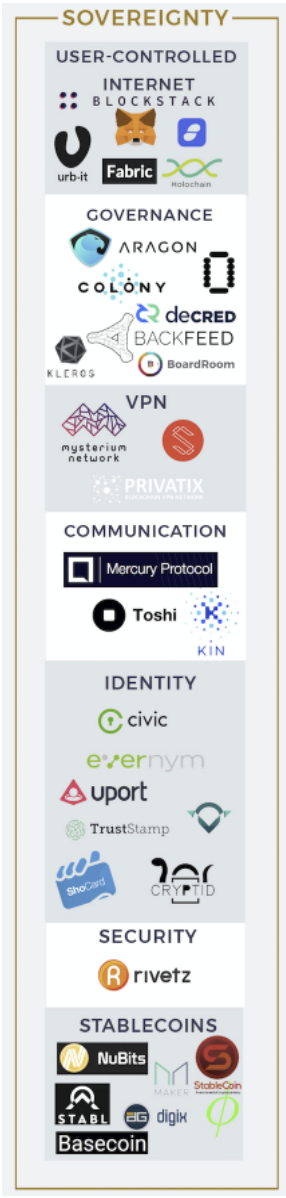
© 2022 UZH, CSG@IfI

Application Domains (1)

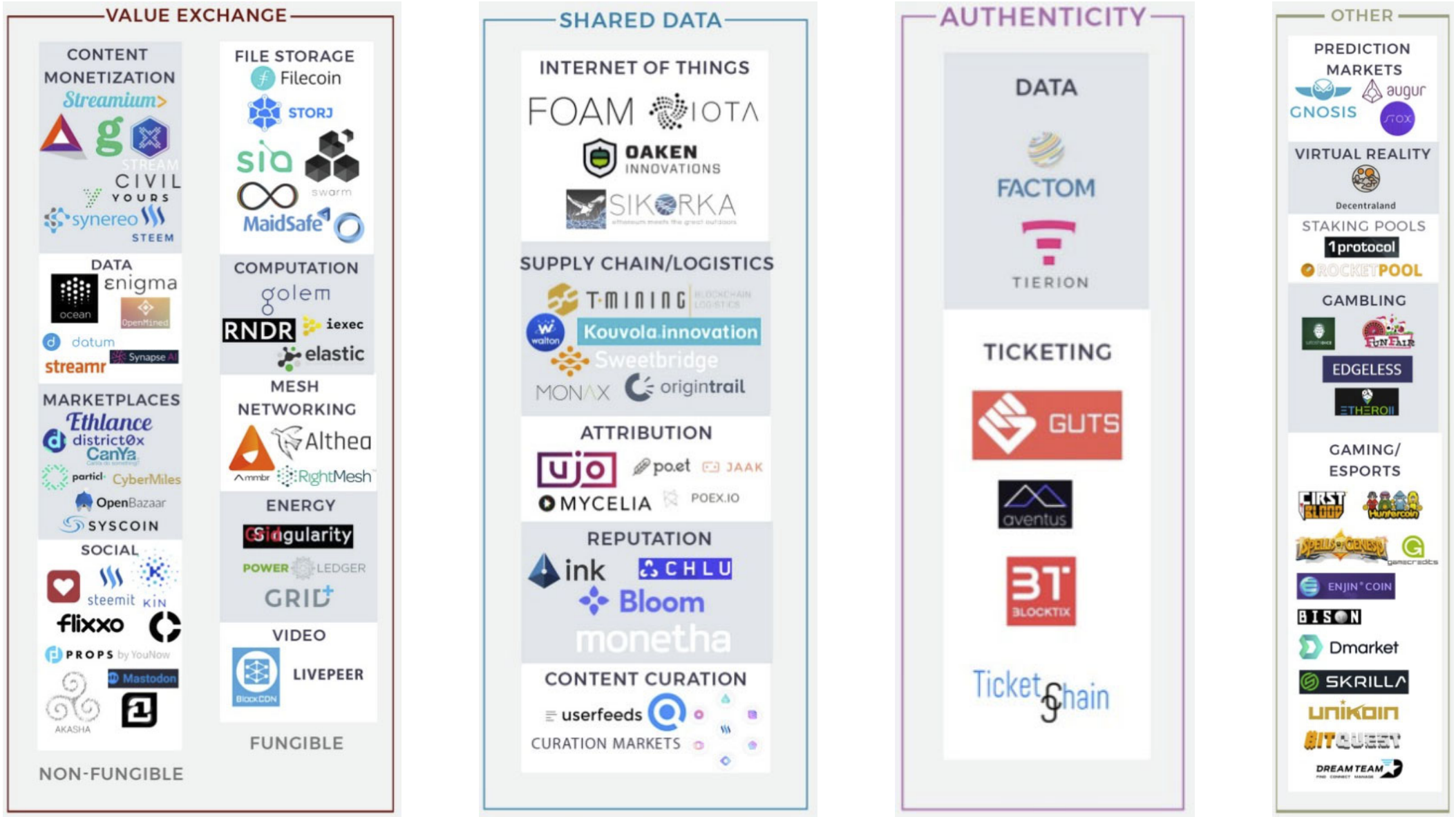








Application Domains (2)



Application Domain Characteristics

Application Domain	Blockchain (BC) Era	BC or Distributed Ledger (DL)	Read (R) and Write (W) Permissions	Mining Permissions
Currencies	1.0	BC	Public R/W	All nodes
Developer Tools	2.0	BC	Public R/W	All nodes
FinTech	3.0	DL	Private R/W	Selected nodes
Sovereignty	3.0	DL	Private R/W	Selected nodes
Value Exchange	3.0	DL	Public R, Private W	Selected nodes
Shared Data	3.0	DL	Public R, Private W	Selected nodes
Authenticity	3.0	DL	Public R, Private W	Selected nodes
Networking	3.0	BC	Private R/W	Selected nodes
Interoperability	4.0	BC	Public R/W	Selected nodes

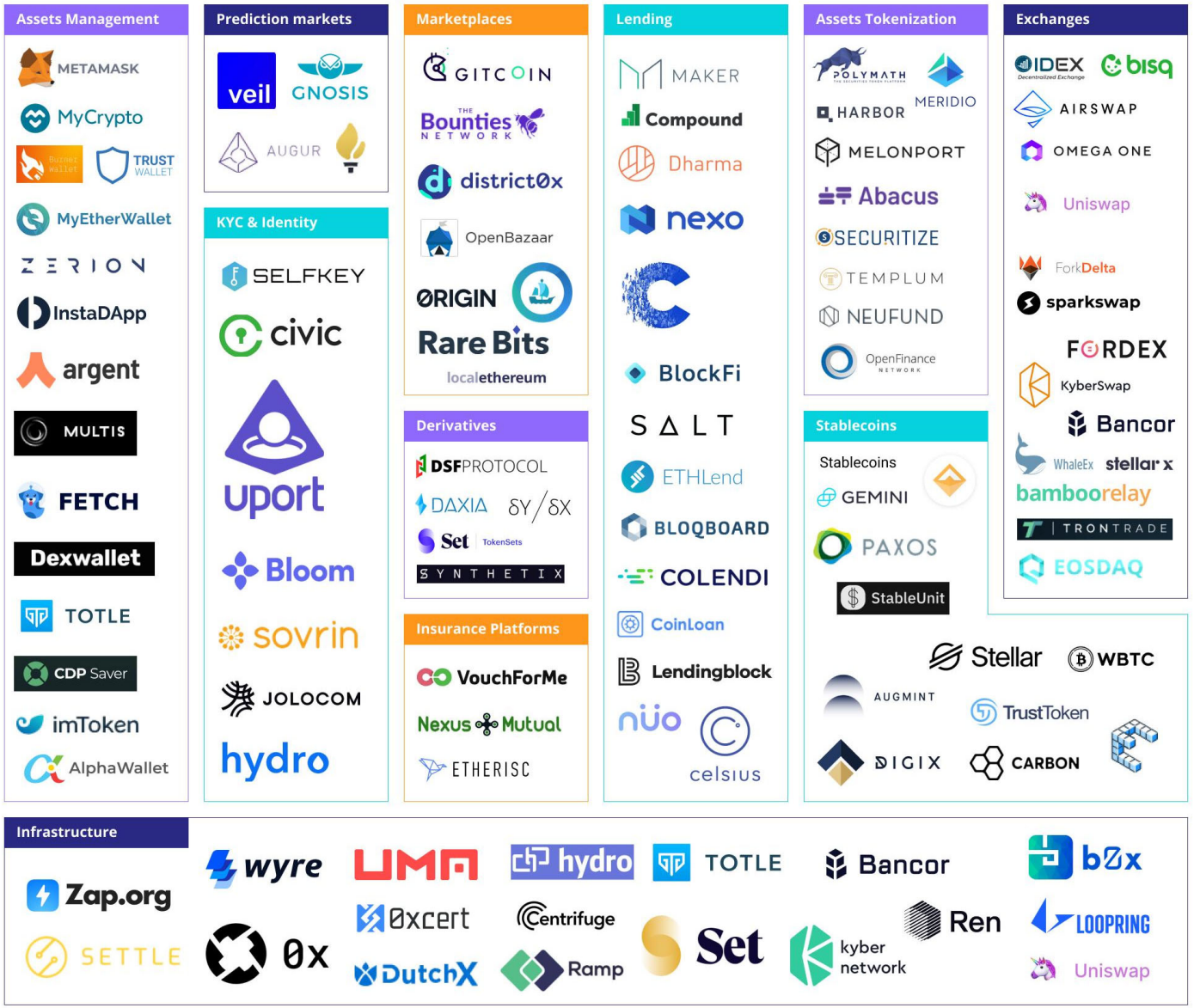
*Based on a general observation, characteristics of specific projects within each application domain may vary

© 2022 UZH, CSG@IfI

Decentralized Finance (DeFi)

- Definition:
  - *"Set of applications and companies offering financial services based on blockchain-based smart contracts". (Bitcoinsuisse)*
  - *"…An umbrella term for a variety of financial applications in cryptocurrency or blockchain geared toward disrupting financial intermediaries". (Coindesk)*
- Major Applications
  - Decentralized Exchanges (DEXs): direct trade cryptocurrencies
  - Stablecoins: cryptocurrency pegged to other assets
  - Lending platforms: direct crypto lending
  - Prediction markets: betting on events

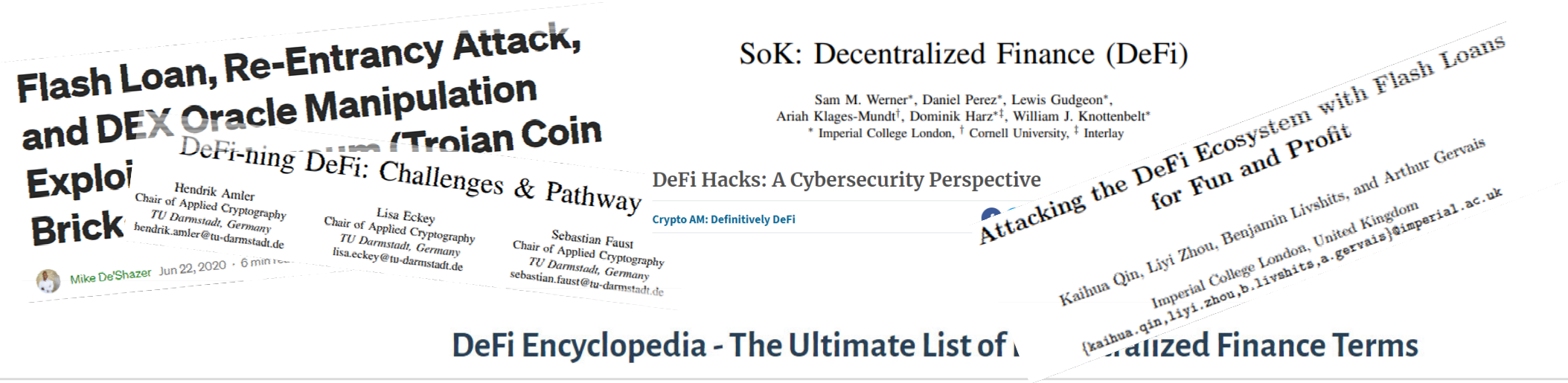
DeFi – Broad Range of Use Cases



Source: defiprime.com

DeFi and SC Security Aspects

- DeFi applications typically represents more risk to investors
  - DeFi inherits traditional Smart Contract (SC) errors, bugs, and vulnerabilities (and intentional scams)
  - Introduction of new vectors, such as flash loans
    - Uncollateralized lending without warranty that must be repaid in the same block, otherwise the transaction is cancelled



Effects of Blockchain Eras

- Different BC Eras present different requirements
  - Performance: transactions per second, latency
  - Reliability: number of nodes
  - Security: trade-offs between confidentiality and transparency



The consensus mechanism is the main responsible to ensure these aspects!

- As a consequence, different consensus mechanisms were necessary with the BC and DL development
  - Since traditional Byzantine Fault Tolerance (BFT) could not deliver all demands, election-based ones (PoW, PoS) appeared

Consensus Mechanisms

Reminder: Permissionless and Permissioned Blockchains

Characteristic	Permissionless	Permissioned
Governance	Public	Private/consortium
Participation	Free join and leave	Authorized
Node identity	Pseudonymous	Revealed
Transparency	Open	Closed/open
Network size	Large (thousands or more)	Small (tens to hundreds)
Network connectivity	Low	High (often fully connected)
Network synchrony	Asynchronous / partially synchronous	Partially synchronous / synchronous
Transaction capacity	Low (often sub-ten to tens)	High (often thousands)
Application examples	Cryptocurrency, public records	Inter-bank clearing, business, supply-chain
Consensus examples	**PoW, PoS**	**PoA, PBFT**

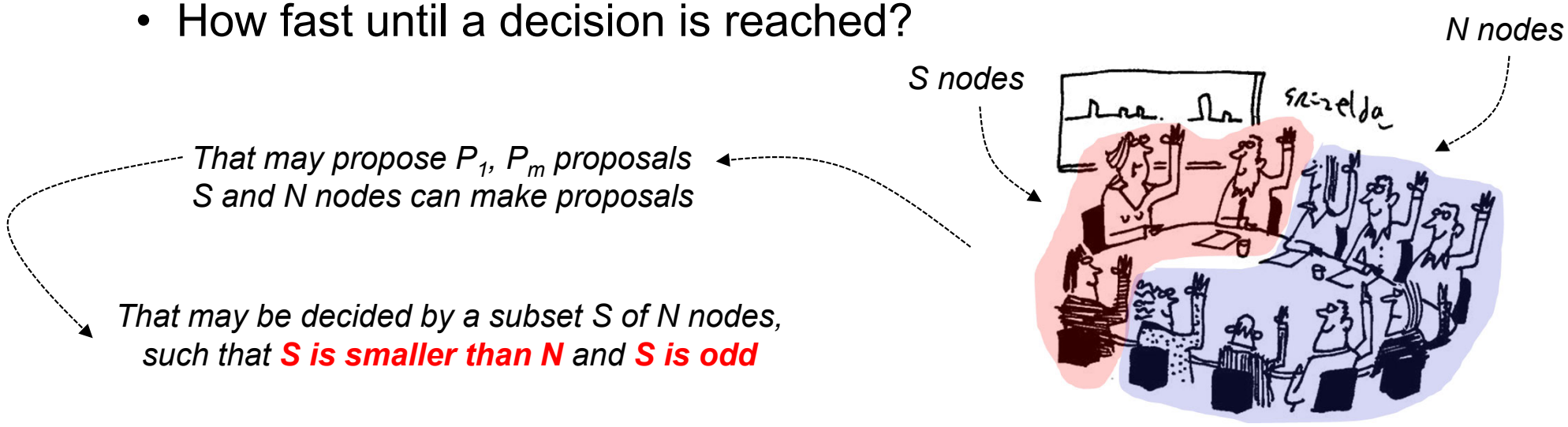
© 2022 UZH, CSG@IfI

Mechanisms for Distributed Agreement

- Also called "Distributed Consensus" algorithms
- The 4 key characteristics https://pradeeploganathan.com/blockchain/consensus/
  - Uniform agreement: No two nodes decide differently
  - Integrity: No node decides twice
  - Validity: If a node decides on value *v*, then *v* was proposed by some node
  - Termination: Every node that does not crash eventually decides on some value
- Given a cluster of *N* nodes and a set of proposals P_1 to P_m, every non-failing node will eventually decide on a single proposal P_x without the possibility to revoke that decision. All non-failing nodes will decide on the same P_x.

Mechanisms for Distributed Agreement

- How to decide on a single consensus proposal P_x?
- There are many aspects a consensus mechanism has to address:
  - How many of the non-failing nodes are entitled to vote?
    - How to decide who is entitled to vote?
  - How to ensure a decision in a deterministic manner?
    - How fast until a decision is reached?



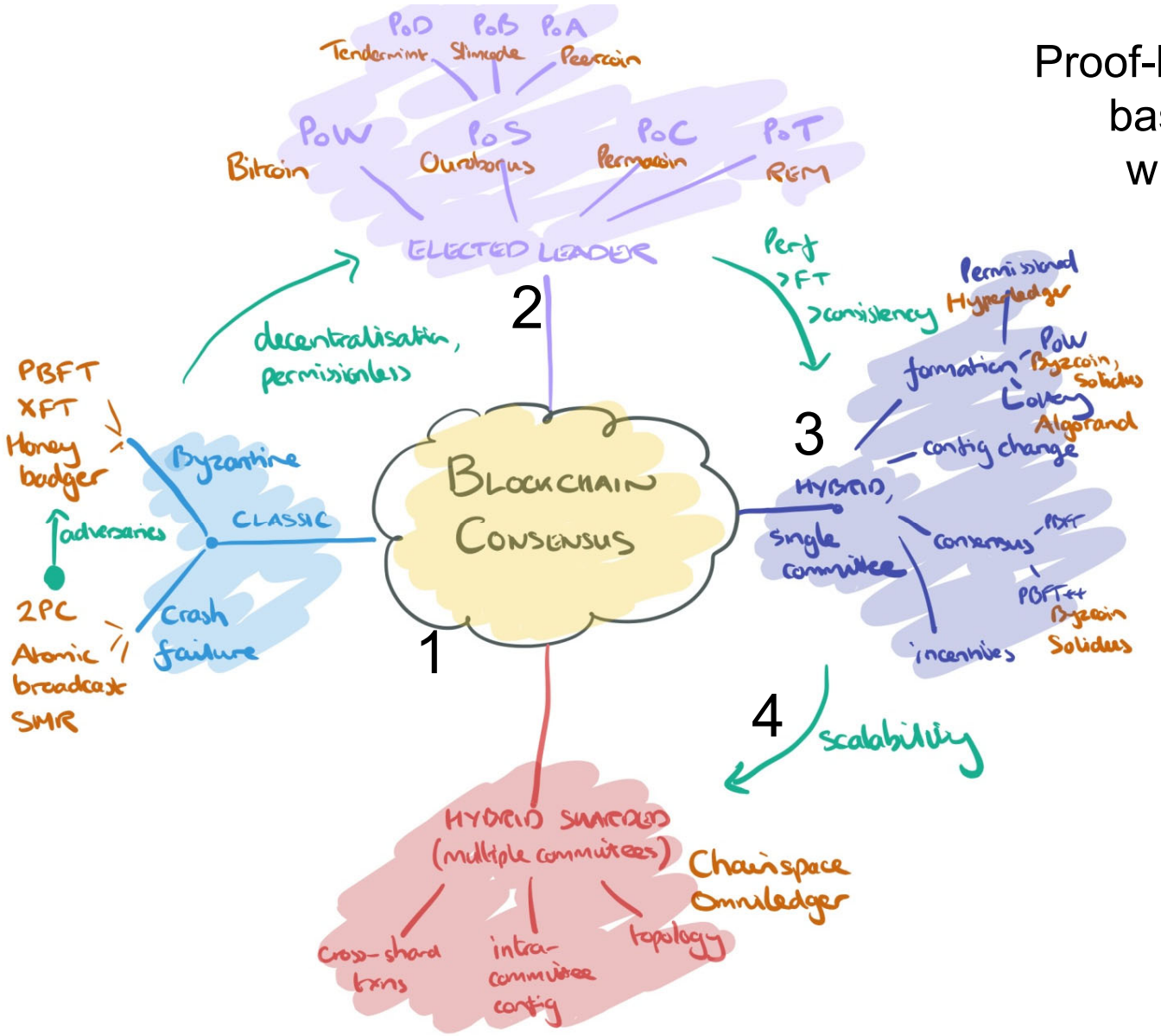
Generic Components of a Blockchain Consensus Protocol

Component	Purpose	Counterpart in Traditional Consensus Protocols	Available Options (Examples)
Block Proposal	Generating blocks and attaching essential generation proofs (for Sybil attack resistance)	Clients issuing operation requests and the primary server starting the consensus	Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Authority (PoA), Proof-of-Retrievability (PoR), Proof-of-Elapsed Time (PoET), Round Robin, Committee-based
Information Propagation	Disseminating blocks and transactions across the network	Reliable broadcast of operation requests	Advertisement-based gossiping, block header soliciting, unsolicited block push (broadcast), relay network (for mining pools).
Block Validation	Checking blocks for generation proofs and validity of enclosed transactions	Signature check and execution of operation requests	Proof checking (for proof-of-X block proposal), digital signature & eligibility checking (for committee-based block proposal)
Block Finalization	Reaching agreement on the acceptance of validated blocks	Servers reaching an agreement on current state, executing requests and logging the result	Longest-chain rule, GHOST rule, BFT and other Byzantine agreements, check-pointing
Incentive Mechanism	Promoting honest participation and creating network tokens	N/A	Network token rewards (block rewards, transaction fees), eligibility for issuing new transactions

© 2022 UZH, CSG@IfI

Evolution of Consensus Mechanisms

Byzantine fault tolerance-based, which is a more traditional approach based on rounds of votes.



Proof-based or leader-based consensus, whereby a leader is elected and proposes a final value

Figure

Consensus Mechanism Types (1)

- Classical Consensus Models
  - Crash failure models → honest nodes failing
  - Byzantine Failure Tolerance (BFT)
    - State machine replication
  - BFT General's Problem

HyperLedger (SOLO, Kafka mechanisms), Stellar



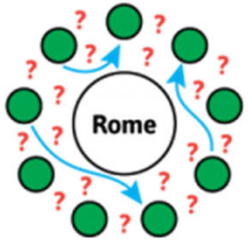
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.



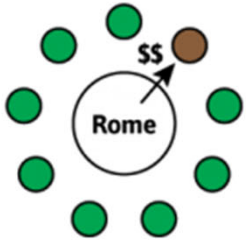
In order to launch a succesful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.



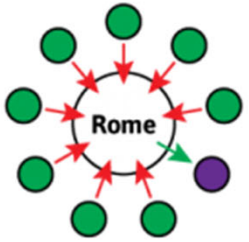
The decision to either **attack** or retreat is put up to a **vote**. Whichever option receives **more than 50%** of the votes, that's what the Generals will do (**retreat** in the example above).



Problem 1
The generals communicate by using **couriers**, who have to cross unknown areas controlled by the Romans, risking capture or their message becoming corrupt.



Problem 2
Each of the generals could be bribed by the Romans: **Traitorous Generals**.



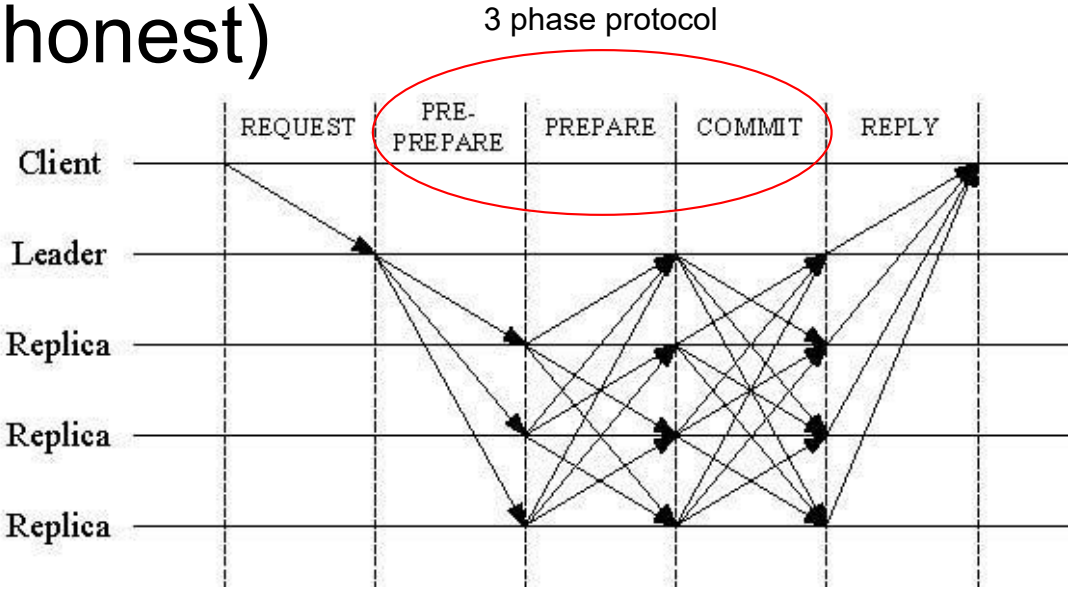
Problem 3
Any of the Generals can make the wrong decision, regardless of the vote: **Improperly Functioning Generals**.

CryptoGraphics.info

Byzantine Fault Tolerance (BFT)

- Described as the capacity of a system to handle or survive unreliable situations and (all kinds of) failures
- Practical BFT (PBFT): assume a small fraction of nodes as Byzantines (dishonest)

1. A client sends a request to invoke a service
2. The primary leader multicasts the requests to the replicas
3. Replicas execute the request and send a reply to the client
4. The client wats for F+1 replies from different replicas with the same result



n = Total # of nodes in network

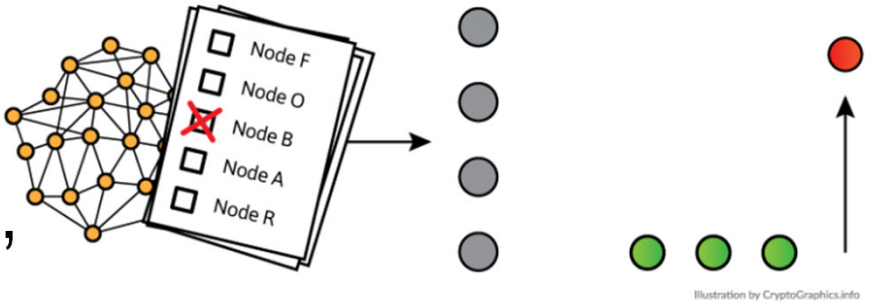
$f = \frac{n-1}{3}$ (Max # of faulty nodes)

PBFT property

- Other examples: XFT, HoneyBadger

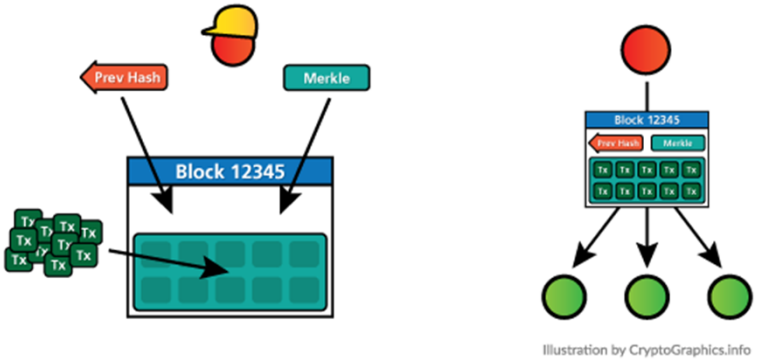
delegated BFT (dBFT)

- Nodes in the network elect a group of consensus nodes (*e.g.*, CoZ)
- Leader/speaker randomly chosen from consensus nodes, remainders are delegates



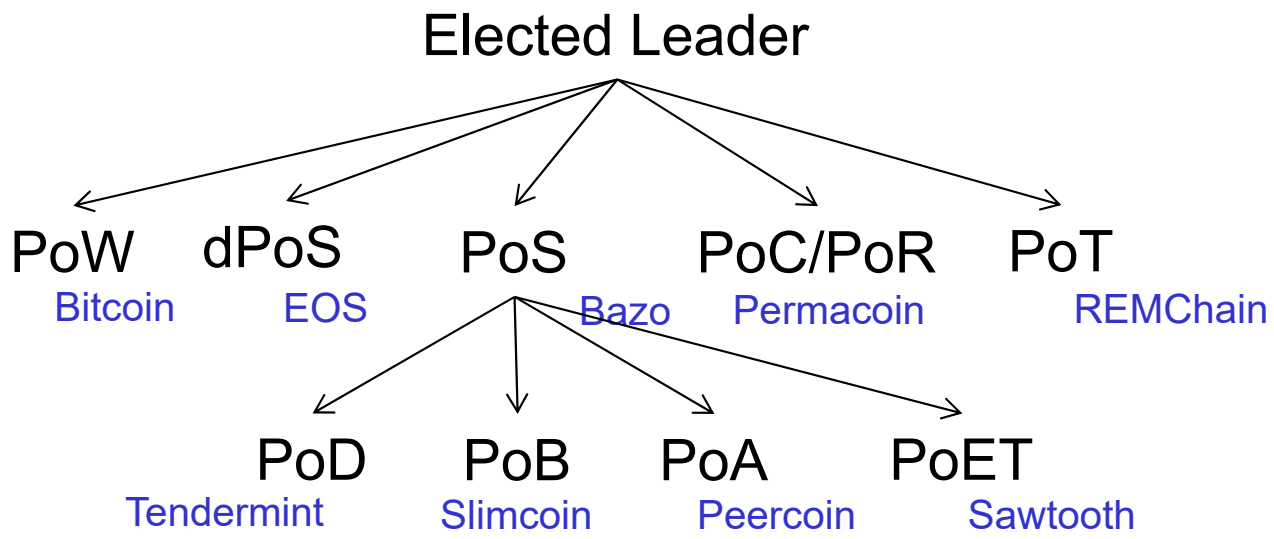
CoZ: City of Zion (NEO BC)

- Leader/speaker creates new block, needs to be positively checked by 2/3 of all delegates
- If 2/3 agree, block is added to the chain
- Countermeasures for dishonest leader/speakers or delegates



Consensus Mechanism Types (2)

- Elected Leader Models
  - Probabilistic elected leader in a
    - Lottery-like
    - Competition, or
    - Probabilistic algorithm



PoX: Proof-of-X, where *X*=

A: Age
B: Burn
C: Capacity (storage)
D: Deposit
S: Stake
T: Trust
R: Retrievability
W: Work
d: delegated

Proof-of-Work (PoW)

- Set of transactions becomes available, block is created, by utilizing the following data
  - Transaction(s), hash of previous block
  - Nonce (arbitrary number, can only be used once)
  - Other information (depending on BC)
- Hash of new block is calculated
- Checking performed once hash was computed
  - Hash is above the target value → Another miner may have found a suitable hash, block attached to local BC, but miner lost the lottery, otherwise nonce will be incremented, retry
  - Hash is below the target value → This miner won the lottery and the new block's hash determines the PoW result

Partial Hash Collision (PoW)

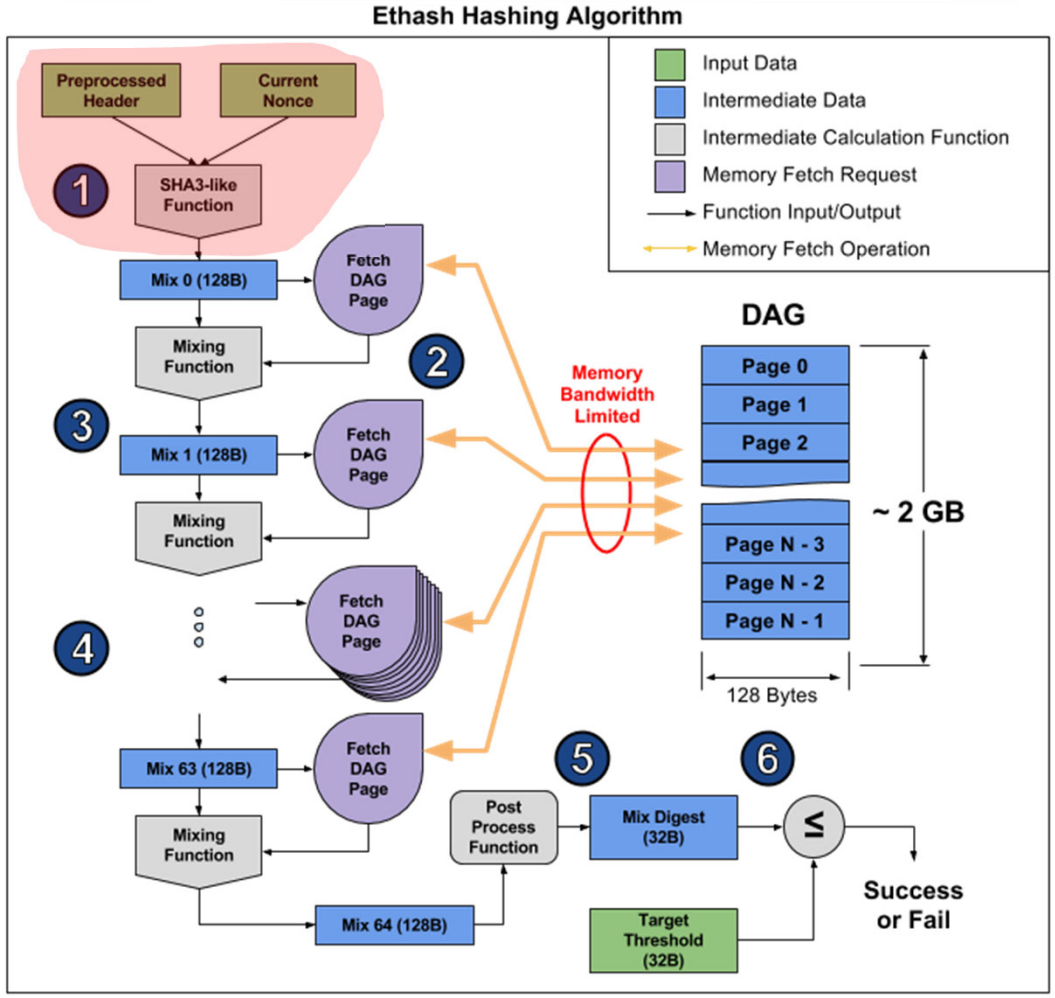
- Key: One cannot compute an input from an output
  - To find a hash with *N* zeros at input start, requires 2**N* computations, which proves computational work performed
  - Hashing an incrementing "nonce" as hash input, leads to zeros

```
in 3e-05 seconds, nonce = 0 yielded 0 zeros.  value = 4c8f1205f49e70248939df9c7b704ace62c2245aba9e81641edf...
in 0.000138 seconds, nonce = 12 yielded 1 zeros.  value = 05017256be77ad2985b36e75e486af325a620a9f29c54...
in 0.000482 seconds, nonce = 112 yielded 2 zeros.  value = 00ae7e0956382f55567d0ed9311cfd41dd2cf5f0a7137...
in 0.014505 seconds, nonce = 3728 yielded 3 zeros.  value = 000b5a6cfc0f076cd81ed3a60682063887cf055e47b...
in 0.595024 seconds, nonce = 181747 yielded 4 zeros.  value = 0000af058b74703b55e27437b89b1ebcc46f45ce55d6....
in 3.491151 seconds, nonce = 1037701 yielded 5 zeros.  value = 00000e55bd0d2027f3024c378e0cc511548c94fbeed0e....
in 32.006105 seconds, nonce = 9913520 yielded 6 zeros.  value = 0000007 7a77854ee39dc0dc996dea72dad8852afbde6....
in 590.89462 seconds, nonce = 186867248 yielded 7 zeros.  value = 0000000225060b16117b23dbea9ce6be86ac439d....
in 4686.171007 seconds, nonce = 1424462909 yielded 8 zeros.  value = 000000002dd743724609a9f57260e2492908d....
```

- Distributed game sets the difficulty *N* of the game
- Players accumulate points by creating blocks
  - Hashing the previous block, finding a hash of the new block with enough zeros, and transmitting this block to everyone

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)



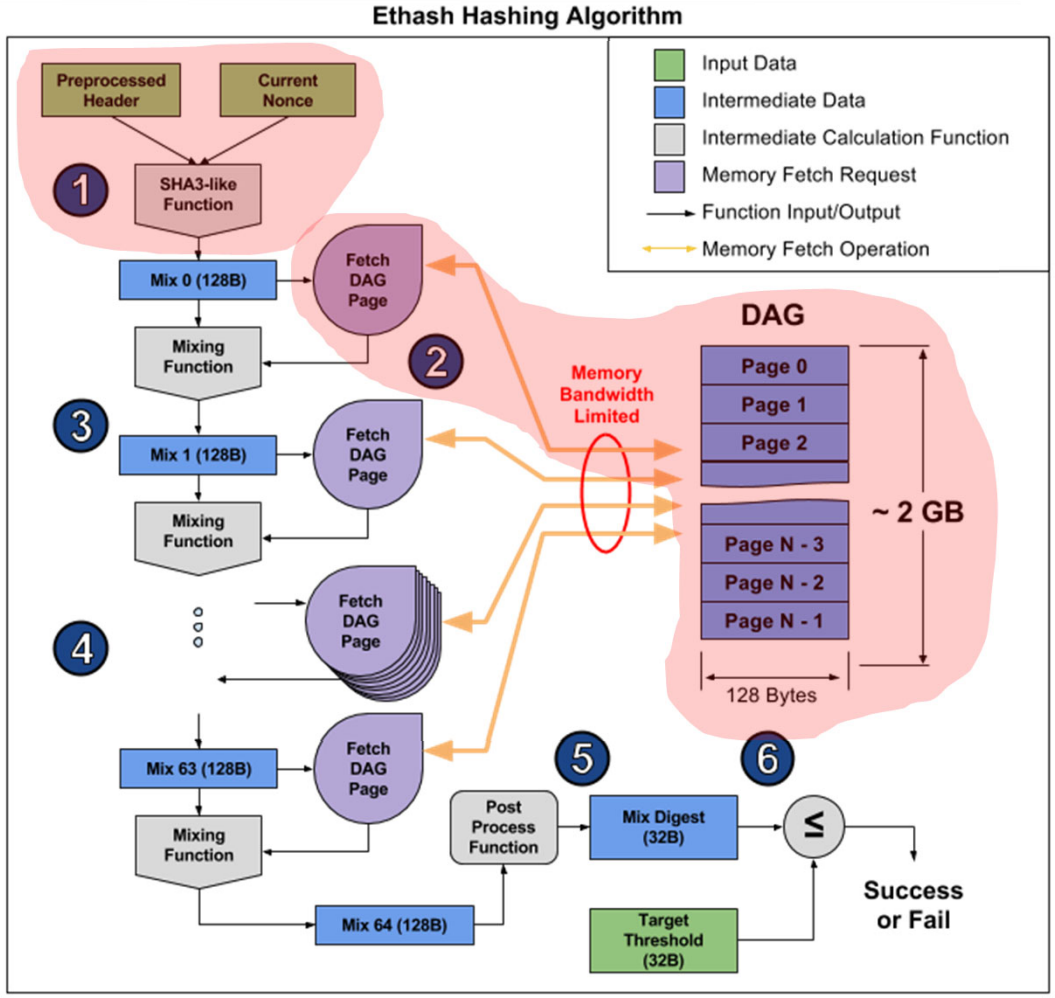
1

The **Pre-processed Header** (derived from the latest block) and the **Current Nonce** (the current guess), are combined using a SHA3-like algorithm to create our initial 128 byte mix, called **Mix 0** here.

- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)



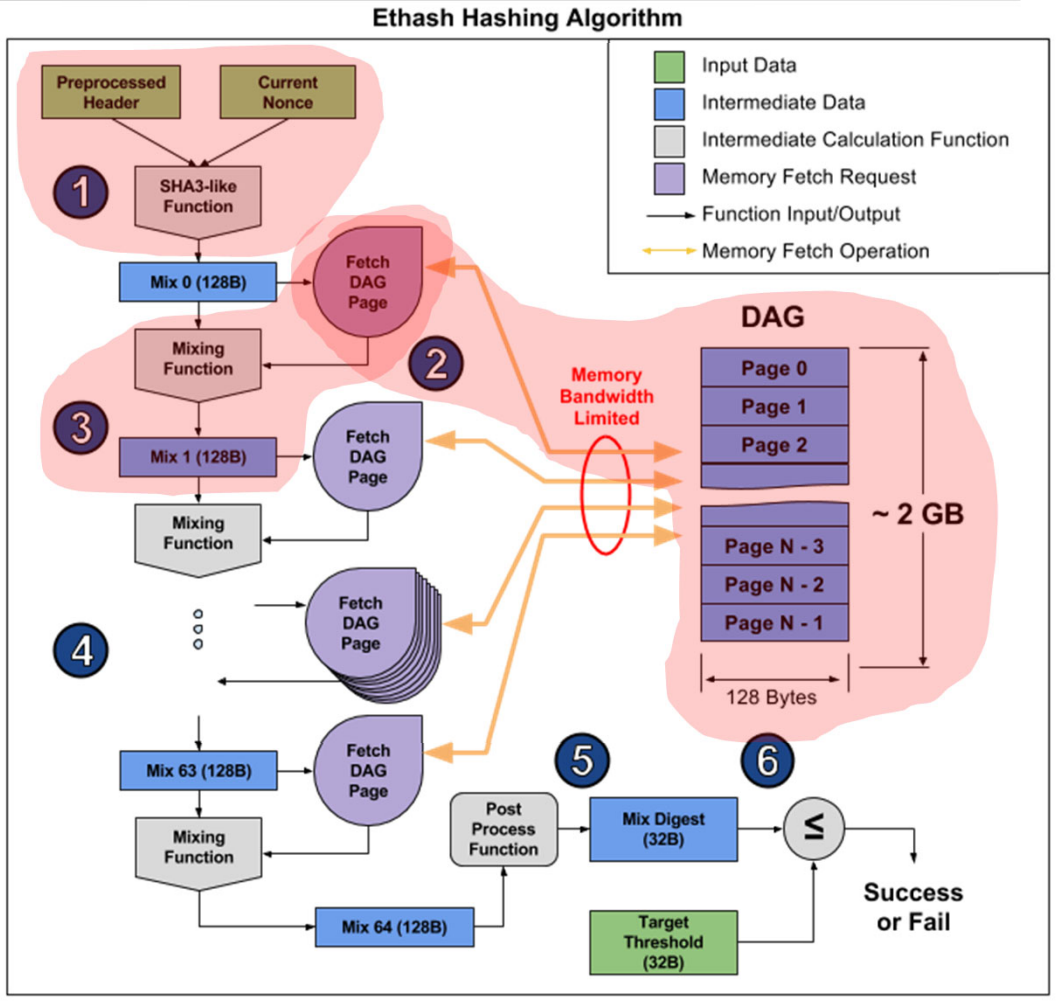
1 2

The **Mix** is combined with the retrieved DAG page. This is done using a Ethereum-specific mixing function to generate the next mix, called **Mix 1** here.

- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)



1 2 3

The **Mix** is used to compute which 128 byte page from the DAG to retrieve, represented by the **Get DAG Page** block.

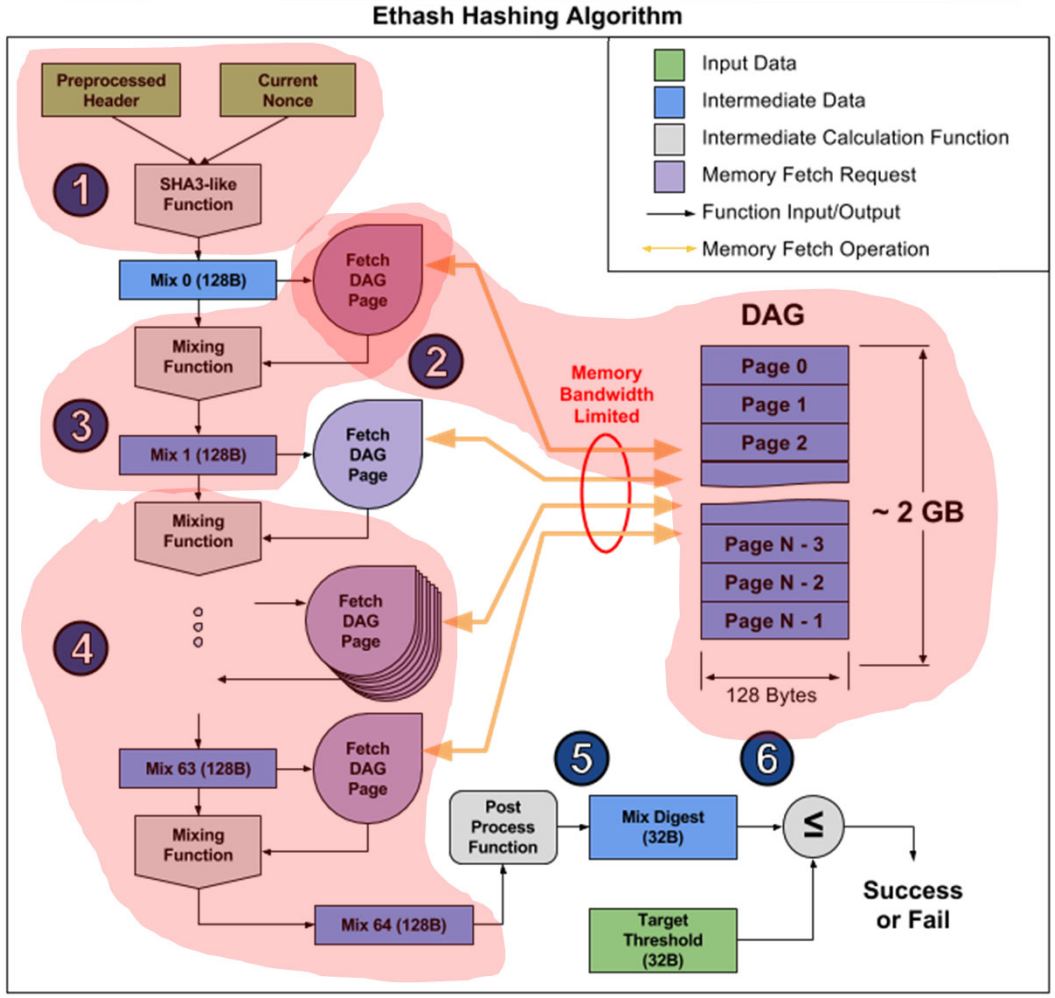
- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)



Steps 2 & 3 are repeated 64 times, finally yielding **Mix 64**.



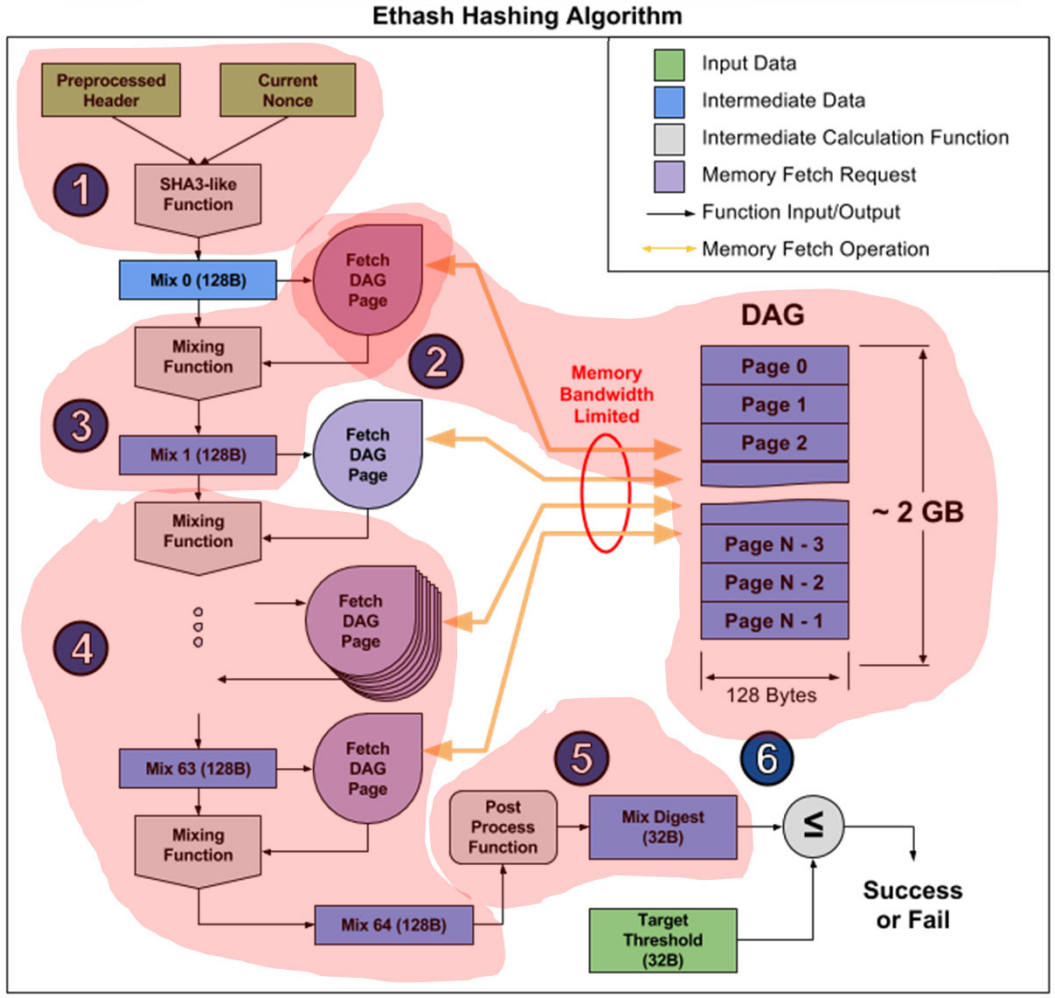
- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)

1 2 3 4 5

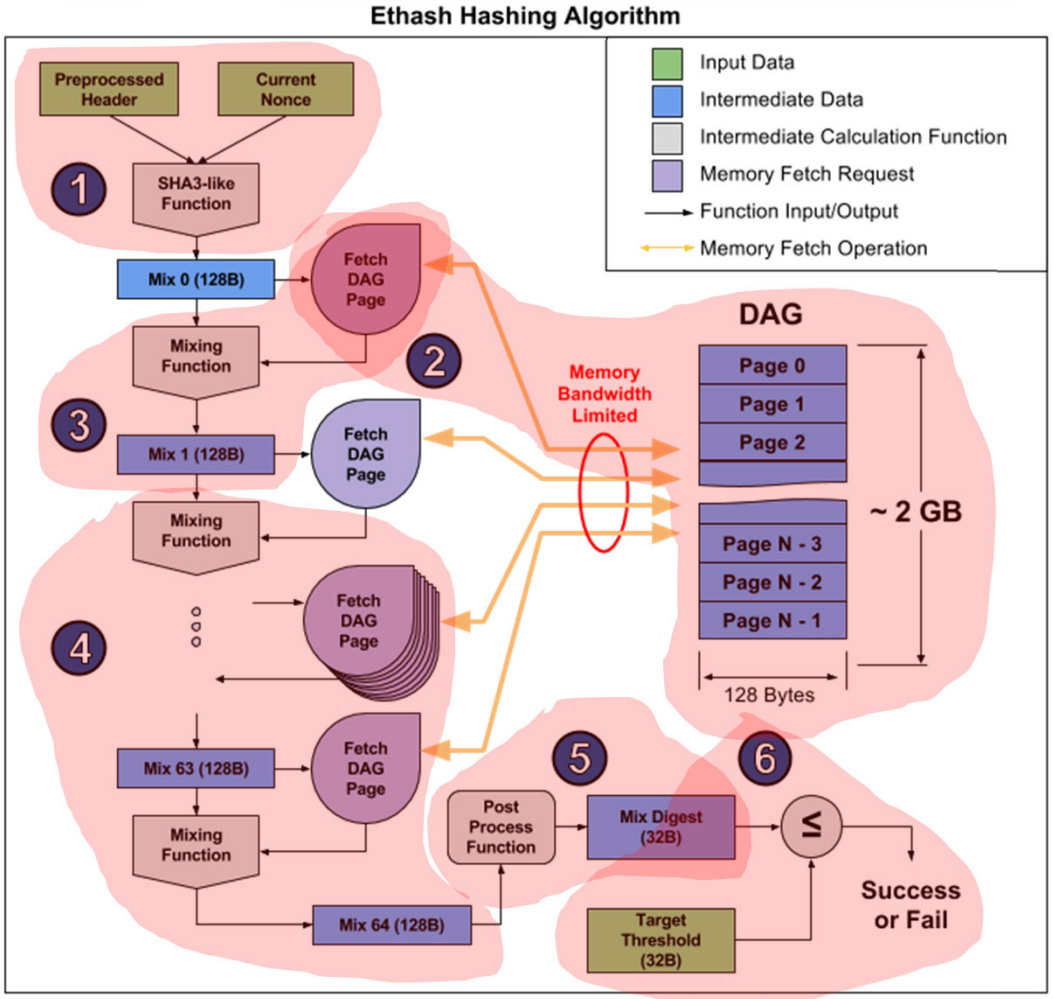
Mix 64 is post processed, yielding a shorter, 32 byte **Mix Digest**.



- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

Ethereum PoW – Ethash

- Also based on partial hash collision (target hash)
  - Works as a memory-bound cryptographic sponge
  - Requires heavy fetching of data on the DAG (RAM intensive process)



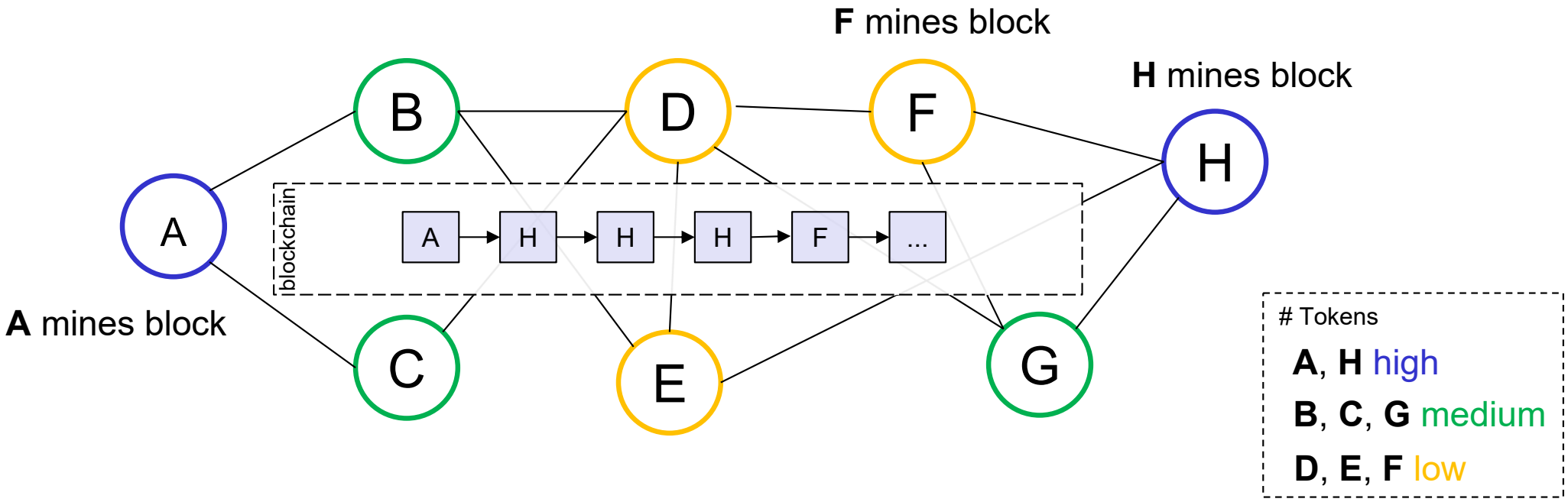
1 2 3 4 5 6

Mix Digest is compared against the predefined 32 byte **Target Threshold**. If **Mix Digest** is less than or equal to **Target Threshold**, then the **Current Nonce** is considered successful, and will be broadcast to the Ethereum network. Otherwise, **Current Nonce** is considered invalid, and the algorithm is rerun with a different nonce

- DAG (Directed Acyclic Graph) - randomly generated dataset
- Designed to be ASIC-resistant as it only require large amounts of memory, which favors smaller miners

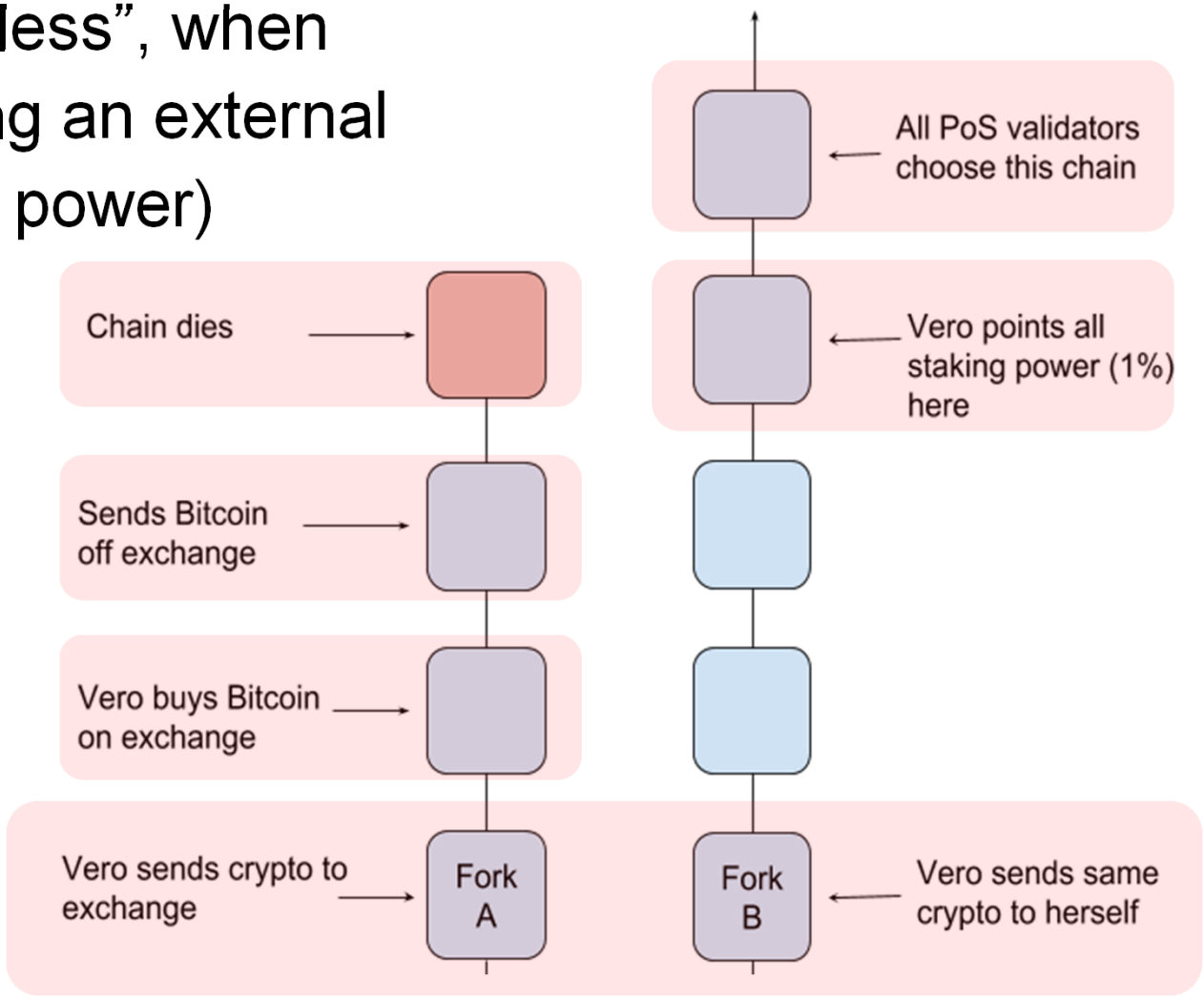
Proof-of-Stake – PoS (1)

- Blocks are “mined” according to the amount of “tokens” he or she holds (stake)
  - The higher is the number of tokens (coins) at stake, the higher is the “mining power” (linear relation)
  - Nodes receive block reward as inherent incentive



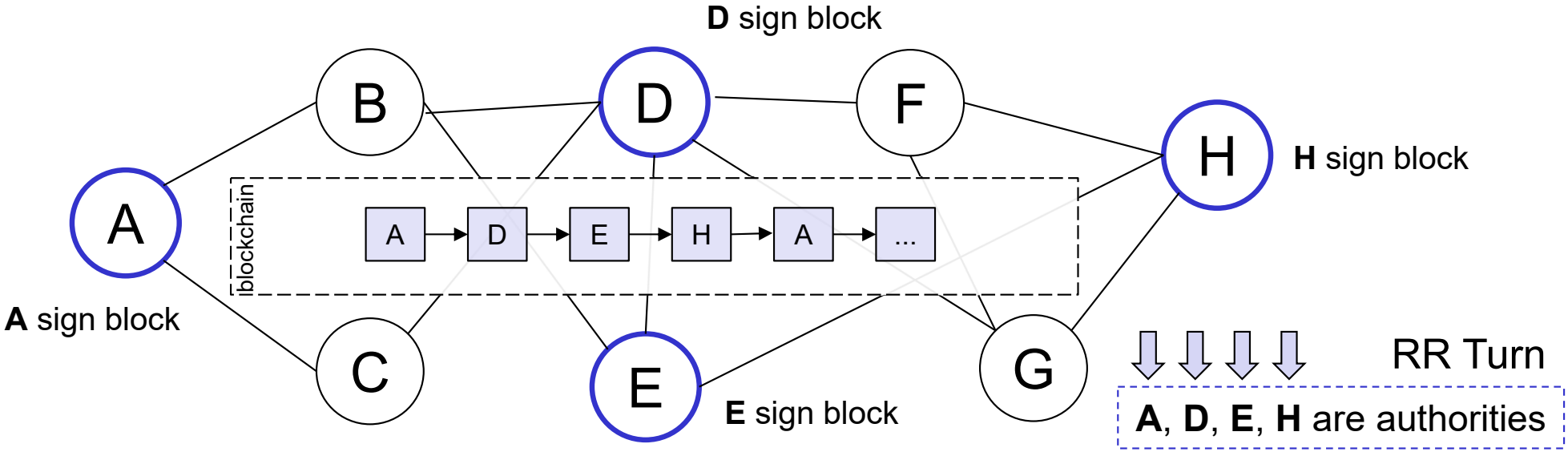
Proof-of-Stake – PoS (2)

- Nothing-at-stake problem
  - Creating forks is “costless”, when someone is not burning an external resource (*e.g.*, mining power)
    - PoS alone is “unworkable” and leads to misuse
- PoS variant:
  - DPoS: nodes can delegate their stakes to a list of nodes



Proof-of-Authority (PoA)

- PoA is a modified form of PoS, where instead of stake a validator's identity performs the role of stake
- Authorities (nodes) are allowed to create news blocks
  - Clique (practical implementation) of PoA
    - Requires N/2+1 (more than 50%) of signers to be honest
    - Authorities sign new blocks in a Round-robin (RR) fashion



Alternatives (1)

- Proof-of-Burn (PoB)
  - Mechanisms should proof that participants burned “coins”
    - Sending them to a verifiably unspendable address
    - This is expensive from each individual’s point of view
    - But it consumes no resources other than burned asset
    - Ultimate source of scarcity remains PoW-based cryptocurrency



- Proof-of-Capacity (PoC)
  - Also known as Proof-of-Space



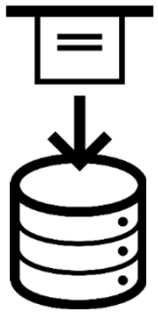
    - Miner has to proof a (free) storage capacity to be entitled to create blocks
    - Miners has to stake their disk (hard-drive or SSD) capacity to create blocks
      - Similar to PoS in this sense

Alternatives (2)

- Proof-of-Elapsed Time (PoET)
  - PoET follows a lottery system that spreads chances of winning equally across network participants



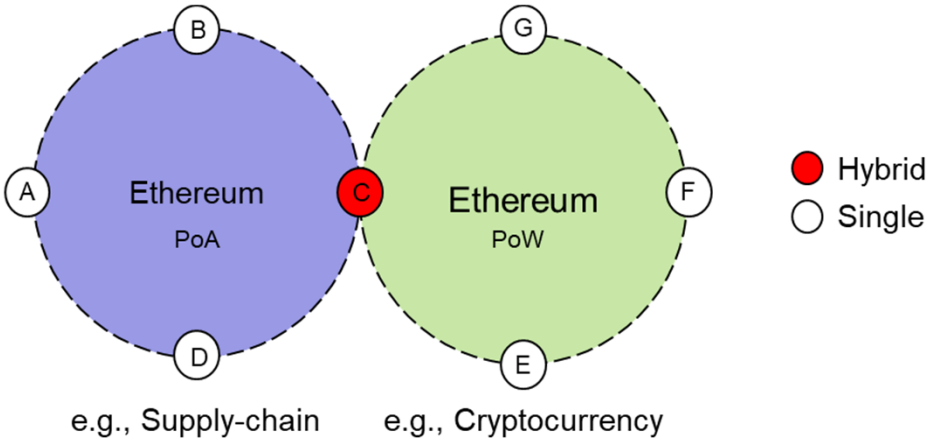
    - Generates a random wait time for each node in the blockchain network
    - Each node must go to sleep for that duration
    - The node with the shortest wait time will wake up first and will win the block, thus, being allowed to commit a new block to the blockchain
- Proof-of-Retrievability (PoR)
  - Defined as “a compact proof by a file system (prover) to a client (verifier) that a target file *F* is intact”



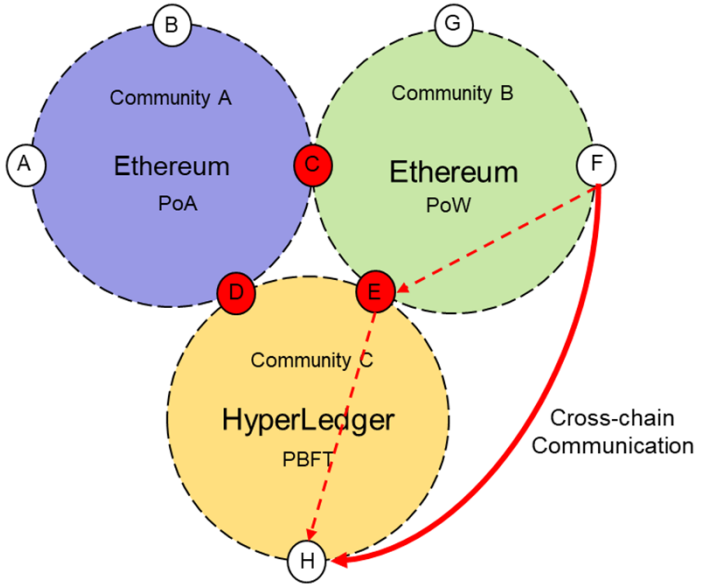
    - A client stores a message ***m*** on a remote server and keeps only a short private ***fingerprint*** locally
    - At some later time, when the client wishes to verify the integrity of its message, it can run an audit protocol in which it acts as a verifier, while the server proves that it has the client’s data

Hybrid Consensus

- Hybrid consensus models
  - Using a single consensus results in limitations
    - Combination of different consensus mechanisms



- Hybrid sharding
  - System can be organized into shards (communities)
    - Cross-chain communications
    - Applied by CSG's Bazo BC (*cf.* later)



Applications of Consensus Mechanisms within CSG Projects

Pollution Monitoring



Trading Contracts

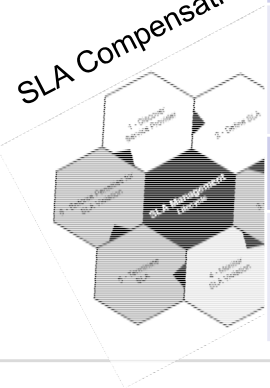


Project	BC Technology	Application Domain	Era	Consensus
IoT-based Pollution Monitoring	Ethereum	Environment	3.0	**PoW**
Blockchain-based Trading Contracts	Ethereum	Marketplaces	3.0	**PoW**
Blockchains for Cold-chains (BC4CC)	Multiple	Interoperability	4.0	**Multiple**
BLW: Foodchain	Ethereum	Food Supply-chain	3.0	**PoW**
Cryptocurrency from Scratch: BAZO	BAZO	Cryptocurrency	1.0	**PoS**
Blockchain Signaling System (BloSS)	Ethereum	Network Security	3.0	**PoA**
Blockchain eVoting	Dedicated BC	eVote	3.0	**PoA**
Automated SLA Compensation	Ethereum	Business Support	3.0	**PoW**

BloSS



SLA Compensation



A Comparison of Blockchain Consensus Protocols (1)

Consensus	Block Proposal	Block Validation	Information Propagation	Block Finalization	Incentive Mechanism	Fault Tolerance	Throughput (TPS)
BFT-SMR (PBFT, Honey-BadgerBFT)	Client operation request	Signature check	Broadcast	Mutual agreement on the same state	N/A	33% servers	Thousands [1 s – 1 m]
Nakamoto (Bitcoin, Litecoin)	PoW	PoW check	Gossiping	Longest-chain rule	Block reward and transaction fee	50% computing power	Sub-ten [10 m- 60 m]
Nakamoto-GHOST (Ethereum)	PoW (Ethash)	PoW check	Gossiping via secure channels	A variation of GHOST rule	Block reward and transaction fee	50% computing power	Tens [6 m – 10 m]
Bitcoin-NG (Waves-NG)	PoW for key blocks	PoW check for key blocks	Gossiping	Longest-chain rule	Block reward and transaction fee	50% computing power	Hundreds [10 m]
Hybrid PoW-BFT (PeerConsens, SCP, ByzCoin)	PoW-based BFT committee election	PoW check	Broadcast among BFT committee	Longest-chain rule for chain (PBFT for transactions)	N/A	33% computing power	Hundreds [10 s – 1 m]
Chain-based PoS (Peercoin, Nxt, PoAct)	PoS	PoS check	Gossiping	Longest-chain rule	Block reward and transaction fee	50% deposited stake value	Tens [10 m - 60 m]
Committee-based PoS (Ouroboros, Praos, CoA, Snow White)	PoS-based committee election	Proposer eligibility check	Broadcast among committee	Longest-chain rule	Block reward	50% token wealth	Hundreds [1 m - 10 m]

A Comparison of Blockchain Consensus Protocols (2)

Consensus	Block Proposal	Block Validation	Information Propagation	Block Finalization	Incentive Mechanism	Fault Tolerance	Throughput (TPS)
BFT-based PoS — Tendermint (Cosmos Hub)	PoS-based round robin	Proposer eligibility check	Broadcast among validators	BFT (adapted DLS)	Block reward	33% token wealth	Thousands [1 s – 3 s]
BFT-based PoS — Algorand	PoS-based committee Election	Proposer eligibility check	Broadcast among committee	BFT (adapted Byzantine agreement)	Block reward	33% token wealth	Hundreds [20 s - 1 m]
BFT-based PoS — Casper FFG (Ethereum 2.0)	PoW (Ethash)	PoW & Checkpoint tree check	Broadcast among validators	BFT (with staked votes)	Block reward for miners and validators	33% deposited stake value	Thousands (if sharding used)
DPoS (EOSIO, Lisk, BitShares)	PoS with stake Delegation	Delegate eligibility check	Broadcast among delegates	BFT (suggested, not limited to)	Block reward and vote reward	33% delegates	Thousands [< 1 s]
PoA (Rinkby, Kovan, POA Network)	PoA	Block proposer identity check	Broadcast	HoneyBadger-BFT	Transaction fee	50% TEEs (33% if BFT used)	Tens to hundreds [10 s - 1 m]
PoET (Hyperledger Sawtooth family)	PoET within TEE	TEE certificate check	Broadcast	PBFT	N/A	50% IDs (33% if BFT used)	Tens to hundreds [10 s - 1 m]
PoTS	PoTS based committee Election	Proposer eligibility& TEE cert. check	N/A	N/A	N/A	50% token wealth	Unavailable
PoR (Permacoin)	PoR	File retrievability check	N/A	N/A	N/A	50% storage space	Unavailable

© 2022 UZH, CSG@IfI

A Comparison of Blockchain Consensus Protocols (3)

Consensus	Block Proposal	Block Validation	Information Propagation	Block Finalization	Incentive Mechanism	Fault Tolerance	Throughput (TPS)
RPCA (Ripple)	Any server can propose transactions	UNL membership check	Broadcast to UNL peers	Accepting > 80% voted transactions	Transaction fee	20% nodes in each UNL	Thousands [< 1 s]
BlockDAG — SPECTRE — PHANTOM	PoW	PoW check	Gossiping	Pairwise ordering/ kcluster- blocks Ordering	N/A	50% computing power	Thousands or more [10 s - 2 m]
BlockDAG — Conflux	PoW	PoW check	Gossiping	GHOST rule to finalize pivot chain	N/A	50% computing power	Thousands [5 m - 20 m]
TxDAG — Tangle (IOTA)	Approving two tips & PoW	Tip approval check and PoW check	Gossiping	Highest cumulative weight rule	Eligibility for issuing new transactions	50% computing power	Thousands or more [1 m - 10 m]
TxDAG — Byteball (Obyte)	Approving tips per main chain index	Tip approval Check	Gossiping	Witnesses consensus on main chain	Commissions collected from storage fees	50% (or 6) witnesses	Thousands or more [30 s – 10 m]
TxDAG — Nano (Nano)	Cross-chain sender-receiver & PoW	Sender account and PoW check	Gossiping	Stake weighted voting	Unspecified	50% token wealth	Thousands or more [1 s - 10 s]
TxDAG — Avalanche	Approving one parent	Parent approval check	Gossiping	Confidence ordering to solve conflict	Unspecified	50% participants	Thousands [1 s - 5 s]

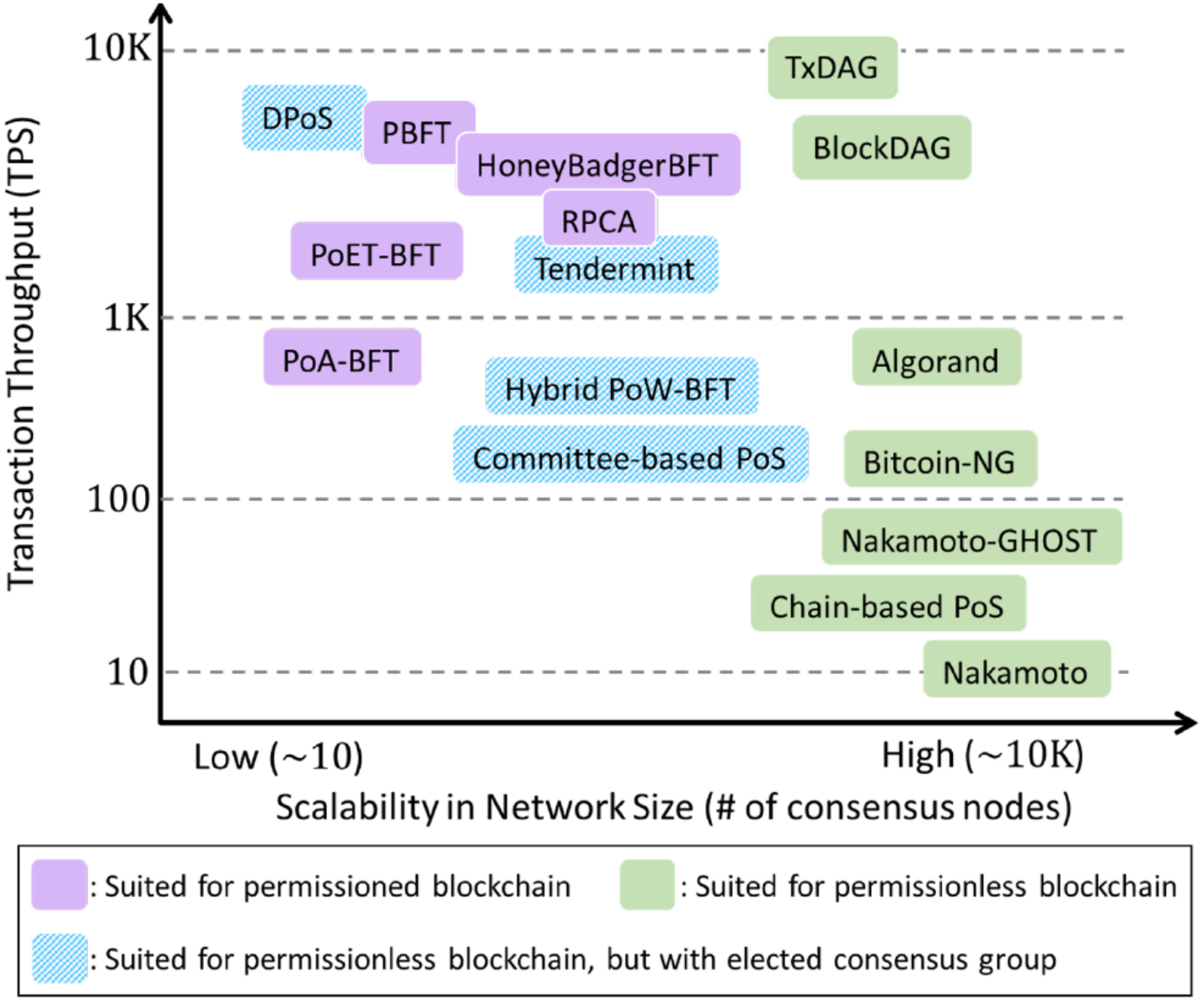
© 2022 UZH, CSG@IfI

Details on Aspects Used for Comparison

- GHOST: Greedy Heaviest Observed Sub-tree
- Bitcoin-NG: Next Generation Bitcoin
  - Extends concept of electing a leader in BTC to selecting a leader in an epoch, who handles all trans-actions until a new leader is selected.
  - Performance improvement by decoupling Bitcoin's blockchain operation into two planes: leader election and transaction seriali-zation. It divides time into epochs, each runninf a single leader
- SCP: Scalable Consensus Protocol (network partitioned into shards)
- Nxt is „Peercoin Nxt"
- CoA: Chain-of-Activity
- DLS is the author's name (as stated in the paper)
- Casper FFG: Friendly Finality Gadget
- PoTS: Proof-of-TEE-stake (Trusted Execution Environment)
- PoR: Proof-of-Retrievability

https://arxiv.org/pdf/1510.02037.pdf

Overview of Consensus' Scalability



A Survey of Distributed Consensus Protocols for Blockchain Networks
https://arxiv.org/pdf/1904.04098.pdf

Cryptocurrencies and Bitcoin



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Electronic Payment Systems
  - Micropayments
- Cryptocurrencies
  - Definitions
  - Overview
  - ICO
- Bitcoin
  - Characteristics and Selected Technical Details
  - Operations, Protocol, Scripting Language
  - Mining and Evolution
  - Parameters
  - Advantages and Drawbacks

Electronic Payment Systems

Electronic Payments (or Systems)

- Traditional payment approaches
  - Cash, check
- Gap in the past: no transfer of money via the Internet
- Electronic payment approaches
  - Means of running payments electronically over a network, such as the Internet or dedicated networks (IP-based)
  - Examples (Systems): Electronic Fund Transfer (EFT), debit/ credit card, ecash, software wallets, smart cards, crypto-currencies, Financial Electronic Data Interchange), interbank clearing (SWIFT)
- Micropayment systems
  - Pre-pay, post-pay, pay-as-you-go (once per transaction)

All with "measurable" centralized elements

Electronic Payments and Cryptocurrencies

- Electronic money created solely by IT means
- Formalized process (centralized or distributed) and software exists, which
  - controls the cryptocurrency's creation,
  - enables and protects transactions, and
  - may hide identities of its users
- Secure, anonymous, and electronic on-line payments determine a set of requirements for any, but at the same time trustful handling approach
  - Similar requirements are posed for any centralized approach

- Advances in cryptography, compute performance, and BCs as an underlying distributed ledger enable fully decentralized electronic payment solutions

Cryptocurrencies

Cryptocurrencies (1)

- A form of digital cash with general characteristics
  - Faster, cheaper, reliable than “paper/coin” issued currency
- Main two alternatives for creation of coins
  - Issued by government(s) – centralized approach
    - Involving Central Bank, banks, brokers, end-users
  - Issued in a fully distributed manner – modern cryptocurrency
    - Users transact directly with each other (no involvement of banks)
    - Users store money themselves
- General problems of cryptocurrencies
  - Countermeasures against distributed fraud and manipulation
    - Lack of trust
    - Malicious transactions
    - Double spending

Cryptocurrencies (2)

- Automated prevention mechanisms against fraud
  - Issuing of currency to be traceable & deterministic (algorithm)
  - All transactions can be recorded in a decentralized manner
  - All transactions can be verified, own and everyone's else transactions
    - Cryptographic mechanisms essential
- Blockchains, distributed ledgers or public records, provide the key basis for modern cryptocurrencies
  - They do not require a Trusted Third Party (TTP) nor trust as such, neither banks or users → "trustless"
  - Everyone can trace the money being sent, received, verified, and recorded

Blockchains and Cryptocurrencies

- Cryptocurrencies can either be
  - Mined or
  - Pre-created/burned

Mining Crypto Currency (CRY-M)	Pre-creating Crypto Currency with Distribution (CRY-P)	No Use of Crypto Currency (NCRY)
The result of mining is a block with a reward in the form of crypto currency. Bitcoin and Ethereum reward with bitcoins and ethers, respectively. Some blockchains allow to define various other crypto currencies or assets besides its native crypto currency.	Instead of mining crypto currency, the currency can be pre-created and distributed in an Initial Coin Offering (ICO). The incentive to mine a block is to collect transaction fees. Other variations include "Proof-of-Burn" (PoB) or "Proof-of-Possession" (PoP) using another crypto currency.	Some blockchains do not need any kind of native crypto currency, but allow for overlay assets. Especially private blockchains do not use a native currency.

Cryptocurrency Overview (Never Complete)

~ 2900 cryptocurrencies today:
https://coinmarketcap.com/

Approach	Accessibility	Consensus	Crypto Currency
Bitcoin	Public	PoW/ASIC	CRY-M/Bitcoins
Ethereum	Public	PoW/MEM-HARD	CRY-M/Ethers
Ethereum Casper/Serenity	Public	PoS/PoA	CRY-M/Ethers
Litecoin	Public	PoW/MEM-HARD	CRY-M/Litecoin
Monera	Public	PoW/MEM-HARD	CRY-M/XMR
Lisk	Public	PoS/Del	CRY-M/Lisk
R3 Corda	Private	TE	NRCY
Openchain	Private/sidechain	TE	NRCY/various
IOTA	Public	PoW	CRY-P/IOTA tokens
Eris:DB	Private	PoS	CRY-M/Ethers
Chain Core	Private	TE	NRCY/various
Hyper Ledger	Private	TE/PoW/PoS	NRCY
Nxt	Public	PoS	CRY-P/various
Stratis	Private/sidechain	PoW (PoS in future)	CRY-M/STRAT token
Multichain	Private	PoW	NRCY/various
BigchainDB	Private	TE	NRCY/various
Rootstock	Public/sidechain	PoW (Bitcoin)	CRY-M/Bitcoin-Rootcoin
Counterparty	Public	PoW (embedded Bitcoin consensus)/PoB	CRY-P/various
Ripple	Public	TE	CRY-P/Ripple/various
Stellar	Public	TE/PoP	CRY-P/Lumen/various

Cryptocurrency ≠ Blockchain

Many Coins with Similar Mechanisms

- All electronic, digital, backed by a "scarce" resource
  - Thus, avoiding double spending is the key!
  - Bitcoin: SHA-256 partial hash collision: time, CPU, electricity
  - Ethereum: variant of Dagger-Hashimoto, time, CPU, memory, electricity, miner store dataset: 1 GB, verification needs 16 MB
  - Litecoin: script partial hash collision: time, CPU, memory, electricity
  - Ripple XRP: Unique node list (trusted validators, 1000): Web of trust
  - PPCoin/Peercoin: Proof-of-Stake (PoS)/PoW:
    - Holding 1% will generate 1% of coins
    - 1% inflation per year
    - Energy-efficient PoS

https://coinmarketcap.com/

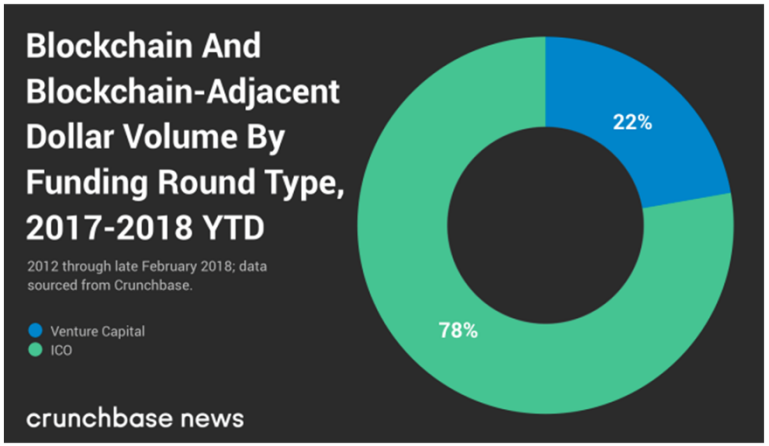
Initial Coin Offering (ICO)

- Initial Coin Offering (ICO) is a means of crowdfunding
  - Tokensale: release of a new cryptocurrency, e.g., Ethereum
- It can be done with today's technologies! However, …
  - Micro payments with today's banking system not really feasible
    - Solution: Use central service collection funds until threshold
  - Banking transactions are expensive (sending to Africa/India?)
    - Solution: Increase threshold (*e.g.*, low for SEPA, higher otherwise)
  - IBAN can change – if you change your bank
    - Solution: keep IBANs in sync with users where possible
- Alternatively, use of a blockchain-based approach
  - Publicly, world-wide accessible via cryptocurrency
  - ICOs offer a dedicated "payment system" alternative

ICO Economic Background

- ICO Funding https://www.icodata.io/stats/2018
  - 2019: US$ 346,089,025 (until July 31)
  - 2018: US$ 7,812,150,041
  - 2017: US$ 6,226,689,449
  - 2016: US$ 90,250,273
- ICOs delivered at least 3.5x more capital to blockchain startups than VC (Venture Capital) in 2017+2018

The bubble has imploded …



Wall Street Journal:
20 percent of all ICOs are fraudulent
SATIS Group Report:
78% of ICOs are Scams

Currently "open" ICOs:
https://topicolist.com/upcoming-icos

ICO Pros/Cons and Technical Steps

Fundraising Advantages	Drawbacks	Investor Advantages	Drawbacks
Everyone can participate	Likelihood of failure, 2017 52% of ICOs failed	There is a “token”!	Token may show no value or worse, will be scam
Global audience	Time-consuming technical and administrative set-up	Easy participation for tech-savvy people	Due diligence time-consuming, even is tech-savvy people are involved
	Regulations country-specific		Locally incompatible regulation

- Create public/private key pair for pay-ins
  - Store these keys in a bank vault, copy public keys
- Create an ICO backend/frontend
  - Backend with minimal endpoints: tiers, register, address, whitelist
  - ICOnator: easy, secure, configurable, scalable open source ICO engine

https://iconator.io/

- Minting – aka – create coins out of thin air ☺
- Consolidation: x investors, x accounts → 1 account (KYC!)

KYC: Know Your Customer

Bitcoins

WIRED SUBSCRIBE » SECTIONS » BLOGS » REVIEWS » VIDEO » HOW

Sign In | RSS Feeds

FEATURES
For *John Carter*, Director Andrew Stanton Leaps From Animation to Live-Action Sci-Fi

START
MIT's Sebastian Seung Wants Computers to Map the Brain

PLAY
The Five-Year Engagement Takes Director Nick Stoller Off the Grid

/ MAGAZINE

FEATURES 19.12

The Rise and Fall of Bitcoin

759 348 123

By Benjamin Wallace ✉ November 23, 2011 | 2:52 pm | Categories: Wired December 2011

Tweet +1 Share



The Economist

Log in | Register | Subscribe

World politics | Business & finance | Economics | Science & technology | Culture | Blogs

Babbage
Science and technology



Previous | Next | Latest Babbage Latest from

irtual currency

Bits and bob

Jun 13th 2011, 20:30 by [illegible] | LONDON AND G.T. | MELBOURNE

Like 1.8k Tw

de fr it

Ihr Ort: Zürich 19° Mi [illegible] Übersicht Schweiz

Registrieren Login

20 minuten ONLINE

Video TV Infografik Games [illegible]ekte Apps RSS Inhalt A-Z Suchen

Schweiz | Ausland | Panorama | Wirtschaft | Sport | Ei[illegible]/M | [illegible]eo[illegible] | Entertainment | Digital | Mehr

News SMI Alle Indices Ratgeber Geld [illegible] [illegible]ukte Finanzrechner Dossiers

Ihre Story, Ihre Informationen, Ihr Hinweis? feedback@20minuten.ch

BITCOIN, DIE DEVISE IM WEB

07. Juni 2011 07:15; *Akt: 07.06.2011 09:11*

Der gefährliche Cyber-Dollar

von Gérard Moinat - Die Online-Währung Bitcoin wirft hohe Wellen. Es sei das «gefährlichste Open-Source-Projekt aller Zeiten» und «gefährde

powered by homegate.ch

Immobilien in Zürich

1.0 Zimmer Zi , Charmante möblierte Zimmer im Herzen von Zürich
Hornergasse 15 8001 Zürich



Immobilien finden

PLZ 8000

Preis unlim. bis unlim.

From 2011 – already a decade ago!

Bitcoins in the News

- March 31, 2017 (NZZ)
  „Die Digitalisierung revolutioniert den Rohstoffhandel“
- March 25, 2017 (Handelsblatt)
  „Wann kommt der Durchbruch? “
- March 28, 2016 (FAZ)
  „Der nächste große Umbruch
  in der Finanzwelt rückt näher “
- April 12, 2017 (WSJ)
  „People Love Talking About Bitcoin More Than Using It“

As of 2017

KOMMENTAR

Bitcoin über 8000 Dollar: Die Spekulationslust ist zurück

Der Bitcoin ist wachgeküsst. Innerhalb von Wochen hat sich die Notierung der Kryptowährung verdoppelt.

Werner Grundlehner
14.5.2019, 16:21 Uhr

In 2018+19

Nach einer langen Agonie über fast das ganze Jahr 2018 und einem ebensolchen Jahresbeginn 2019 ist der Bitcoin seit Anfang April von 4000 auf über 8000 $ geklettert.

MEISTGEL

Immer stärk
kalte Enteig

Michael Rasch, F

Bitcoin

Übersicht News Hintergrund Meinung Service

News



Weshalb der Bitcoin ein Comeback feiert

Der Einstieg eines US-Vermögensverwalters in das Geschäft mit Bitcoin ist nur ein Treiber des aktuellen Höhenflugs. Experten warnen, dass mit starken Preisschwankungen zu rechnen ist. Mehr...

Bernhard Kislig. 15.05.2019



Bitcoin durchbricht auch Marke von 8'000 Dollar

Die Kryptowährung setzt ihren Kursanstieg der vergangenen Wochen fort. Mehr...

14.05.2019



Viel Startkapital - ungewisse Zukunft

Stiftungspräsident Ryan Jesperson ist optimistisch, dass das mit einer halben Milliarde Dollar ausgestattete Start-up Tezos seine Probleme überwunden hat. Mehr...

Bernhard Kislig. 04.12.2018



Ist der Bitcoin am Ende?

Fast 60 Milliarden Dollar weniger Wert als noch vor knapp einer Woche: Der Sinkflug vieler Digitalwährungen hat sich auch am Dienstag fortgesetzt. Mehr...

20.11.2018

Bitcoin: Zwei Chancen für neue Kursrekorde

Dass der Bitcoin bald wieder seine Rekordwerte von knapp 20000 $ erreicht viele Beobachter für unwahrscheinlich. Zwei Entscheidungen in den USA aber genau dafür sorgen.

Patrick Herger
22.5.2019, 11:00 Uhr

ERKLÄRT

Bitcoin Hausse vorerst auf Eis gelegt – und jetzt?

Angefangen hat alles vor mehr als zehn Jahren mit einem abstrakten Aufsatz, verfasst unter dem Pseudonym Satoshi Nakamoto. Heute sorgt die Digitalwährung Bitcoin selbst bei normalen Anlegern oft für Bluthochdruck. Die wichtigsten Fakten zum Hype.

Werner Grundlehner / Thomas Schürpf / David Bauer
Letzte Aktualisierung am 15.08.2019

CNBC MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV WATC

CRYPTOCURRENCY

Bitcoin breaks above $16,000 for the first time since January 2018

PUBLISHED THU, NOV 12 2020•6:15 AM EST | UPDATED FRI, NOV 13 2020•2:14

Ryan Browne
@RYAN_BROWNE_

SHARE

KEY POINTS

- Bitcoin's price price rose as high as $16,019 just after 5 a.m according to data from industry site CoinDesk.
- The reason for the move higher wasn't immediately clear, t

Der Bitcoin-Kurs seit Anfang 2020

Kurs in US-Dollar

45.000
40.000
35.000

FXEMPIRE Type EUR/USD or Gold All

MARKETS CRYPTO NEWS FORECASTS EDUCATION FXTM ACADEMY TRADE NOW

4.53 ▲ +0.0210 +0.01% Bitcoin: 46,541 ▼ -1,664.65 -3.58% WTI Oil: 58.153 ▲ +0.0940 +0.16

Home > News

The Crypto Daily – Movers and Shaker January 22nd, 2021

It's a particularly bearish day for the crypto majors. Failure for Bitcoin to move back through to $30,000 levels would weigh heavily on the broader market.

Bob Mason
Jan 22, 2021 12:52 AM GMT



COINTELEGRAPH
The future of money

News Markets Magazine People Cryptopedia Industry Cor

WIRTSCHAFT Mein Profil Suche

Unternehmen & Konjunktur Geld & Recht Karriere Börse

Historisches Hoch

Bitcoin knackt 50'000 Dollar-Marke

Die älteste und wichtigste Cyber-Devise hat eine neue Hürde überwunden. Das liegt vor allem am Elektroautohersteller Tesla – aber nicht nur.

February 16, 2021

Publiziert heute um 14:56 Uhr

0 Kommentare

Mai Juni Juli Aug Sept Okt Nov De Jan 2021 Feb

herunterladen

In 2020+21

Mein Profil Suche

& Konjunktur Geld & Recht Karriere Börse

HitBTC 800+ Trading Pairs, 1 Trading Pl

SHIRAZ JAGATI JUN 07, 2020

Crypto Crime on the Rise — Good Odds of 2020 Becoming a Record-Breaker

Experts believe that a more holistic crypto ecosystem, especially one that is more AML-centric, is needed to help prevent the spread of crypto scams in 2020.

Abo Der Tesla-Chef und die Börsen

Die vier Risiken von Elon Musks Bitcoin-Deal

Der Kauf von Bitcoin für 1,5 Milliarden Dollar durch Musks Unternehmen Tesla hat der Kryptowährung zu einem neuen Allzeithoch verholfen. Die Entwicklung ist auch mit Gefahren verbunden. Wir zeigen die wichtigsten vier davon.

Markus Diem Meier
Publiziert: 09.02.2021, 15:55

50 Kommentare



Vermeintlicher Bitcoin-Millionär

Passwort vergessen - 220 Millionen futsch?

Stand: 13.01.2021 01:38 Uhr

Ein Deutscher in den USA schrieb das Passwort für seine digitale Bitcoin-Börse auf einen Zettel. Der ist nun du Zugang zu den Bitcoins, die inzwische

Von Marcus Schuler, ARD-Studio San Franc

≡ The New York Times

Lost Passwords Lock Millionaires Out of Their Bitcoin Fortunes

Bitcoin owners are getting rich because the cryptocurrency has soared. But what happens when you can't tap that wealth because you forgot the password to your digital wallet?

"Maximum Credible Accident"



© 2022 UZH, CSG@IfI

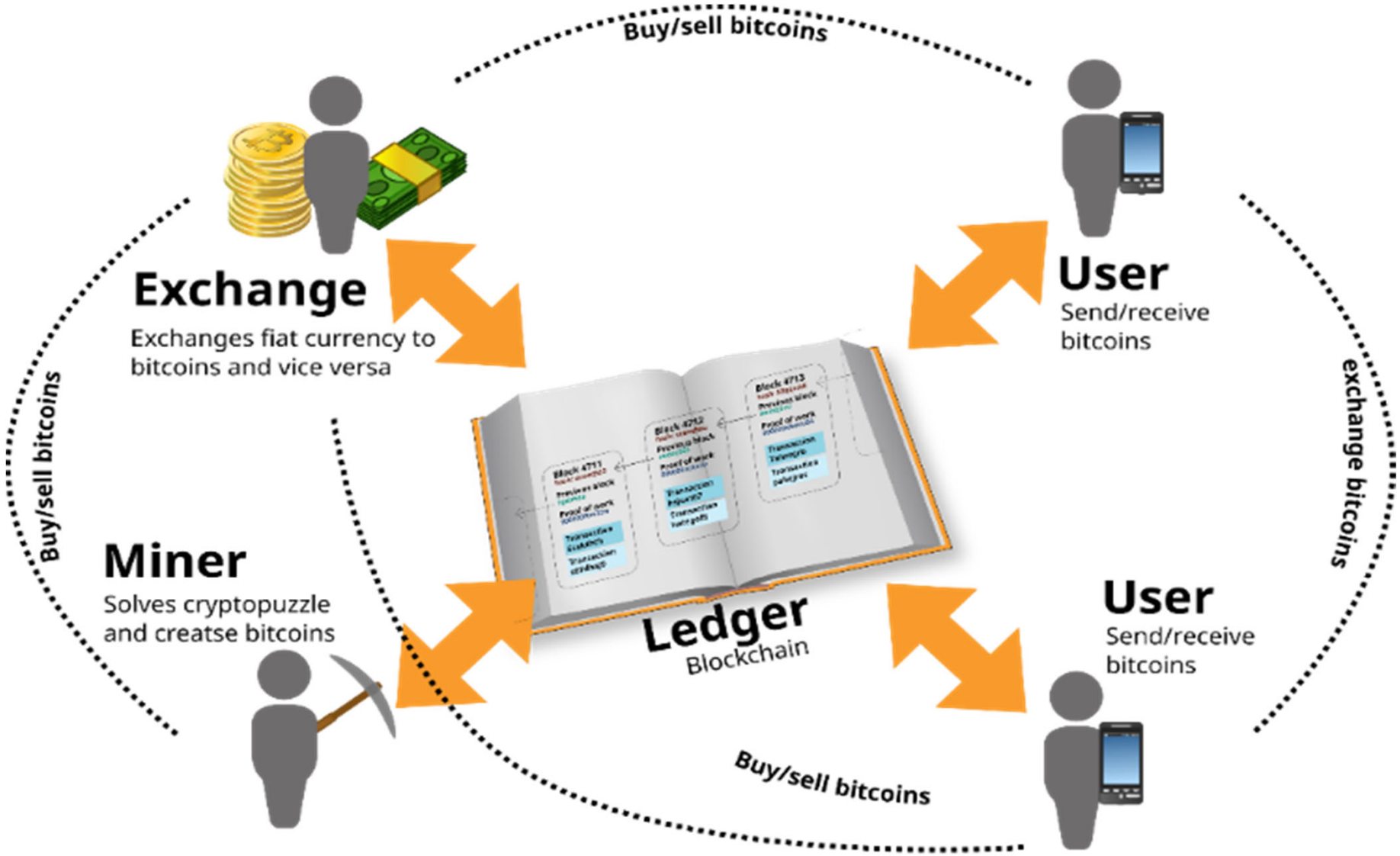
Bitcoin Characteristics (1)

- Bitcoins are an ***experimental*** cryptocurrency (digital)
  - Bitcoin is fully peer-2-peer (no central entity)
  - 1st Bitcoin issued on January 3, 2009
  - Smallest unit: 0.00000001 BTC = 1 satoshi
- Key characteristics
  - Maximum of 21 million BTC
  - Every transaction broadcast to all peers
    - Every peers knows all transactions (~200 GByte as of today)
  - Validation by proof-of-work (partial hash collision SHA-256)
    - Difficult to fake Proof-of-Work (PoW)
    - No double-spending
- The Bitcoin initiator/inventor is unknown so far

Bitcoin Characteristics (2)

- Not relying on trust, but on strong cryptography
- Weak anonymity (pseudonymity)
  - All peers know all transactions
  - Clustering: *e.g.*, if a transaction has multiple input addresses, assume those addresses belong to the same wallet
- Bitcoin not controlled by a single entity
  - Only one development community, but not a central bank – Bitcoin Core's SegWit vs. BTU (Bitcoin Unlimited)
- Creation of Bitcoins performed via the reward for validation of payments (partial hash collisions)
- Bitcoins can be exchanged for real currencies
  - Several companies allow to exchange BTC for Dollar, Euro

The Big Picture – Bitcoin (BTC)



Bitcoin Addresses

- Bitcoin address is an identifier of 26-35 alphanumeric characters, beginning with the number 1 , 3, or bc1 that represents a possible destination for a Bitcoin payment
  - Addresses can be generated at no cost by any user of Bitcoin
- Currently 3 formats
  1. P2PKH which begin with the number `1` , eg: `1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2` .
  2. P2SH type starting with the number `3` , eg: `3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy` .
  3. Bech32 type starting with `bc1` , eg: `bc1qar0srrr7xfkvy5l643lydnw9re59gtzzwf5mdq` .
- Bitcoin transfer: Bitcoin sent to a person by sending bitcoins to one of their addresses
  - Created on demand (offline, too), may be used multiple times
  - Multi-signature addresses: combination of multiple private keys

P2PKH: Pay-to-Publickey-Hash, P2SH: Pay-to-Script-Hash, Bech32: Segregated Witness Address

https://en.bitcoin.it/wiki/Address

Bitcoin Mechanisms



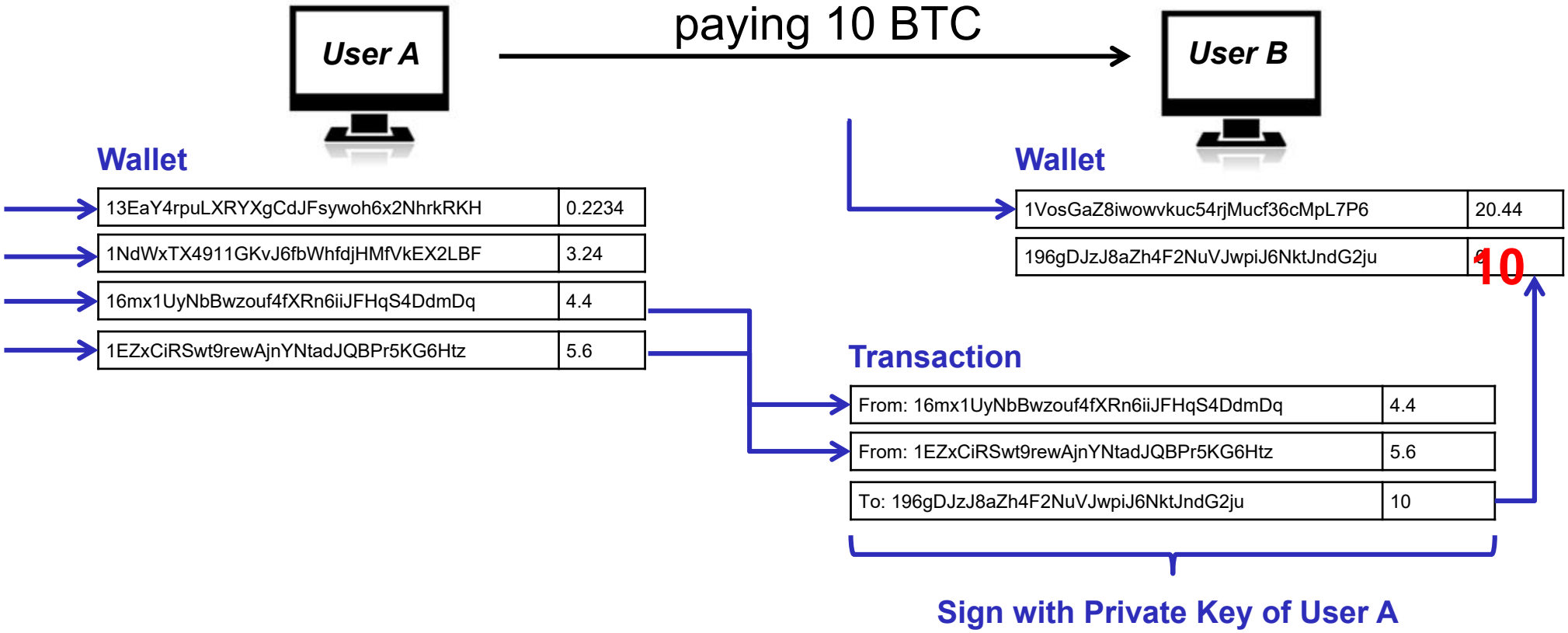
- UTXO (Unspent Transaction Output)
  - Represents a chain of ownership as a chain of digital signatures where the owner signs a message transferring ownership of his UTXO to the receiver's public key
- A wallet holds public-private keys (wallet.dat)
  - Public key, ECDSA 256 bit → Bitcoin address (receives BTC)
  - Private key used for signing transactions

ECDSA: Elliptic Curve Digital Signature Algorithm

- Transaction
  - Peer A to send BTC to peer B → Creates transaction message
  - Transaction contains input/output (BTC from/to)
  - Peer A broadcasts transaction to all peers in the network
  - Transaction stored in blocks → Block is created, verified ~1 min

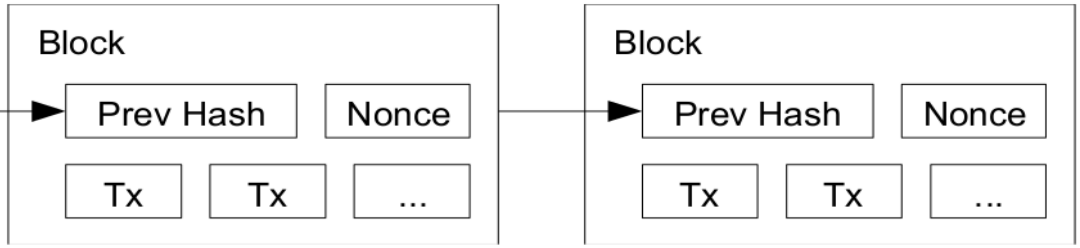
Main Bitcoin Operations

- Private key authorizes the transaction ("access")
  - If keys are stolen, thief may use "your" coins
  - If keys are lost, coins are lost



Double Spending and Coin Generation

- The Bitcoin blockchain avoids double spending by
  - Ensuring that transactions in blocks are confirmed
  - Guessing value that results in number of zero bits is “hard”
    - 00000000000001805ff174586b6acf100f733aaf634e92f9580b4fac9272ed97
  - Chained PoW
- Generation of coins
  - Mining/creating blocks



    - Every time 210,000 blocks are added to the blockchain, mining reward is halved to ensure a steady supply of Bitcoins
      → Miner gets now a 6.25 BTC reward (May 17, 2020, at 11.14 UTC)
    - The reward decreases to 3.125 BTC expected in 2024, ends 2140 (sic!)
    - Reward is halved → half the inflation → lower available supply → higher demand → higher price → miners' incentive still remains, regardless of smaller rewards, as the value of Bitcoin is increased in the process

Bitcoin Protocol

- Tx in detail

version		01 00 00 00
input count		01
input	previous output hash (reversed)	48 4d 40 d4 5b 9e a0 d6 52 fc a8 25 8a b7 ca a4 25 41 eb 52 97 58 57 f9 6f b5 0c d7 32 c8 b4 81
	previous output index	00 00 00 00
	script length	8a
	scriptSig	47 30 44 02 20 2c b2 65 bf 10 70 7b f4 93 46 c3 51 5d d3 d1 6f c4 54 61 8c 58 ec 0a 0f f4 48 a6 76 c5 4f f7 13 02 20 6c 66 24 d7 62 a1 fc ef 46 18 28 4e ad 8f 08 67 8a c0 5b 13 c8 42 35 f1 65 4e 6a d1 68 23 3e 82 01 41 04 14 e3 01 b2 32 8f 17 44 2c 0b 83 10 d7 87 bf 3d 8a 40 4c fb d0 70 4f 13 5b 6a d4 b2 d3 ee 75 13 10 f9 81 92 6e 53 a6 e8 c3 9b d7 d3 fe fd 57 6c 54 3c ce 49 3c ba c0 63 88 f2 65 1d 1a ac bf cd
	sequence	ff ff ff ff
output count		01
output	value	62 64 01 00 00 00 00 00
	script length	19
	scriptPubKey	76 a9 14 c8 e9 09 96 c7 c6 08 0e e0 62 84 60 0c 68 4e d9 04 d1 4c 5c 88 ac
block lock time		00 00 00 00

http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html

Bitcoin Script ("Smart Contract")

- Non-Turing complete (*e.g.*, no loops)
  - Very "limited" Smart Contract (SC) language
- With scripts
  - Multisig, n-of-m, escrow, and dispute mediation
  - Micropayment channel, refund tx in future
- Opcodes
  - Data operations: OP_PUSHDATA1, OP_PUSHDATA4,...
  - Flow control: OP_IF, OP_ELSE, ...
  - Stack: OP_DUP, OP_SWAP, ...
  - Arithmetic: OP_ADD, OP_ABS, ...
  - Crypto: OP_SHA256, OP_CHECKSIGVERIFY

Bitcoin Scripting Language (Examples)

- ScriptSig

  Part of a transaction, which contains the required signatures and the script, which unlocks a UTXO for spending.

  PUSHDATA

  signature data and SIGHASH_ALL

  PUSHDATA

  public key data

- ScriptPubKey

  Script which controls how bitcoin can be spent. Most bitcoin scripts translate as follows: "In order to spend this bitcoin, one must produce a signature belonging to this public key".

  OP_DUP

  OP_HASH160

  PUSHDATA

  Bitcoin address (public key hash)

  OP_EQUALVERIFY

  OP_CHECKSIG

Bitcoin Wallets – Off-line/Paper

- For long-term storage of Bitcoin (or giving as gifts) it is "not safe" to store Bitcoin in an exchange or online wallet
  - Generate new address(es)
    - Private key and QR code
  - Printing own tamper-resistant Bitcoin wallets
    - Optional: tamper-evident serialized hologram stickers
  - Example: Bitcoin Cash



  - No crypto-code is run on the Web server
  - No addresses are transmitted over the Internet

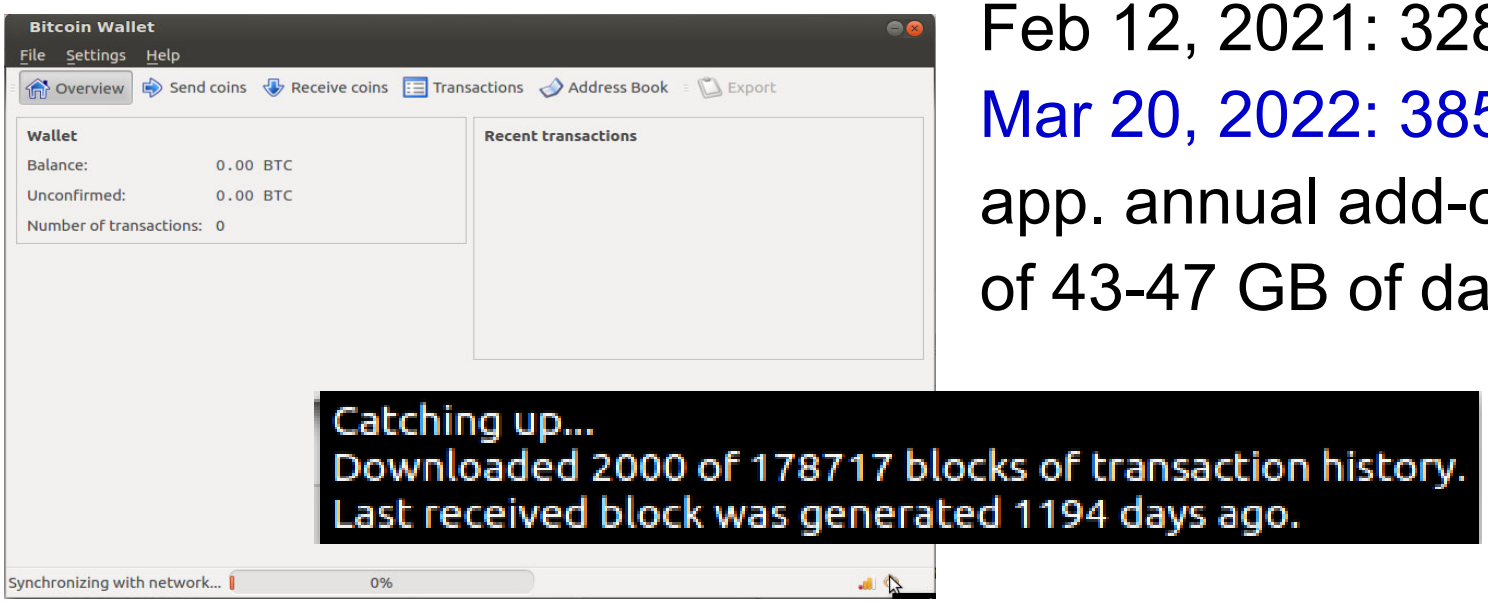
https://bitcoinpaperwallet.com/

Anonymity and Pseudonymity

- Reasonable anonymity with Bitcoin is complicated and perfect anonymity may be impossible
  - Sending and receiving Bitcoins is pseudonymous!
- Wallet deficits – as an example concern
  - Weak privacy aspects due to the pseudonym “Bitcoin Address”
  - Nodes can know almost exactly which addresses constitute accounts
    - Bitcoin users use a new address for each transaction to avoid the transactions being linked to a common owner
    - Best practice, but not sufficient due to multi-input transactions
- Committed bloom filters and heuristics for improved wallet performance and Simplified Payment Verification (SPV) security

Bitcoin Software and BC Size Over Time

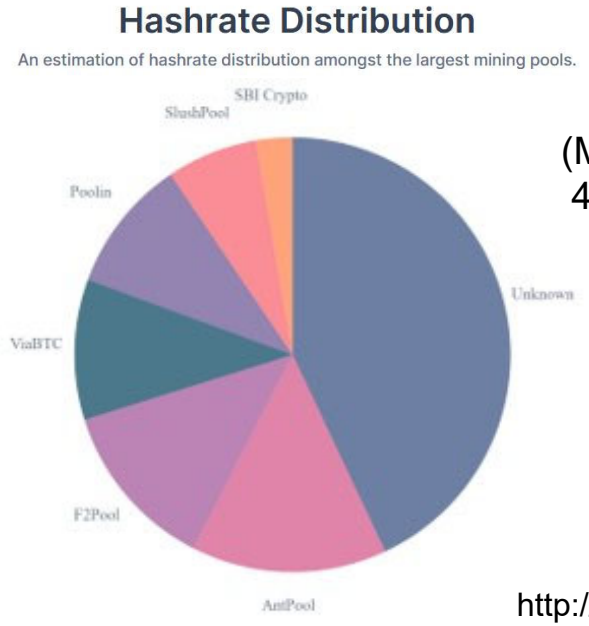
- Bitcoin is also the name of the software
  - 2012: app. 2 hours and 1.8 G less disk space later, full bitcoin BC was downloaded
  - 2013: 8 GB, 2014: 19 GB, 2015: 36 GB disk space
  - 2016: 71 GB, 2017: 120 GB, 2020: 280 GB disk space, Feb 12, 2021: 328.2 GB, Mar 20, 2022: 385.7 GB, app. annual add-on of 43-47 GB of data



Bitcoin Mining (1)

- Sets of "big" miners
  - Miners specialized, AMD GPUs, FPGA, ASIC (Application-specific Integrated Circuit)

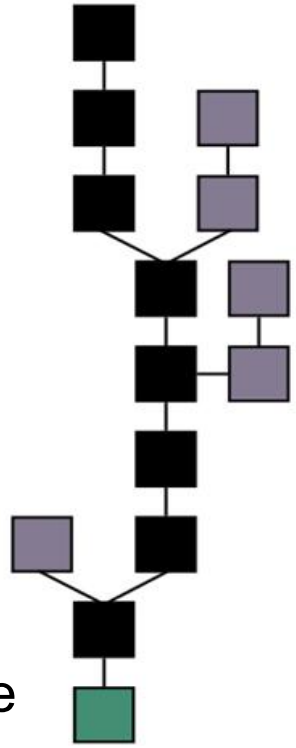
https://en.bitcoin.it/wiki/Mining_hardware_comparison



(March 28, 2020)
4 Days Average

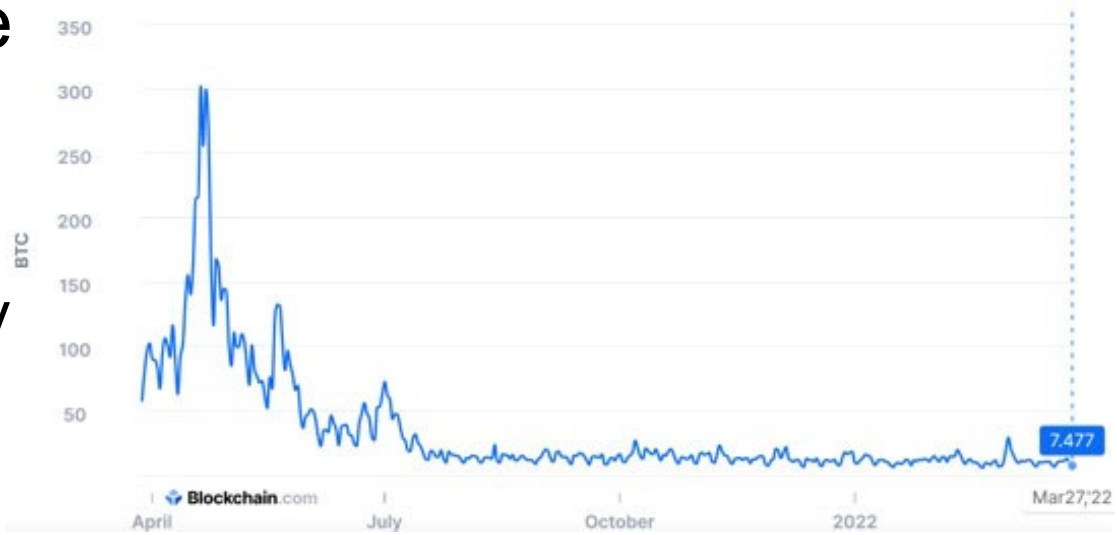
http://blockchain.info/pools

- Reminder
- Mining = creating valid blocks
- Blocks are linked to previous blocks
  - Longest block survive (most difficult)
- Different level of confirmations
  - 6 block confirmation is considered secure

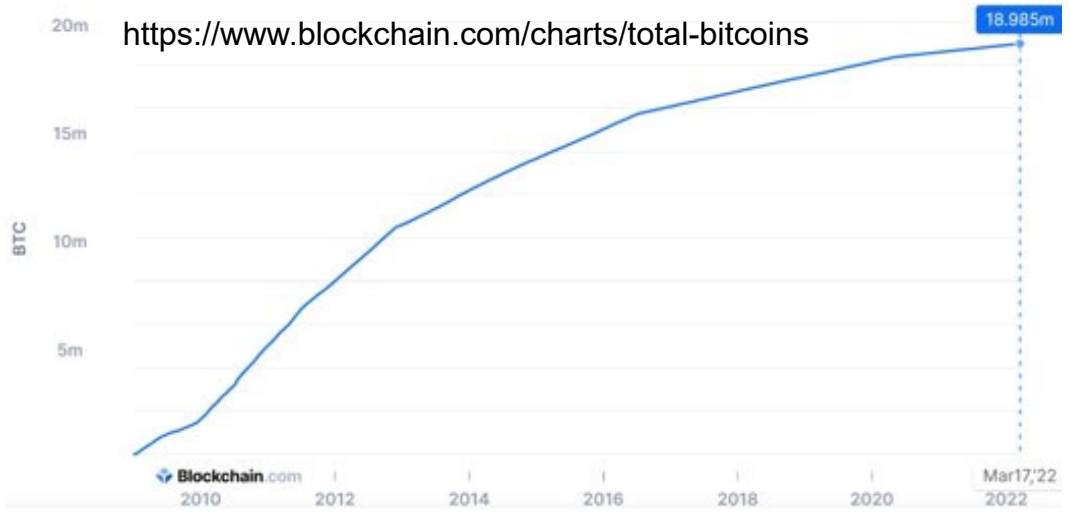


Bitcoin Mining (2)

- Dangerous, if someone has more than 51% computing power
  - Can exclude and modify ordering of transactions
- Miner receives also tx fees
  - Total tx fees in Bitcoin:
    - Total BTC value of all tx fees paid to miners
- Total amount of BTC created (absolute values)



https://www.blockchain.com/charts/total-bitcoins



© 2022 UZH, CSG@IfI

Mining Evolution (1)

- CPU (Central Processing Unit) ≈ **100 M hashes/s**



https://99bitcoins.com/20-insane-bitcoin-mining-rigs/

Mining Evolution (2)

- GPU (Graphics Processing Unit) ≈ **1 G hashes/s**



https://bitcointalk.org/index.php?topic=7216.560

Mining Evolution (3)

- FPGA (Field Programmable Gate Arrays)
  - **≈ 10 G hashes/s**



http://www.openmobilefree.net/?p=1308

Mining Evolution (4)

- Application-specific Integrated Circuit (ASIC) Mining Forms ≈ **1-10 T hashes/s**
- "Life Inside a Secret Chinese Bitcoin Mine"
  - 4,000 BTC per month

https://www.youtube.com/watch?v=K8kua5B5K3I



- KnCMiner

https://www.coindesk.com/kncminer-declares-bankruptcy-cites-upcoming-bitcoin-subsidy-halving

  - Operator of the giant mining facility in Boden, Sweden
  - Maker of mining computers
  - Bankruptcy (May 2016) just after
    - 25 BTC ➔ 12.5 BTC reward changed



Mining Evolution (5)

- Scenario: “New” ASIC miner
  - Example: ANTMINER S
    - ≈ 13 TH/s, 1300 W

https://shop.bitmain.com/promote/antminer_s9i_asic_bitcoin_miner/overview

  - Difficulty increment: 0.5% or 5.5%?







ANTMINER S7	Very low difficulty increment
Difficulty	520808749422
Rate BTC/USD	1300
Electricity	0.09 / 0.18 CHF/kWh
HW Break Even	264 days

Bitcoin Transactions

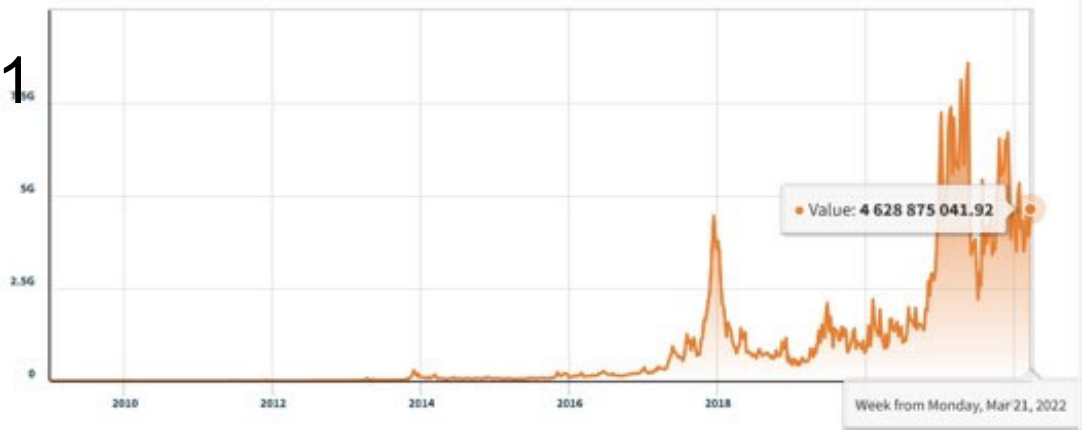
- 218,202 tx on Mar 27, 2022
- Peak 453,530 May 9, 2019
  - 3 tx/s on average and maximum of 7 tx/s

https://ycharts.com/indicators/



- BTC tx volume
  - Volume as of May 17, 2021 ~ 8,573,363,131 US$

https://data.nasdaq.com/data/BCHAIN/ETRVU-bitcoin-estimated-transaction-volume-usd



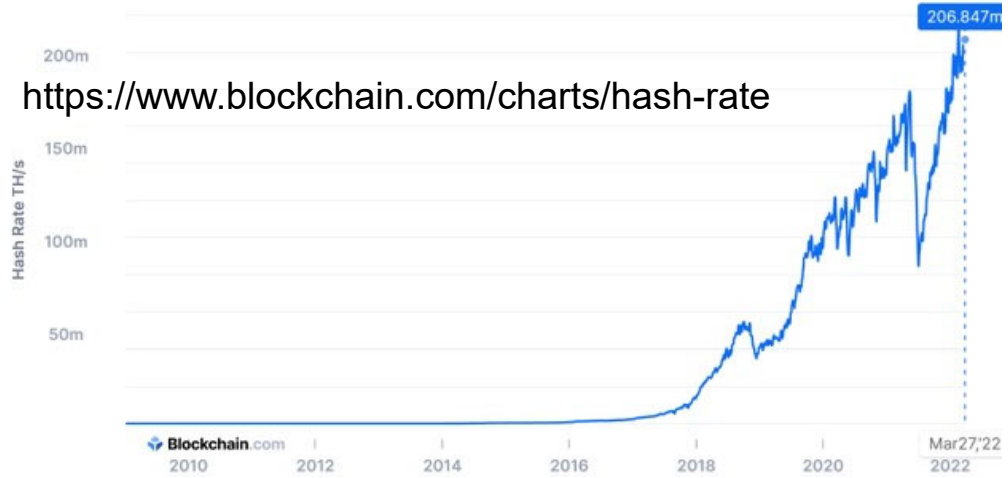
© 2022 UZH, CSG@IfI

Bitcoin Hashrate and Difficulty

- Network Hashrate
  - ~ 47,453,476 PetaFLOPS
  - ~ 16,500,000 in 2016
  - ~ 4,400,000 in 2015
  - ~ 713,756 in 2014
  - ~ 900 in 2013
  - ~ 155 in 2012
- Fastest supercomputer Japanese Fugaku
  - – High Performance Linpack (HPL) benchmark at 442 PetaFLOPS with 7.3 m cores

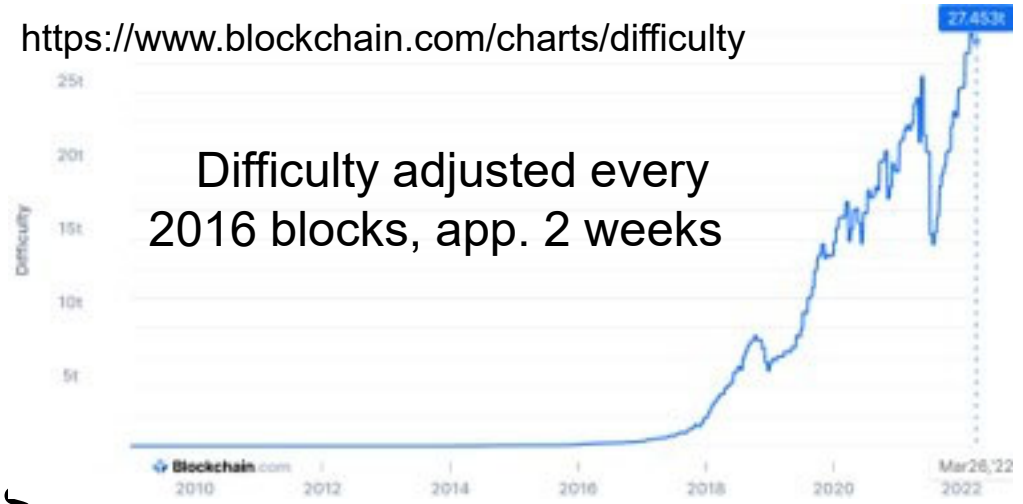
https://top500.org/

https://www.blockchain.com/charts/hash-rate



https://www.blockchain.com/charts/difficulty

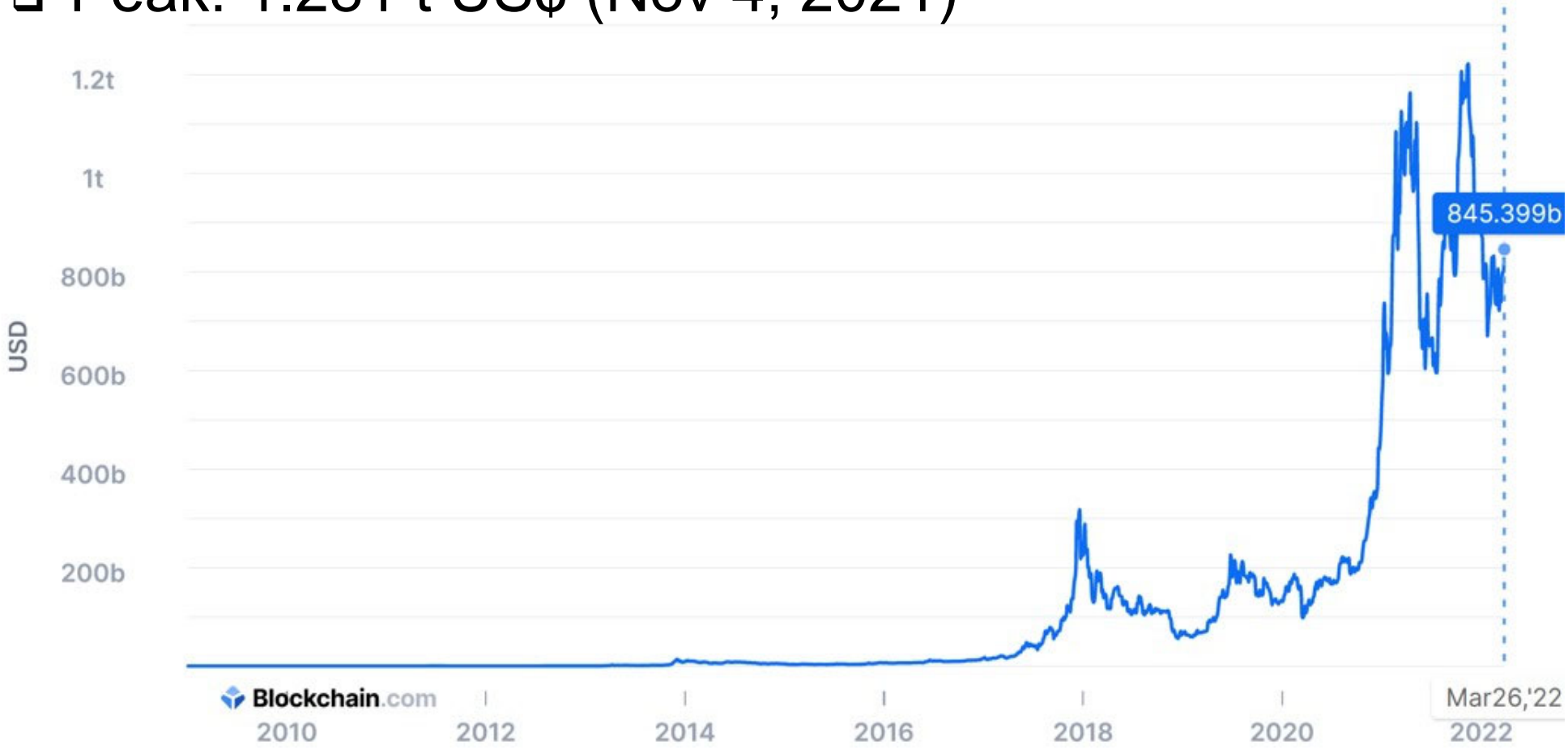
Difficulty adjusted every 2016 blocks, app. 2 weeks



© 2022 UZH, CSG@IfI

Bitcoin‘s Market Capitalization in US$

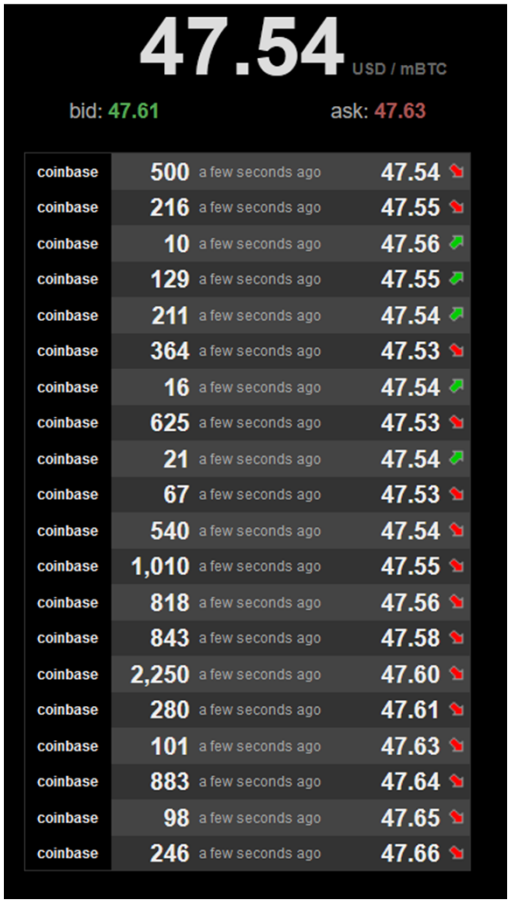
- Peak: 1.281 t US$ (Nov 4, 2021)



https://www.blockchain.com/charts/market-cap

Bitcoin Value and Coins Mined (1)

- 1 BTC
  - ≈ 384.04 US$ (February 11, 16),
  - ≈ 8,127.52 US$ (April 13, 18)
  - ≈ 6,879.44 US$ (April 12, 20)
  - ≈ 29,388.94 US$ (December 31, 20)
  - ≈ 47,914.12 US$ (February 12, 21)
  - ≈ 47,372.20 US$ (March 28, 22)



https://data.bitcoinity.org/markets/

- At an average energy consumption of 135 TWh/year (March 28, 2022)
  - Swiss use in 2019 overall at 58 TWh

https://www.cbeci.org/

Bitcoin Value and Coins Mined (2)

- High variance of value per day

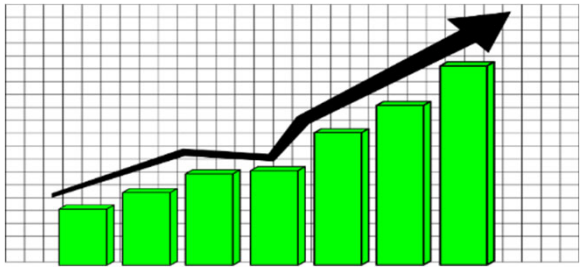
mBTC



https://data.bitcoinity.org/markets/
https://data.bitcoinity.org/markets/

Bitcoin Advantages and Drawbacks (1)

- Advantages
  - Lower (fixed) tx fees
    - 10-30 Satoshi per Byte
  - Scalable
    - Hardware/storage gets faster



  - Anonymity
    - Pseudonymity in reality
    - No privacy concerns (?)
    - Data mining difficult (context?)

- Disadvantages
  - Power consumption
    - Twice as much ++ as entire Swiss use in 2019 alone
    - Not sustainable
  - Not scalable
    - Bitcoin with 7 tps vs. VISA 57,000 tps (typical December) [tps: transactions per sec]
  - Anonymity
    - Can be used for illegal activities

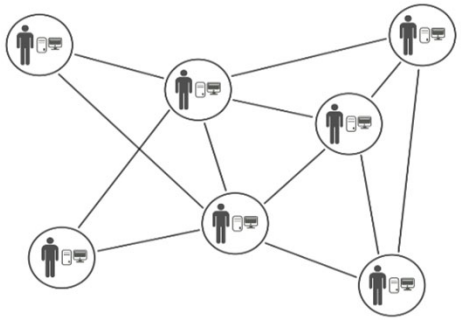


Bitcoin Advantages and Drawbacks (2)

- Advantages
  - No major “crashes”
    - Mt. Gox was an exchange site!

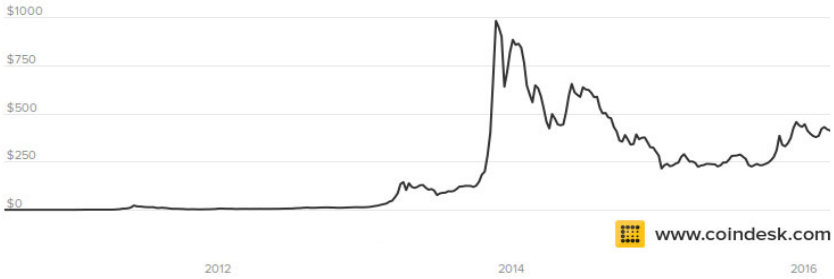


  - Decentralized
    - Open protocol



  - Other blockchain usage
    (Beyond Bitcoins, of course)
    - Smart Contracts
    - Many application areas

- Disadvantages
  - Volatile exchange rate



  - Central elements
    - 5 core PGP fingerprints at GitHub‘s “Bitcoin Core“ (= developers? Not really …)



https://blog.lopp.net/who-controls-bitcoin-core-/

Ethereum, Smart Contracts, and Solidity



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Characteristics and Fundamentals
  - General Purpose BC
  - Node Types
  - Standards and Roadmap
- Technical Details
- Statistics
- EVM, Implementations, and Smart Contracts
- Solidity and Example SC
- Token Standards
- Summary

Ethereum Characteristics and Fundamentals

Ethereum Characteristics (1)

- Ethereum is a general purpose blockchain (BC)



  - Major goal: Perception of a BC as a platform
    - Supporting the execution of Smart Contracts (SC)
      - Programmable, complex, Turing-complete
    - Empowering the BC beyond FinTec applications (“decoupled”)
      - Key technical enabler for dApps
      - Smart Contracts enabling complex rules to be decentralized
  - Ethereum foundation located in Zug, Switzerland
    - ETH (Ether) Crowdsale of app. 50 million ETH https://icodrops.com/ethereum/
      - At that time at 0.31 US$ per ETH coin, comparable ~ 32,000 BTC
      - White paper released in December 2013
      - Protocols designed from bottom-up (not like LiteCoin or PeerCoin)

Ethereum Characteristics (2)

- Crypto-coin Ether (ETH) = 10^{18} Wei (quintillion US or trillion UK)
- Ethereum prototype and two initial versions applied PoW
  - Current version with migration to PoS, future versions only PoS (?)
- Mining reward at approximately every block (~ 14 s)
  - Always 5 ETH, inflation concerns do not apply
    - If ETH are considered to define the gas price and not a cryptographic currency

- However, *with great power comes great responsibility*
  - Allowing for more complex logics implies on applications, especially dApps, that these are …
    - … more prone to errors – humans' "program"
    - … subject to security flaws – systems and platforms "deficits"
    - … a potential target of (external/internal) attacks

Ethereum Characteristics (3)

- Ethereum uses the account-based model
  - Keeps track of the balance as a global state
    - Balance should be larger or equal than spending amount
  - Simpler than Bitcoin's UTXO



Bitcoin (reminder)

- Balance is checked by summing up the amount of bills (UTXO) in the wallet
- When one wants to spend money, it is possible to use one or more bills (UTXOs)



Ethereum

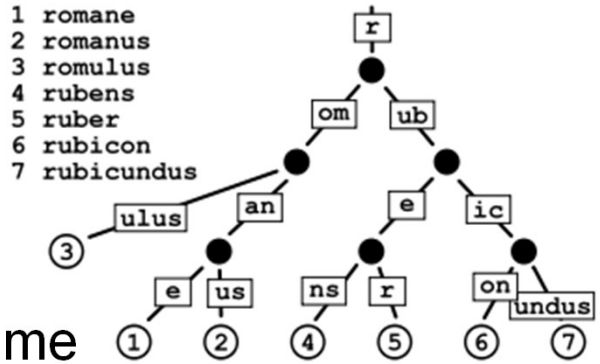
- Simpler
- Like debit card, spending possible only with a balance > 0
- Global state stored in specialized "Patricia Trie"

Ethereum Characteristics (4)

- Ethereum's account-based states are saved as Patricia Merkle Tries (PMT)
  - States not stored directly on the Blockchain → only root hash
  - To prevent attacks, nonce required
- Two account types:
  a) Externally controlled
  b) Contract account
  - Both types operate on Ether (ETH)
- Contracts only execute a tx in response to another tx
  - All action in Ethereum originates from a tx sent from externally controlled accounts

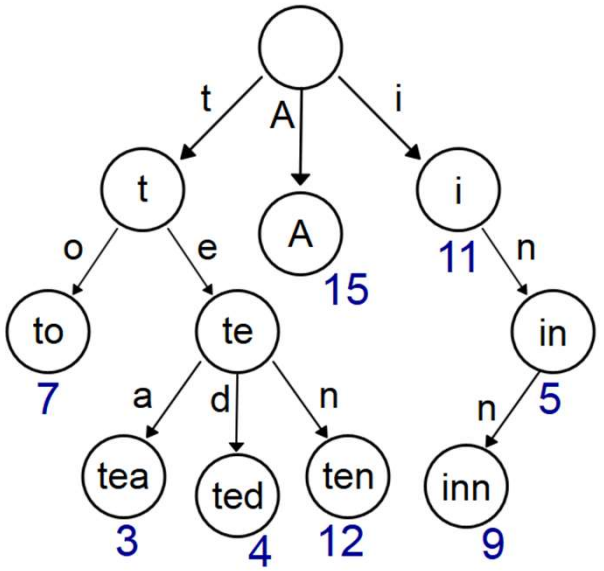
Patricia Tries and Patricia Merkle Tries

https://de.wikipedia.org/wiki/Patricia-Trie



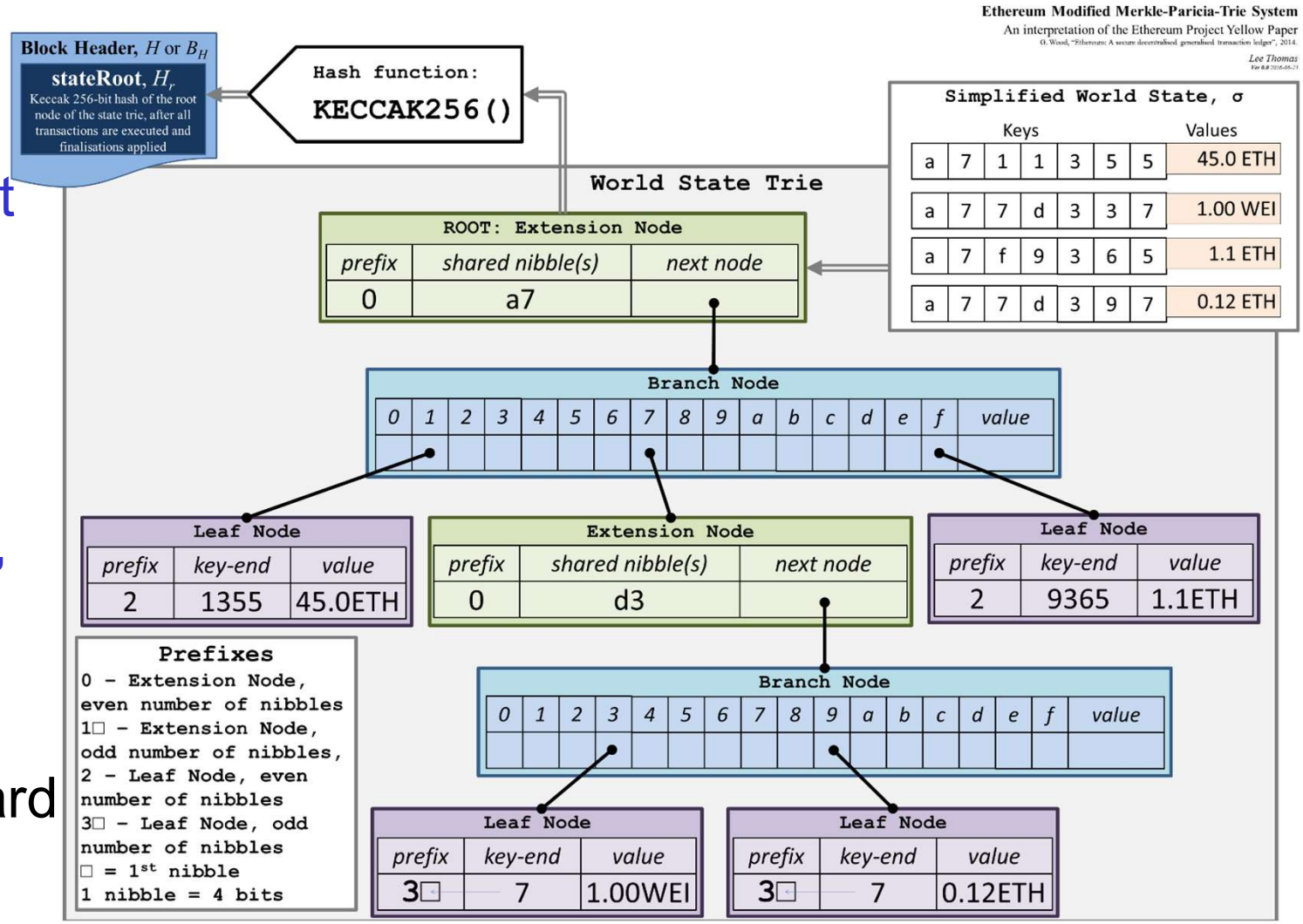
- Patricia Trie
  - Ordered data structure
    (trie = tree of prefixes, here of “words”)
    - Stores multiple character strings at the same time
      - Dynamic set or associative array, where keys are usually strings
    - “*Practical Algorithm to Retrieve Information Coded in Alphanumeric*“

- Ethereum uses modified
  Patricia Merkle Tries (PMT)
  - To cryptographically authenticate data
  - To store all (key, value) bindings
  - Being fully deterministic



Ethereum's Modified PMT

- Efficient to verify whether a hash is correct
- Patricia Merkle Tree with O(log(n)) for inserts, lookups, and deletes compared to O(n) of a standard Merkle Tree



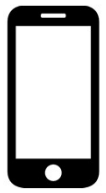
Ethereum Types of Nodes



- An Ethereum node is a device/program that communicates with the Ethereum network
  - Nodes are known as clients, *e.g.*, Geth, Parity
- Full client
  - EVM, verification of transactions, executing of SCs, programs, operations
  - Full nodes that preserve the entire blockchain history
    - Archives the full blockchain: “archive node”
- Light client
  - Relies on a full node
  - Neither verify transactions nor hold a copy of the current Ethereum’s blockchain state

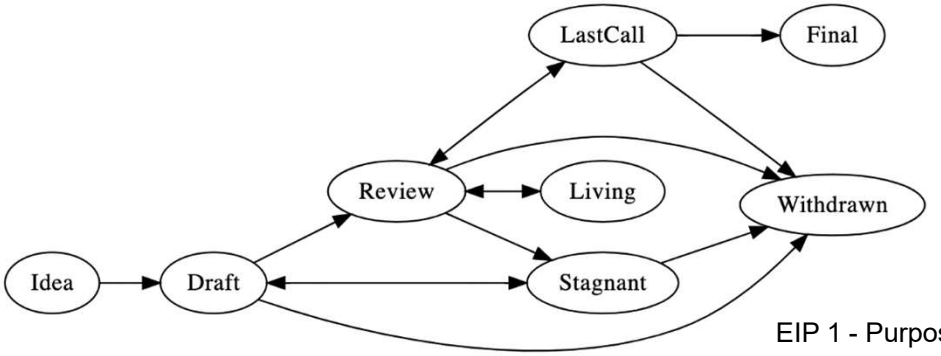
SC: Smart Contract
EVM: Ethereum Virtual Machine





Ethereum Standards (1)

- Ethereum is focused on standardizing its development
  - Interoperable across implementations
  - Consensus reached with the community (standard process)



EIP 1 - Purpose and Guidelines

- Ethereum Improvement Proposals (EIP)
  - Standard for potential new features
  - Contain technical details of the feature
  - Anyone is free to propose a new EIP

https://eips.ethereum.org/

Ethereum Standards (2)

- EIP in final state → released in a future hard fork
- EIP examples:
  - **EIP-145:** Bitwise shifting instructions in EVM (2017)
    - SHL and SHR cost 3 gas (35 with normal arithmetic operators)
  - **EIP-1234:** Constantinople Difficulty Bomb Delay and Block Reward Adjustment (2018)
    - Relaxes the difficulty so that miners are able to mine blocks in 15 s
  - **EIP-2982 -** Serenity Phase 0
    - Phase 0 of Serenity (eth2) → Move from PoW to PoS
- If an EIP relates to application-level standards
  - Called Ethereum Requests for Comment (ERC)
  - *E.g.,* **ERC-20 Token Standard** and **ERC 721 NFT Standard**

SHL: Shift Left
SHR: Shift Right

NFT: Non-Fungible Token

Ethereum Release Roadmap

- Frontier (2015)
  - First release based on Proof-of-Work (PoW)
  - Highly experimental
- Homestead (2016)
  - Several improvements and bug fixes
  - Many planned and unplanned forks (*e.g.*, DAO hack)

Vulnerable SC function attacker got 3.6m ether (~ $70 million USD). Caused the fork on Ethereum and Ethereum classic.

- Metropolis (2017 – current)
  - Also included improvements and bug fixes
  - Two planned forks (Byzantium and Constantinople)
    - Constantinople will be the first trial based on PoS (hybrid with PoW)
- Serenity (planned for 2022 – originally for Dec 2020)
  - Planned to be fully based on PoS

Selected Ethereum Hard Forks

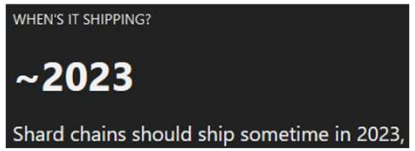
https://ethereum.org/en/history/

Release	Name	Planned	Date	Block Height	Consensus	Description
Frontier	Frontier	Yes	30.07.2015	1	PoW	First release
Frontier	Frontier Thawig	Yes	08.09.2015	20,000	PoW	Bug fixes (no EIP structure yet)
Homestead	Homestead	Yes	14.03.2016	1,150,000	PoW	EIP 2, 7, 8
Homestead	DAO Fork	No	20.07.2016	1,920,000	PoW	SC security fixes
Homestead	EIP-150 Hard Fork	No	18.10.2016	2,463,000	PoW	EIP 150, 158
Homestead	Spurious Dragon	No	22.11.2016	2,675,000	PoW	EIP 155, 160, 161, 170
Metropolis	Byzantium	Yes	16.10.2017	4,370,000	PoW PoS Preparation	EIP 140, 658, 196, 197, 198, 211, 214, 100, 649
Metropolis	Constantinople	Yes	28.02.2019	7,280,000	PoW→PoS	EIP 145, 1014, 1052, 1234
Metropolis	Instanbul	Yes	08.12.2019	9,069,000	PoW→PoS	EIP 152, 1108, 1344, 1844, 2028, 2200
Metropolis	Muir Glacier	No	02.01.2020	9,200,000	PoW→PoS	EIP-2384 (Delayed difficulty bomb)
Metropolis	London	Yes	05.09.2021	12,965,000	PoW→PoS	Fee changes
Metropolis	Arrow Glacier	No	09.12.2021	13,773,000	PoW→PoS	Delayed dif. bomb
Serenity	N/A	Yes	2022?	N/A	PoS	ETH 2.0

Ethereum Roadmap for Serenity – Eth 2.0

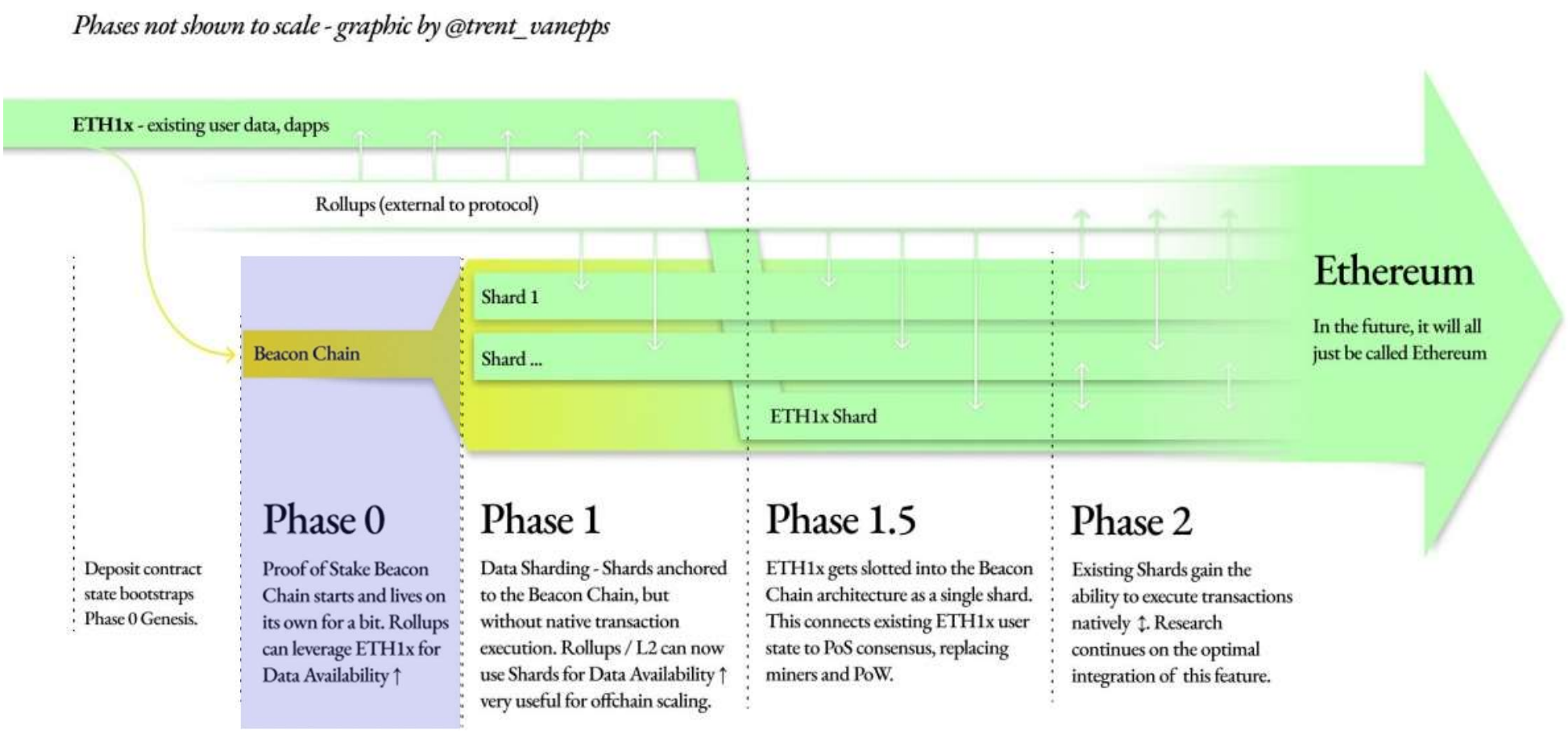
- Serenity: Moving completely from PoW to PoS (plan)
- Roadmap divided into 4 phases
  - Phase 0 – Beacon Chain (Dec 2020) ✓
  - Phase 1 – Shard Chains (Expected 2021) ⚠
    - Phase 1.5 - Docking mainnet with Eth2 (end of PoW)
    - Expected ~ 2013
  - Phase 2 – eWASM (new Ethereum Virtual Machine) ✗
  - Phase 3 – Continued Improvements ✗
- Both mainnet and Beacon will co-exist until phase 1.5
- Eth 2.0 improves
  - Security, scalability, and performance

https://ethereum.org/en/upgrades/shard-chains/



https://consensys.net/blog/blockchain-explained/the-roadmap-to-serenity-2/

Relation of Ethereum (Mainnet) and Eth 2.0



Current Status

https://www.coindesk.com/ethereum-2-0-beacon-chain-goes-live-as-world-computer-begins-long-awaited-overhaul

Beacon Chain (1)

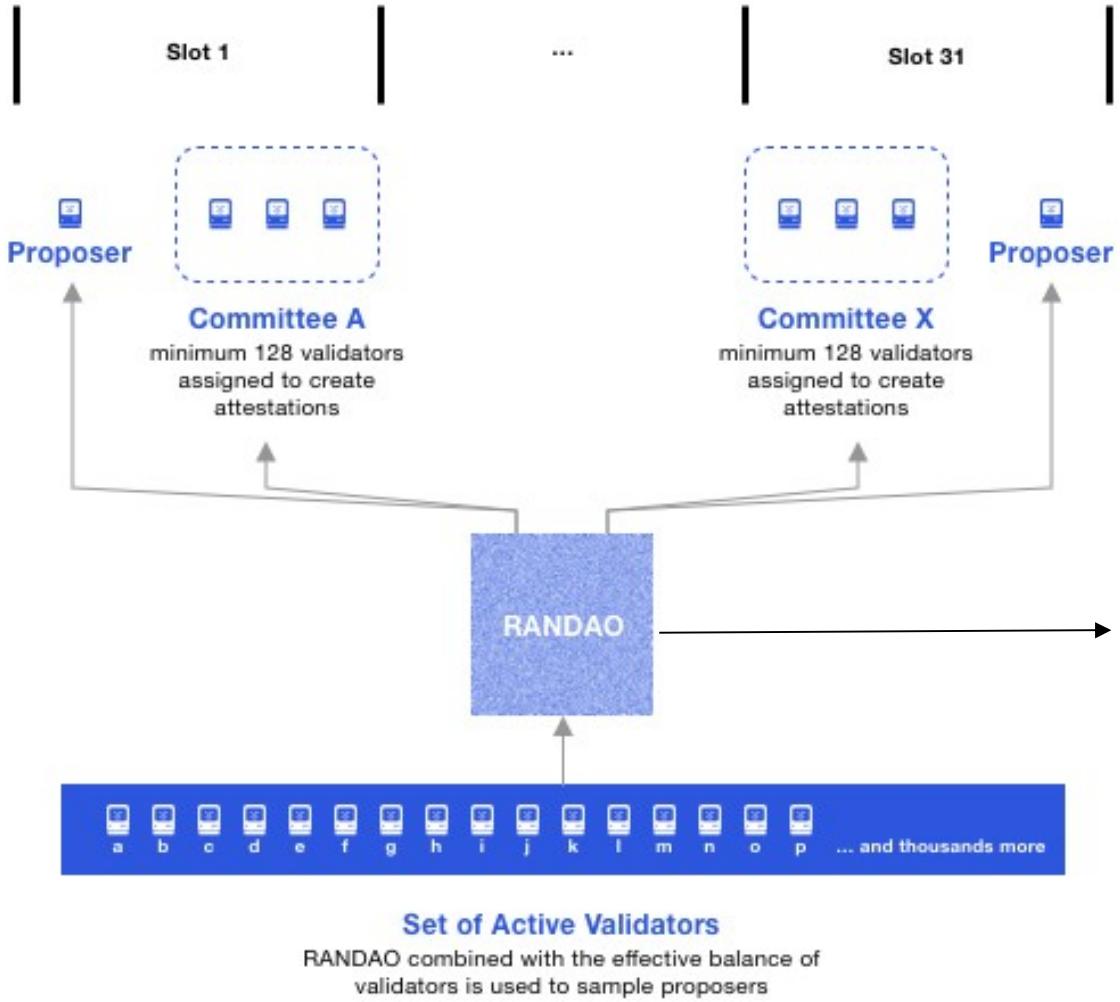
- Introduces PoS to Ethereum
  - Does not change the PoW Ethereum BC (mainnet)
- Coordinates shards and stakers
- Genesis block created on December 1, 2020
  - Required 16,384 deposits of 32 staked ETH
  - Guarantee that the validator node is honest and operational
- Incentives and penalization
  - Propose blocks, attest block, and “whistleblow” validators
  - Inactivity, double voting, or attest different versions
- Less energy and hardware demanding than PoW
  - Validator node can run on a Raspberry PI 4 (US$ 35)

https://beaconscan.com/slot/1

Beacon Chain (2)

- Organized into slots and epochs
  - Slot: interval of 12 s to a validator to propose a block
  - Epoch: group of 32 slots every 6.4 min
- Each slot:
  - Random validator is selected to propose a block
    - Validator's stake impact on the selection
  - Group of random validators is selected to validate (attest) it
    - Committee composed of minimum of 128 validators
- New validators
  - Must deposit (stake) 32 ETH in a SC in the mainnet
  - Validator must be online > 50% of the time in order to have positive returns

Committee Forming



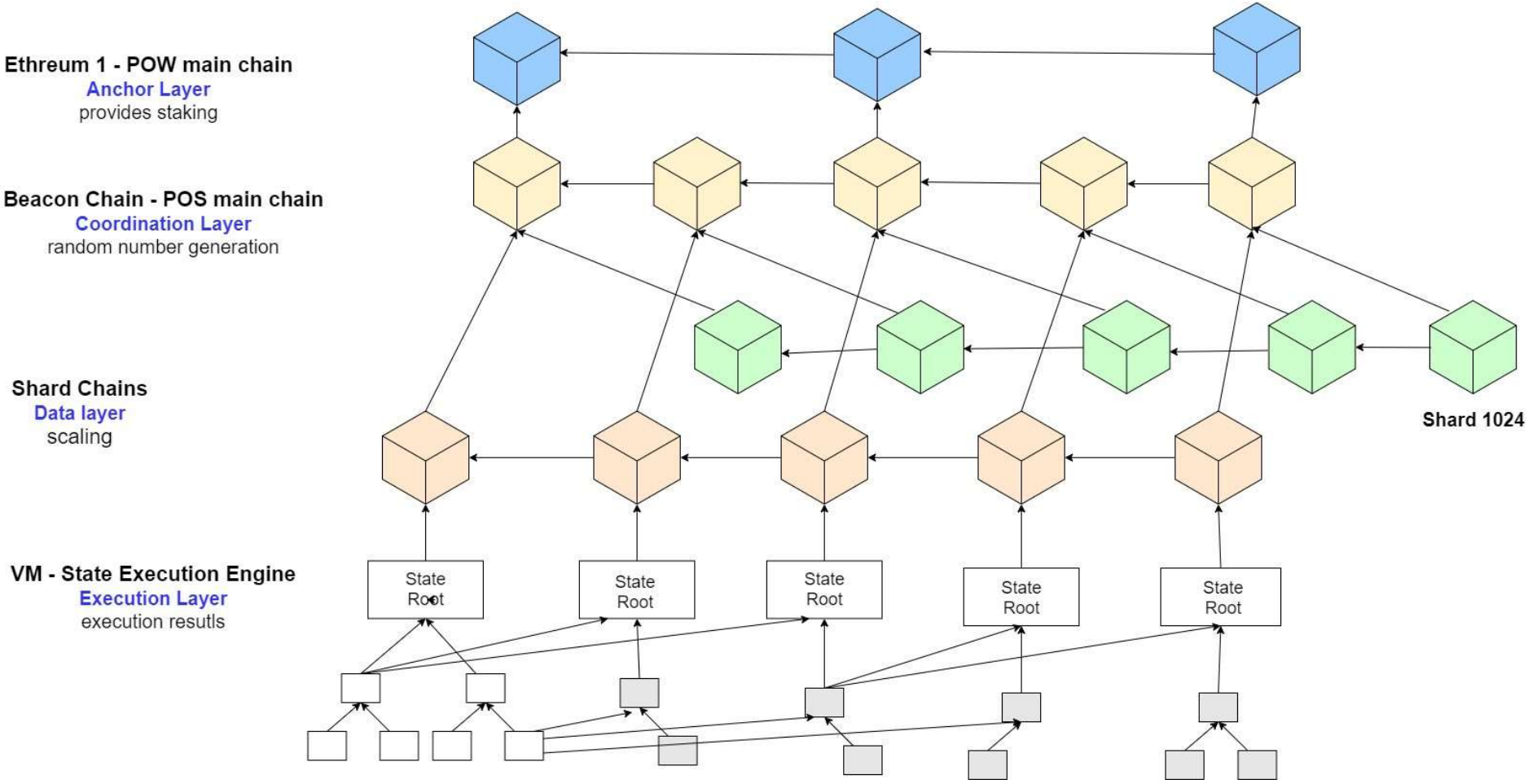
Decentralized Random Number Generator (RNG) based on a commit-reveal approach

https://ethos.dev/beacon-chain/
https://github.com/randao/randao

Shard Chains

- Splitting the BC in new chains (*i.e.*, shards)
  - Split occurs horizontally → reduces network congestion and increases transactions per second (tps) → more scalability
- First shard chains will not handle SC or transactions
  - Provide only rollup functions (*i.e.*, aggregating transactions)
  - Reducing data required for a transaction → up to 100,000 tps
- Community still debating on SC code execution feature
- Beacon chain provides:
  - Communication between shards
  - Synchronization and coordination
- Ethereum mainnet will become a shard in the future

Mainnet, Beacon Chain, Shard Chains



https://medium.com/the-capital/ethereum-2-0-complete-guide-deb1adadbf3d

Ethereum Technical Details

Ethereum's Keccak-256

- Ethereum uses the Keccak 256 hash function for
  - Consensus mechanism (Ethash), block hashing, PMT, and
  - Transaction hash, address generation
- Keccak-256 ≠ SHA3
  - Keccak is the winning algorithm for the SHA-3 implementation competition in 2007 NIST: National Institute of Standards and Technology
  - SHA3 is the NIST standardization (2015) of Keccak-256
- Algorithms produce different hash values!
  - Keccak-256("BCOLN") = 368872a48f382…60883ff205b07e0
  - SHA3("BCOLN") = 01dd15fece345e9…9cfc8afa4017fa374d

https://csrc.nist.gov/publications/detail/fips/202/final
https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/ch04.html

Ethereum Addresses Generation

- Ethereum addresses are hexadecimal numbers
  - Derived from the last 40 characters of the Keccak-256 hash of the public key



- Ethereum address can be encoded as ICAP
  - Partly compatible with IBAN
  - ICAP: XE60 HAMI CDXS V5QX VJA7 TJW4 7Q9C HWKJ D
  - IBAN: CH56 0483 5012 3456 7800 9
  - XE = Ethereum
  - CH = Switzerland

ICAP: Inter exchange Client Address Protocol
IBAN: International Bank Account Number

© 2022 UZH, CSG@IfI

Smart Contract Addresses

- Smart Contract addresses are not random!
- They are deterministically calculated using:
  - `CREATE` OPCode (default)
  - The "sender" address and the "nonce"
    - Nonce equals to number of transactions sent by the address
- **EIP-1014**: Skinny CREATE2 (2018)
  - Allow generation of addresses without deployment
  - Address created using:
    - 0xFF, a constant that prevents collisions with CREATE
    - Sender address
    - Salt (arbitrary value)
    - Contract's Bytecode

Not relying on nonce!

Units in Ethereum

- wei → Smallest unit
  - Gas price value (or cost) expressed in "*Gwei*"
- Ether → Largest unit
- Names following famous figures in Computer Science

Unit	Name	Wei Value	Wei	Long Style US
wei	**-**	**1 wei**	**1**	**Unit**
Kwei	Babbage	1e3 wei	1 000	Thousand
Mwei	Lovelace	1e6 wei	1 000 000	Million
Gwei	Shannon	1e9 wei	1 000 000 000	Billion
Microether	Szabo	1e12 wei	1 000 000 000 000	Trillion
Milliether	Finney	1e15 wei	1 000 000 000 000 000	Quadrillion
Ether	**-**	**1e18 wei**	**1 000 000 000 000 000 000**	**Quintillion**

Ethereum Transactions (tx)

- A transaction tx is a single cryptographically-signed instruction constructed by an actor externally to the scope of Ethereum
  - nonce: keep track of the total number of transactions that account has executed
  - gasPrice: price per unit of gas
  - gasLimit: maximum gas
  - to: target address
  - value: total of Ethers to send
  - data: “Hello World”
    - Contract calls are encoded in this field

```
txnCount = web3.eth.getTransactionCount(web3.eth.accounts[0])
var rawTxn = {
    nonce: web3.toHex(txnCount),
    gasPrice: web3.toHex(100000000000),
    gasLimit: web3.toHex(140000),
    to: '0x633296baebc20f33ac2e1c1b105d7cd1f6a0718b',
    value: web3.toHex(0),
    data:
'0xcc9ab24952616d6100000000000000000000000000000000000000000000000000000000000000000000000000000000'
};
```

Higher gas price leads to faster inclusion (more reward for miners)

A Note on Gas Price and Fees

- On August 5, 2021
  - London Fork was introduced changing the tx fee calculation
  - Implemented EIP-1559 – **Fee Market Change for ETH 1**

https://eips.ethereum.org/EIPS/eip-1559

- Old fee scheme
  - `gasPrice` determines the "priority" of the tx
- EIP-1559
  - `Base Fee` must be paid (determined per block) **Burned**
  - `Priority Fee` determines the "priority" of the tx **Miners receive**
- `Base Fee` Adjustment
  - Based on network congestion (*i.e.*, gas usage in block)
  - If target per-block gas > threshold → fee increases slightly
  - If capacity < threshold → fee it decreases slightly

Block Structure (1)

Blockchain Struct

```
type BlockChain struct {
    config *params.ChainConfig // chain & network configuration

    hc            *HeaderChain
    chainDb       ethdb.Database
    rmLogsFeed    event.Feed
    chainFeed     event.Feed
    chainSideFeed event.Feed
    chainHeadFeed event.Feed
    logsFeed      event.Feed
    scope         event.SubscriptionScope
    genesisBlock  *types.Block

    mu      sync.RWMutex // global mutex for locking chain operations
    chainmu sync.RWMutex // blockchain insertion lock
    procmu  sync.RWMutex // block processor lock

    checkpoint       int          // checkpoint counts towards the new checkpoint
    currentBlock     *types.Block // Current head of the block chain
    currentFastBlock *types.Block // Current head of the fast-sync chain (may be above the block chain!)

    stateCache   state.Database // State database to reuse between imports (contains state cache)
    bodyCache    *lru.Cache     // Cache for the most recent block bodies
    bodyRLPCache *lru.Cache     // Cache for the most recent block bodies in RLP encoded format
    blockCache   *lru.Cache     // Cache for the most recent entire blocks
    futureBlocks *lru.Cache     // future blocks are blocks added for later processing

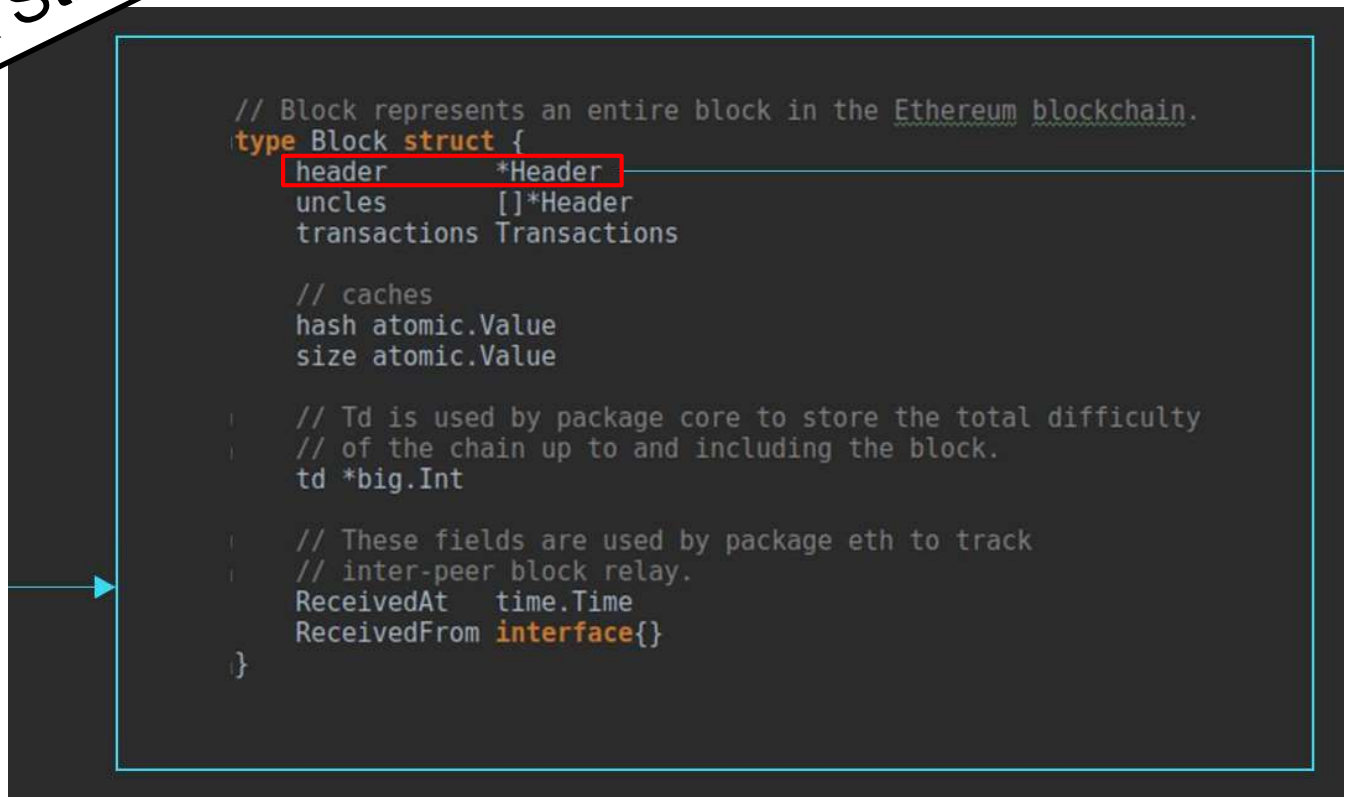
    quit    chan struct{} // blockchain quit channel
    running int32         // running must be called atomically
    // procInterrupt must be atomically called
    procInterrupt int32          // interrupt signaler for block processing
    wg            sync.WaitGroup // chain processing wait group for shutting down

    engine    consensus.Engine
    processor Processor // block processor interface
    validator Validator // block and state validator interface
    vmConfig  vm.Config

    badBlocks *lru.Cache // Bad block cache
}
```

Block Structure (2)

Block Struct



```
// Block represents an entire block in the Ethereum blockchain.
type Block struct {
    header       *Header
    uncles       []*Header
    transactions Transactions

    // caches
    hash atomic.Value
    size atomic.Value

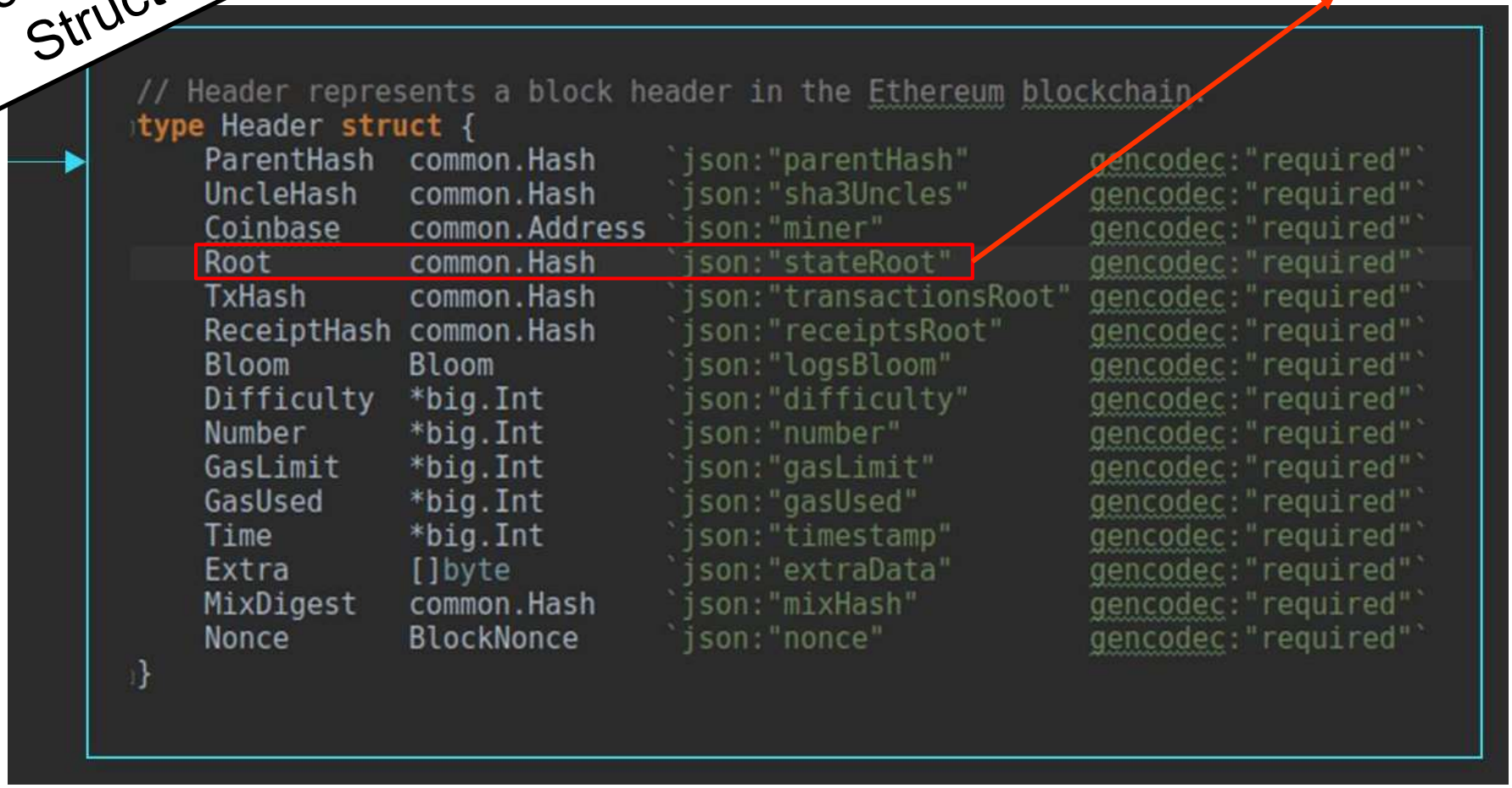
    // Td is used by package core to store the total difficulty
    // of the chain up to and including the block.
    td *big.Int

    // These fields are used by package eth to track
    // inter-peer block relay.
    ReceivedAt   time.Time
    ReceivedFrom interface{}
}
```

Block Structure (3)

Block Header Struct

Hash of the root node of the state trie



```
// Header represents a block header in the Ethereum blockchain.
type Header struct {
    ParentHash  common.Hash    `json:"parentHash"       gencodec:"required"`
    UncleHash   common.Hash    `json:"sha3Uncles"       gencodec:"required"`
    Coinbase    common.Address `json:"miner"            gencodec:"required"`
    Root        common.Hash    `json:"stateRoot"        gencodec:"required"`
    TxHash      common.Hash    `json:"transactionsRoot" gencodec:"required"`
    ReceiptHash common.Hash    `json:"receiptsRoot"     gencodec:"required"`
    Bloom       Bloom          `json:"logsBloom"        gencodec:"required"`
    Difficulty  *big.Int       `json:"difficulty"       gencodec:"required"`
    Number      *big.Int       `json:"number"           gencodec:"required"`
    GasLimit    *big.Int       `json:"gasLimit"         gencodec:"required"`
    GasUsed     *big.Int       `json:"gasUsed"          gencodec:"required"`
    Time        *big.Int       `json:"timestamp"        gencodec:"required"`
    Extra       []byte         `json:"extraData"        gencodec:"required"`
    MixDigest   common.Hash    `json:"mixHash"          gencodec:"required"`
    Nonce       BlockNonce     `json:"nonce"            gencodec:"required"`
}
```

Ethereum Statistics

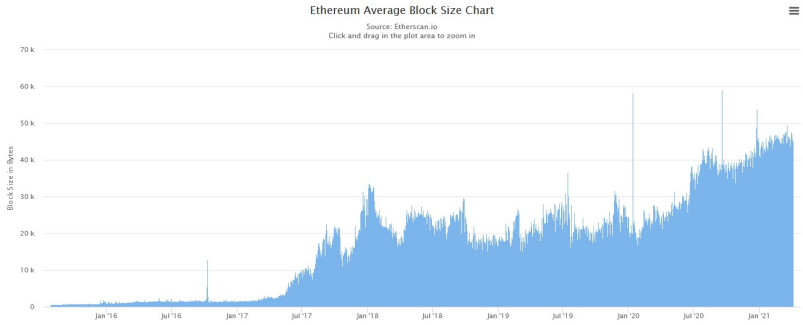
Ethereum Statistics

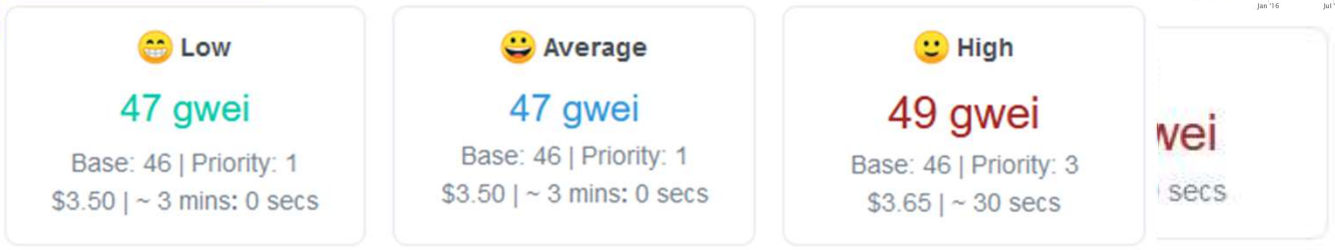
- Basics
  - Price 1 ETH ~ 3,532.57 US$ (as of April 5, 2022)
  - Transactions ~ 1.2 m per day
  - Node count ~ 5,532 nodes https://www.ethernodes.org/
  - Blocksize ~ 80 Kbyte (growing!)
- Top 5 Accounts https://www.etherchain.org/accounts
  - With each above 1.5 m ETH
- Gas price (in GWei)
- Further stats: https://www.etherchain.org/charts

https://coinmarketcap.com/currencies/ethereum/



https://etherscan.io/chart/blocksize





https://etherscan.io/gastracker

Ethereum‘s Market Capitalization [US$] (1)

CoinMarketCap

Top 100 Cryptocurrencies by Market Capitalization

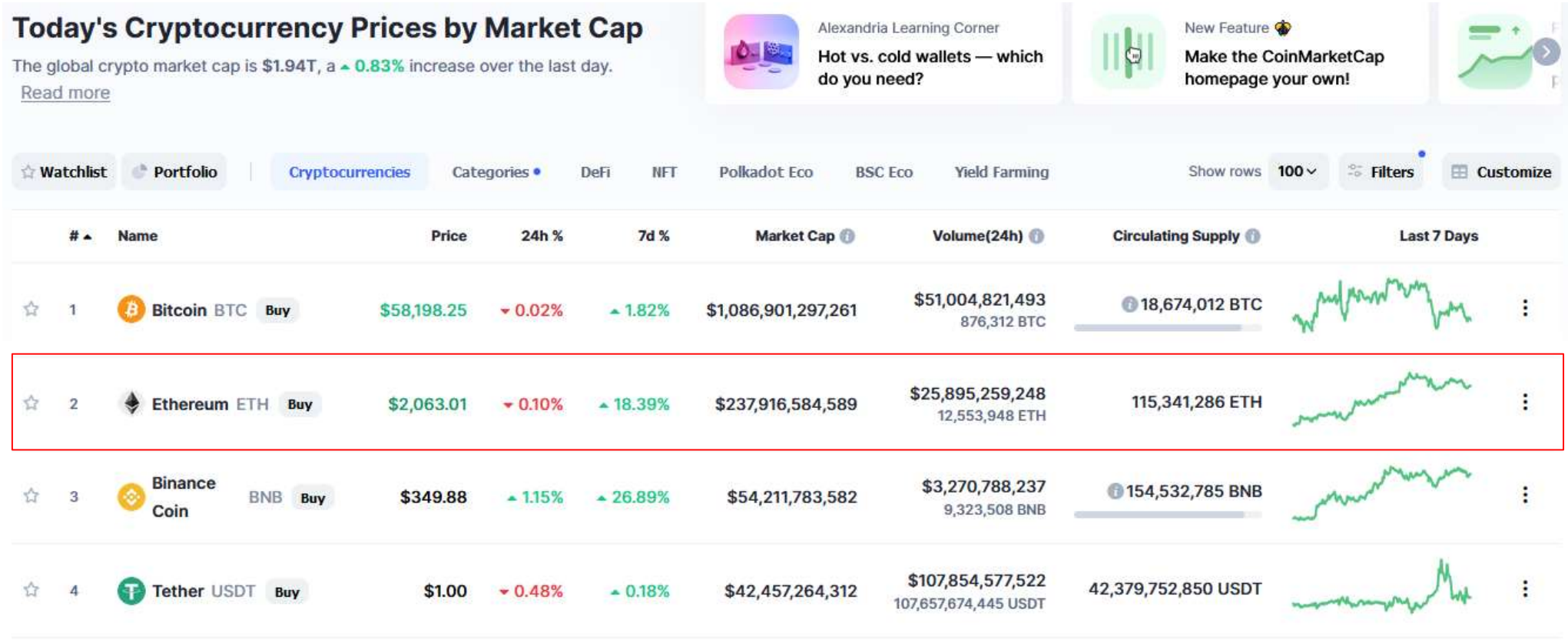
Cryptocurrencies ▾ | Exchanges ▾ | Watchlist | USD ▾ | Next 100 → | View All

#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
1	Bitcoin	$69,980,692,428	$3,987.00	$7,552,943,211	17,552,212 BTC	1.00%	
2	Ethereum	$15,615,642,389	$148.77	$3,635,945,325	104,964,105 ETH	2.49%	
3	XRP	$13,302,659,035	$0.322028	$568,544,969	41,308,998,845 XRP *	0.32%	
4	EOS	$3,519,038,333	$3.88	$1,405,184,199	906,245,118 EOS *	3.53%	

https://coinmarketcap.com/

as of **February 22, 2019**

Ethereum‘s Market Capitalization [US$] (2)



Today's Cryptocurrency Prices by Market Cap

The global crypto market cap is $1.94T, a ▲ 0.83% increase over the last day.
Read more

Alexandria Learning Corner
Hot vs. cold wallets — which do you need?

New Feature
Make the CoinMarketCap homepage your own!

Watchlist | Portfolio | Cryptocurrencies | Categories | DeFi | NFT | Polkadot Eco | BSC Eco | Yield Farming | Show rows 100 | Filters | Customize

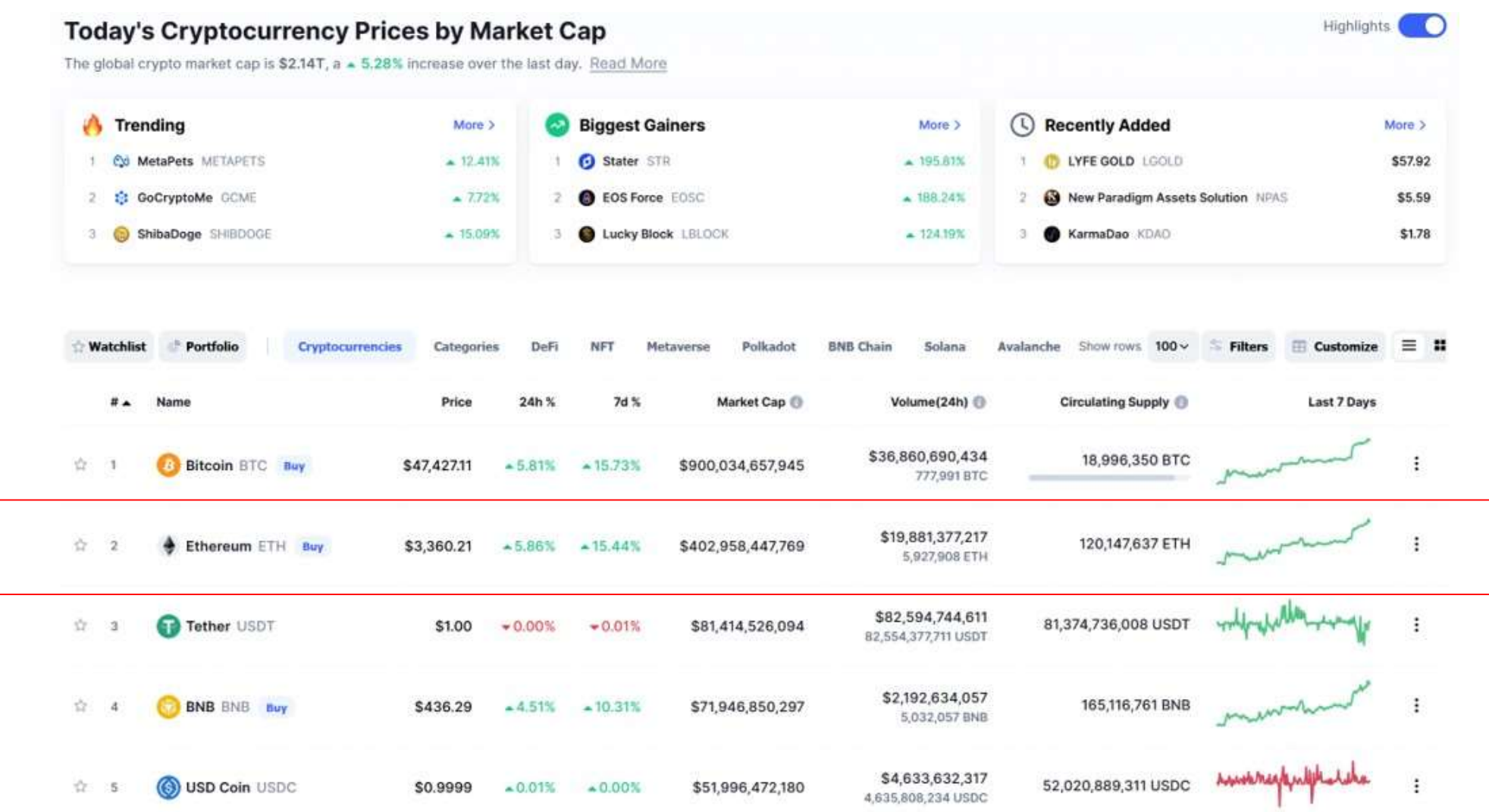
#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC Buy	$58,198.25	▼ 0.02%	▲ 1.82%	$1,086,901,297,261	$51,004,821,493 876,312 BTC	18,674,012 BTC	
2	Ethereum ETH Buy	$2,063.01	▼ 0.10%	▲ 18.39%	$237,916,584,589	$25,895,259,248 12,553,948 ETH	115,341,286 ETH	
3	Binance Coin BNB Buy	$349.88	▲ 1.15%	▲ 26.89%	$54,211,783,582	$3,270,788,237 9,323,508 BNB	154,532,785 BNB	
4	Tether USDT Buy	$1.00	▼ 0.48%	▲ 0.18%	$42,457,264,312	$107,854,577,522 107,657,674,445 USDT	42,379,752,850 USDT	

https://coinmarketcap.com/

as of **April 5, 2021**

© 2022 UZH, CSG@IfI

Ethereum‘s Market Capitalization [US$] (3)



Today's Cryptocurrency Prices by Market Cap

Highlights

The global crypto market cap is $2.14T, a ▲ 5.28% increase over the last day. Read More

Trending		More >	Biggest Gainers		More >	Recently Added		More >
1	MetaPets METAPETS	▲ 12.41%	1	Stater STR	▲ 195.81%	1	LYFE GOLD LGOLD	$57.92
2	GoCryptoMe GCME	▲ 7.72%	2	EOS Force EOSC	▲ 188.24%	2	New Paradigm Assets Solution NPAS	$5.59
3	ShibaDoge SHIBDOGE	▲ 15.09%	3	Lucky Block LBLOCK	▲ 124.19%	3	KarmaDao KDAO	$1.78

Watchlist | Portfolio | Cryptocurrencies | Categories | DeFi | NFT | Metaverse | Polkadot | BNB Chain | Solana | Avalanche | Show rows 100 | Filters | Customize

#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC Buy	$47,427.11	▲ 5.81%	▲ 15.73%	$900,034,657,945	$36,860,690,434 777,991 BTC	18,996,350 BTC	
2	Ethereum ETH Buy	$3,360.21	▲ 5.86%	▲ 15.44%	$402,958,447,769	$19,881,377,217 5,927,908 ETH	120,147,637 ETH	
3	Tether USDT	$1.00	▼ 0.00%	▼ 0.01%	$81,414,526,094	$82,594,744,611 82,554,377,711 USDT	81,374,736,008 USDT	
4	BNB BNB Buy	$436.29	▲ 4.51%	▲ 10.31%	$71,946,850,297	$2,192,634,057 5,032,057 BNB	165,116,761 BNB	
5	USD Coin USDC	$0.9999	▲ 0.01%	▲ 0.00%	$51,996,472,180	$4,633,632,317 4,635,808,234 USDC	52,020,889,311 USDC	

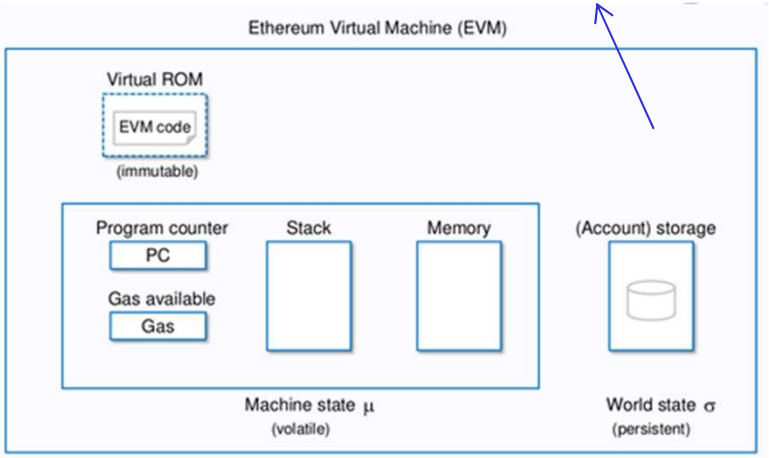
https://coinmarketcap.com/

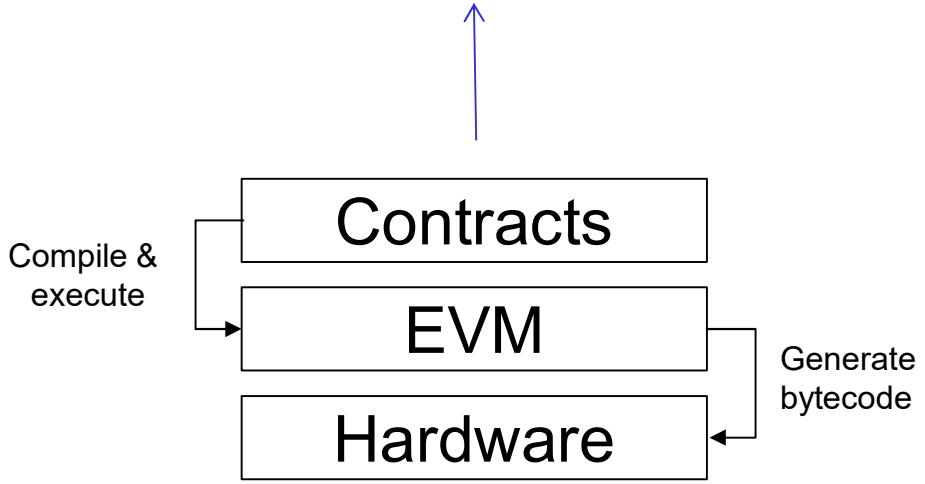
as of **March 28, 2022**

EVM, Implementations, and Smart Contracts

Ethereum's EVM (1)

- Ethereum Virtual Machine (EVM) executes SCs
  - Abstraction layer
    - EVM connects requests (transactions) to and from the network
  - Stack-based architecture





  - SC operations (OP = operands) require "Gas"
    - Mathematical operations
    - Writing data

Table of EVM OP costs:
Gas Costs from Yellow Paper -- EIP-150 Revision (1e18248 - 2017-04-12)
https://docs.google.com/spreadsheets/d/1n6mRqkBz3iWcOlRem_mO09GtSKEKrAsfO7Frgx18pNU/edit#gid=0

Gas Cost Table

- Exact cost of Solidity function can be estimated
  - Useful for optimizations
- Loops → Lead to infinite gas estimations
- Minimum gas amount
  - Transaction: 21,000 gas

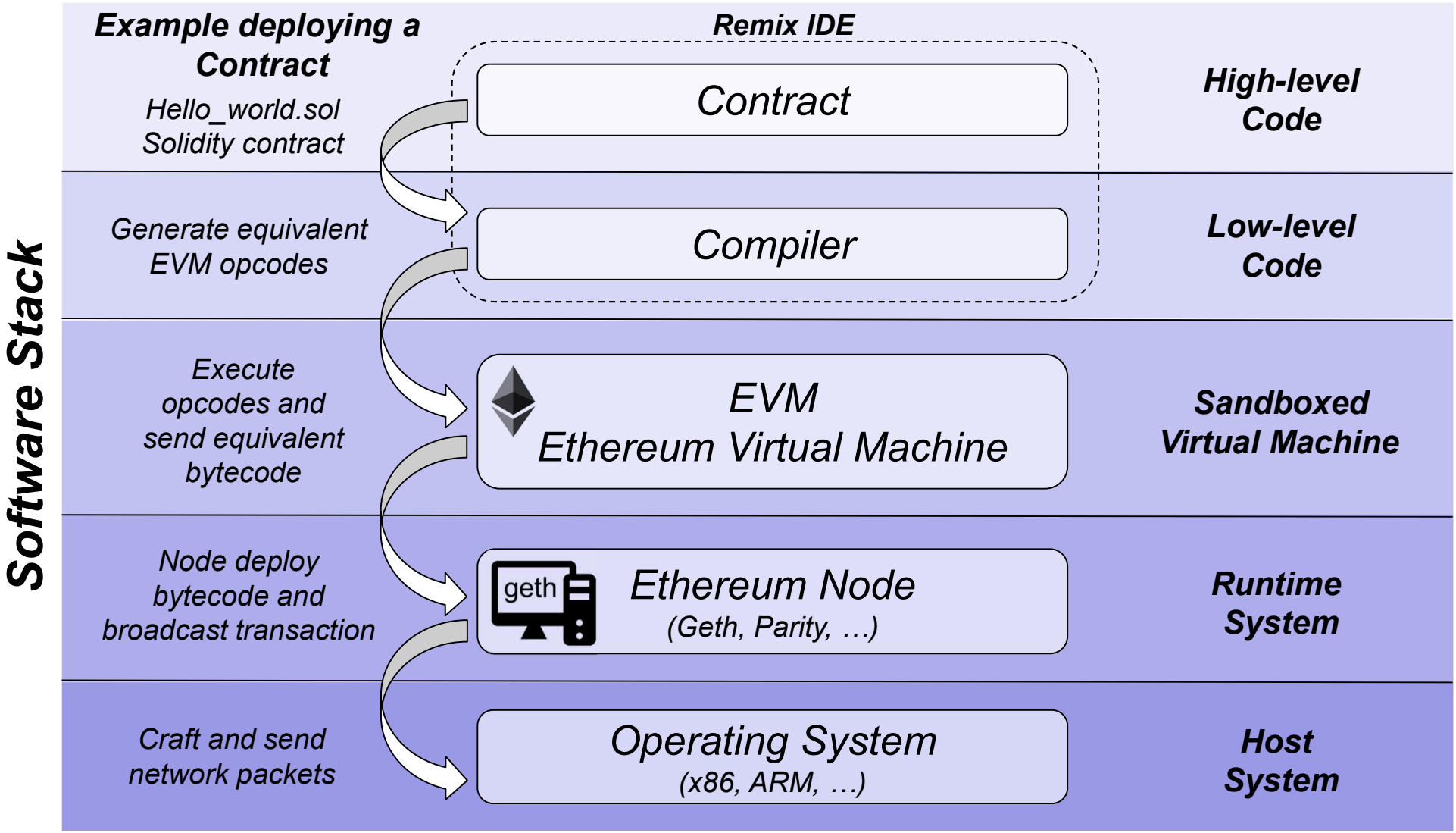
https://ethereum.github.io/yellowpaper/paper.pdf

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
[illegible]	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	20	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

Ethereum's EVM (2)

- Major EVM characteristics
  - Security-oriented virtual machine
    - Program execution is sandboxed
  - Program execution fully deterministic
    - Producing an identical state after transitions performed for any conforming implementation beginning in an identical state
  - Permitting non-trusted code (program = SC) to be executed by a global network of computers
  - Every computational step must be paid for up front (incentivization)
    - Preventing (Distributed) Denial-of-Service attacks
  - Programs/SCs may only interact with each other by transmitting a single arbitrary-length byte array

Ethereum's EVM (3)



© 2022 UZH, CSG@IfI

Ethereum Smart Contracts

- A Smart Contract (SC) is a piece of self-executing and self-enforcing code (program, software)



  - Programs are resistant to censorship of a 3rd party
  - SCs enable complex operations on Blockchain assets
  - SCs are executed directly on-chain

SC Vulnerabilities and Pitfalls

- Contracts are riddled with
  - Bugs and security vulnerabilities
    - SC code is produced by humans and humans are prone to error
    - Most contracts are handling financial assets being publicly readable

- A recent automated analysis of 19,336 Ethereum contracts has shown that at least 8,333 SCs suffer from at least one security issue
  - Ethereum contracts account for 12,205,760 ETH
    - As of April 6, 2021, an equivalent to 25.7 Billion US$

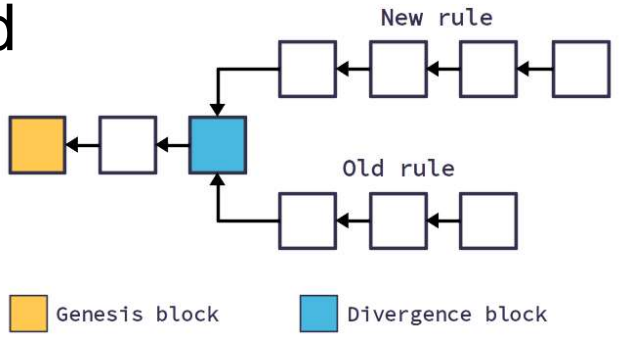
Luu, Loi, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. “Making Smart Contracts Smarter.” ACM CCS, 2016.

Key Difference of SC and "Program"

- Programs are deployed as SCs
- But, SC bugs cannot be patched
  - Once a contract is deployed, its functionality cannot be modified



- Blockchain transactions cannot be rolled back
  - Once malicious transactions are sent, they are recorded and cannot be removed
  - "***Code is the law***" principle



- Unless a hard fork of the blockchain is considered all tx and SCs are final

Solidity and Example SC

Solidity Overview

- Contract programming language similarities to JS and C
  - Types statically typed

    https://solidity.readthedocs.io
    - bool: true and false, int/uint (int8, int16, …, int256), address: 20 bytes, fixed size arrays: bytes1, bytes2, bytes3, ..., bytes32, variable sized: bytes (push), strings

      E.g., ETH transfers
  - Structs

```
struct Account {
  string addr;
  uint amount; //default is 256bits
}
```

  - Mapping (hashing used with strings and addresses)

```
uint counter;
mapping (uint => Account) accounts;
```

  - Contract with state variables, functions, common data types
    - "*There is a high risk and high cost of errors in Solidity code.*"

Solidity Basics

- Version Pragma
  - "reject" compiled code generated by specific compiler versions

```
pragma solidity ^0.4.10; //not before 0.4.10, not after 0.5.0
```

- Comments

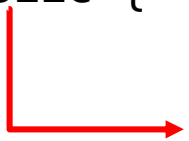
```
// This is a single-line comment.
/*
This is a multi-line comment.
*/
```

- State Variables
  - Expensive for state and data stored inside the Ethereum BC!
    - Better use off-chain alternatives in these cases

```
pragma solidity ^0.4.10;
contract Example1 {
    uint counter;
}
```

A Simple Solidity Smart Contract

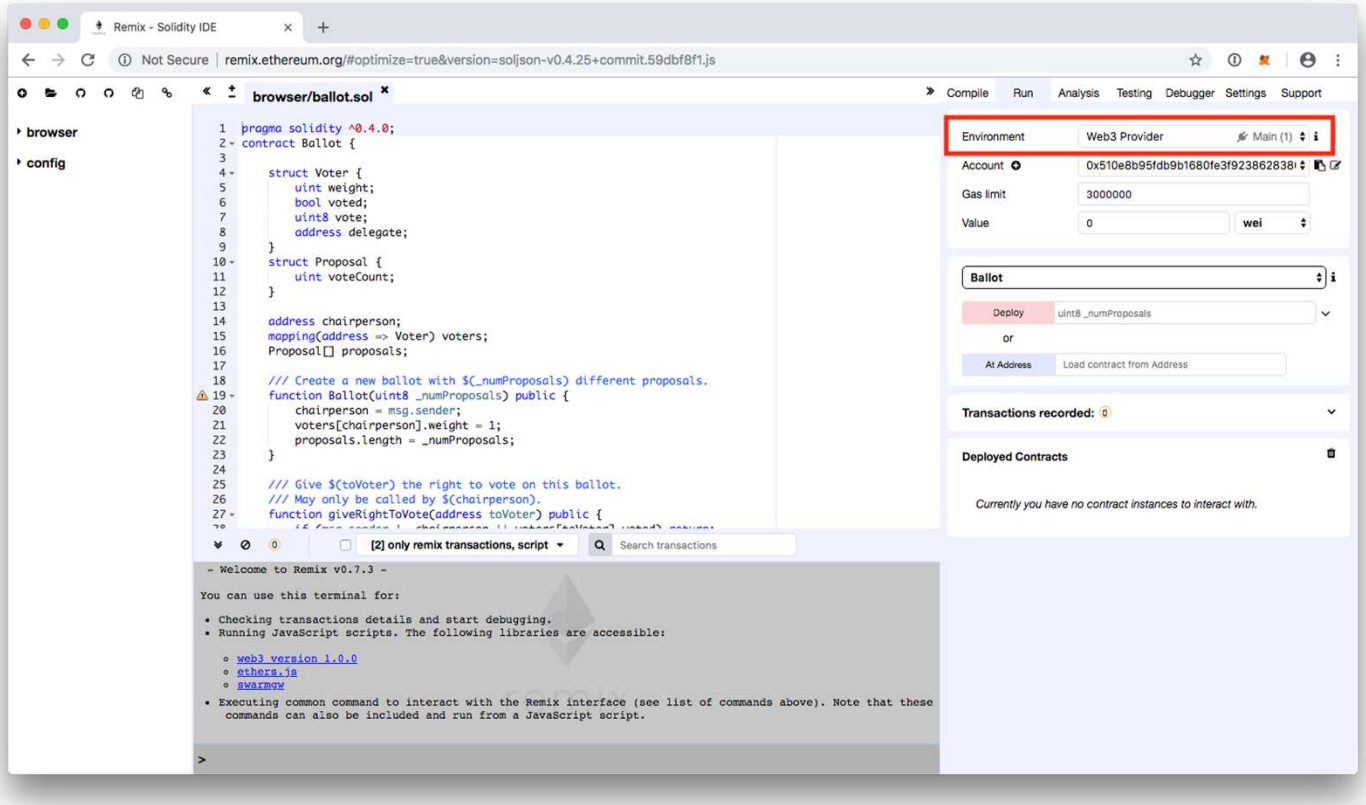
```
1  pragma solidity ^0.4.24;
2
3  contract SimpleStorage {
4      uint256 storedData; // State variable
5
6      // Sets the value of storedData to the value of x
7      function set(uint256 x) public {
8          storedData = x;
9      }
10
11     /*
12      * Returns the value of storedData
13      */
14     function get() public view returns (uint256) {
15         return storedData;
16     }
17 }
```



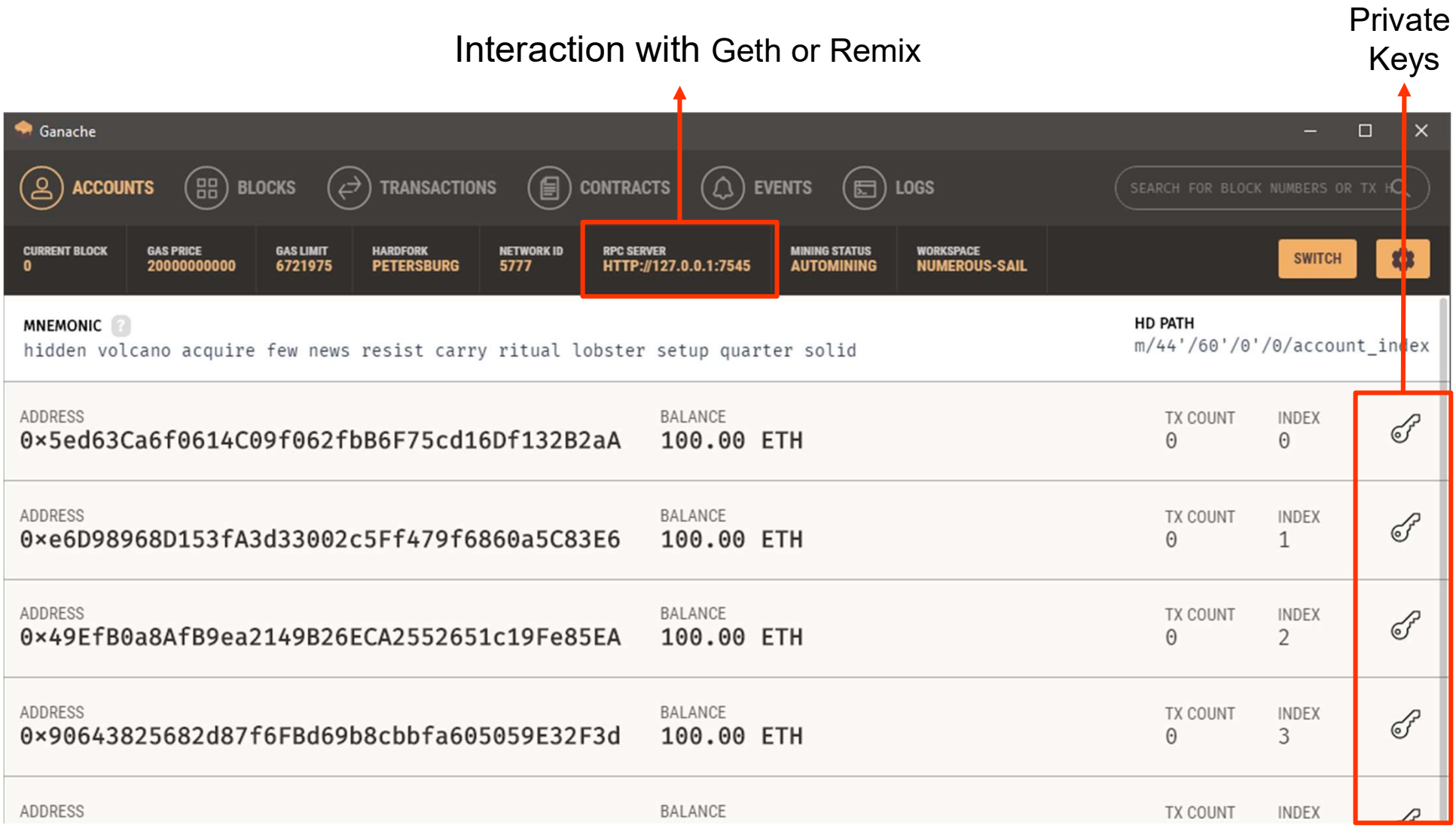
If “**private**”: Everyone sees this function, but it cannot be called from outside this SC.

Development Environment

- Remix IDE (http://remix.ethereum.org)
  - Browser-based development environment with Solidity syntax highlighting



Solidity Deployment – Ganache



Token Standards

Tokens Standards

- Tokens are virtual representations of assets
  - *E.g.,* Lottery tickets, company shares (ICOs), or fiat currency
  - 383,921 ERC 20 Contracts (as of April 2021) https://etherscan.io/tokens
- **ERC 20 – Token Standard**
  - Define standards for the interaction with fungible tokens
    - *I.e.,* all tokens are the same type and have the same value
- **ERC 721 – Non Fungible Token Standard**
  - Define standards for unique tokens
  - Different types and different values
    - *E.g.,* Numbered seats, houses, unique artwork, and collectable cards
- Does not define functions' implementation!
  - Tradeoff between gas saving to improved security

ERC-20

- Functions

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
function approve(address _spender, uint256 _value) public returns (bool success)
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

```
Name = Example Token
Symbol = "EXP"
Decimals = 5 (e.g., 1.25 EXP = 1.25000)
```

- Events

```
event Transfer(address indexed _from, address indexed _to, uint256 _value)
event Approval(address indexed _owner, address indexed _spender, uint256 _value)
```

https://eips.ethereum.org/EIPS/eip-20

ERC-721

Functions

```
function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
function approve(address _approved, uint256 _tokenId) external payable;
function setApprovalForAll(address _operator, bool _approved) external;
function getApproved(uint256 _tokenId) external view returns (address);
function isApprovedForAll(address _owner, address _operator) external view returns (bool);
```

Each Token must have a global ID (sha256) that does not change

Caller is responsible for verifying that "**_to**" can receive the NFT

Events

```
event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);
event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);
event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
```

https://eips.ethereum.org/EIPS/eip-721

Takeaways on Ethereum

- Ethereum is a highly flexible blockchain
  - Account-based model saved in states
  - Community engaged in its standardization
    - Anyone can propose improvements in Ethereum
  - Moving from PoW to PoS
    - Improvements on scalability, transactions per second, and security
    - Energy-efficient consensus
- Provides Turing-complete SCs
  - SC are deterministic → No randomness involved
    - Must be externally called to execute its functions
  - Once deployed → No changes in the code (immutable)
  - Coding errors happen → Security of SCs is crucial

Blockchains' Probabilities and Security



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Blockchains and Probabilities – Basics
  - Definition of probability
  - Selected relations: Keys, mining, consensus, attacks
- Security Basics
  - Privacy and Risk, Pillars and Aspects
    - Variants: RSA, ECC
    - Mechanisms: Hashes, ZKP, Nonce, Trust
- Blockchains' Security Aspects
  - Simplified Payment Verification
  - Anonymity
  - Concerns
- Blockchains and Quantum Security

Basics

Reminder – Probability

- Probability/likelihood is a numerical description of
  - how likely an event is to occur or
  - how likely it is that a proposition is true
- Probability of an event *A* is mathematically defined:
  - P (*A*)
  - Opposite: P(¬A) = 1 − P(A)
- Representation of a probability via a number
  - Between 0 and 1
    - 0 indicates impossibility and 1 indicates certainty
  - Between 0% and 100%
    - 0% indicates impossibility and 100% indicates certainty

Probabilities and Blockchains – Keys

- Public-private key pairs
  - Generation of public-private key pairs performed independently at very many sites (for each tx, possibly)
    - Mechanism based on prime numbers
      - How many prime numbers exist?
      - π(x) is the number of primes less than or equal to x
      - Prime number theorem: approximation of π(*x*) with *x*/(ln *x* - 1)
  - Two public-private keys may be the same
    - Probability of that "astronomically low" (not zero), technically irrelevant
    - Primality tests: Sieve of Eratosthenes, probabilistic Miller–Rabin tests
      - "Composite number" at 75% chance of being detected per round
      - After 64 rounds of testing, the probability that a composite number is undetected is at probability of 2^{-128}
    - Prime factorization of large numbers still remains very "hard"

Probabilities and Blockchains – Mining (1)

- Mining
  - Probability of winning “block race” proportional to computational power of participants
    - Time restriction on block mining; Bitcoin set to the block time: 10 min
    - Computational hashing power increase ⇨ chance of discovering a new block under 10 min increases ⇨ increase target value
  - Centralized mining “attached” with

    (1) unfairness and (2) danger of “51% attack”
    - Incentives to “combine” miner into pools
    - Chances to take over a BC from a country with “many” miners
      - Mining power of above 51% at one country

M. Saad et al.: Exploring the Attack Surface of Blockchain: A Systematic Overview, 2019

Probabilities and Blockchains – Mining (2)

https://csrc.nist.gov/glossary/term/nonce

- Hashing based on a "nonce" ("number only used once")
  - SHA256 (#block, nonce, data, previous block's bash) ⇨ 256 bit hash
  - Each hexadecimal digit with 16 possible values for each of these 32 Byte
    - 64 hex "digits" lead to 16^{64} SHA256 cryptographic hash values
- A target value may be (18 leading zeros, read as hex values)
  - 000000000000000005d97dc000 … 000 [64 hex "digits"]
- Number of total valid hashes is 16^{46}

Based on https://www.linkedin.com/pulse/how-does-bitcoin-blockchain-mining-work-kirill-eremenko

  - Only 64 – 18 = 46 non-zero digits remain
  - Probability that a randomly picked hash is valid can be calculated as:
    - $16^{46} / 16^{64} = 16^{-18} = 0.000,000,000,000,000,02\%$
    - This is the probability that any given nonce value will generate a valid hash for the current block
    - Cryptographic puzzle: Miners compete to find that nonce, which generates a valid hash for the upcoming block ⇨ reward

Probabilities and Blockchains – Consensus

- Consensus mechanisms
  - Assumptions: run in a distributed manner
  - Randomness for multiple parties trying to come to agreement determined by probability of parties' behavior (trusted or malicious)
    - Constraints as upper bounds on

    (a) number of messages sent within a block time (throughput) and

    (b) time it takes for message to be sent (latency)

    ⇨ BC (public) makes it hard to "consent" an agreement via messages needed, thus, a probability is attached to the consent found

    ⇨ solved by PoW and (many) messages needed

      – P2P protocols minimize the count of messages needed
    - PoS: selection of subcommittee ⇨ non-biased? Not possible!

Probabilities and Blockchains – Attacks

- Stakeholders
  - Trusted nodes
  - Maliciously behaving “trusted nodes”
  - Probabilities of an “internal” attacker being successful within

  (a) given public BC or

  (b) given private DL?
  - Many assumptions needed to be made to determine the respective model and its implications
    - *E.g.*, “PeerMart: Secure Decentralized Pricing and Accounting for Peer-to-Peer Systems” analyzed the likelihood of how many peers need to be trusted to operate a decentralized market place in a P2P manner
  - Attack models differ strongly in incentives
    - Economic, destructive, “gaming”, hacking, altruistic, …

Probabilities – A Few "Odds" Examples

- 1 in 693: Full house in Poker
- 1 in 4,164: Four-of-a-kind in Poker
- 1 in 576,000: Being struck by lightning
- 1 in 13.9 m: Winning the 6 out of 49 lottery

  $> 1 * 10^{-7}$
- 1 in 139.838.160: Winning Euromillions (5+2) lottery

  $> 1 * 10^{-8}$
- 1 in 300 m: Dying in a shark attack

  $> 1 * 10^{-8}$
- 1 in 182 trillion: Meteor landing on your house

  $> 1 * 10^{-13}$

© 2022 UZH, CSG@IfI

Security Basics

Internet Privacy and Credentials

- When do people care about Internet privacy?
  - Any technical aspect, which was not asked for and has not been mentioned – not a surprise ☹
  - Payments *were not* mentioned!



- What about credentials?
  - Credentials refer to the verification of identities or suitable approaches for authentication
    - Public-private key pair defines a credential
  - 81% of all hacking activities are targeted to obtain credentials (credential harvesting) https://www.securityweek.com/foundation-cyber-attacks-credential-harvesting

Reminder – Vulnerability, Threat, and Risk

- Vulnerability (Verletzlichkeit):
  - A quality or characteristic of a system that provides an opportunity for misuse.



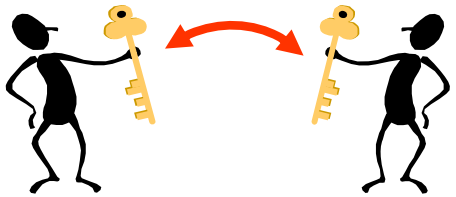
- Threat (Bedrohung):
  - Any potentially malicious or otherwise occurrence that can have an undesirable effect on assets and resources of an IT system.
- Risk (Risiko):
  - = Threat X Vulnerabilities ***OR*** Likelihood X Impact



Security defines a process of risk management, supported by a set of suitable (technical, economic, behavioral) measures!

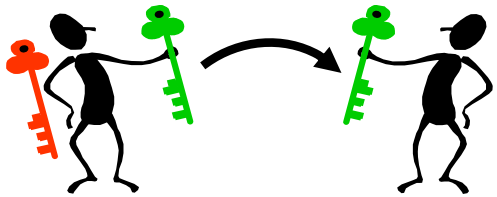
Reminder – Cryptographic Variants

- Symmetric cryptography
  - Entities own a shared, secret key



- Advantages
  - Small overhead/calculation
  - Short keys
- Drawbacks
  - Key exchange complicated
  - No commitment

- Asymmetric cryptography (public key cryptography)
  - Key pair of private/public parts



- Advantages
  - Public keys easy to publish
  - Commitment possible (CA)
- Drawbacks
  - Longer keys
  - Larger overhead/calculation

RSA (1)

- Encryption defines a function f, mapping plain text to cipher text; decryption defines the inverse of f.

RSA: Rivest-Shamir-Adelman

- f requires 5 properties:
  1. f is one-to-one (uniquely invertible)
  2. f is easy to compute (encryption easy)
  3. f^{-1} is difficult to compute (decryption difficult for senders)
  4. f has a domain that is easy to sample from (Bob easily generates a key)
  5. Existence of an easy-to-compute function d of the input of f making computing f^{-1} easy (Bob decrypts easily)

f is one-way

f is a trapdoor function

RSA (2)

- Example trapdoor function

  a mod b: remainder after dividing a by b

  - $\boldsymbol{f}(x,e,p,q) = (x^e \bmod pq, pq, e)$
  - p and q are prime numbers, x (encoded) plaintext, $y=x^e \bmod pq$ is the cipher text
  - $\boldsymbol{d}(x,e,p,q)=e^{-1} \bmod (p-1)(q-1)$ (property 5.) that is $e\boldsymbol{d} \bmod (p-1)(q-1) = 1$
- f is the basis for the RSA Cryptosystem
  - (e, pq) public encryption key (“public key”)
  - (p, q, d) secret decryption key (“secret key”)
  - Alice generates a private/public key pair once (re-use later)
  - Alice encrypts/signs message with her private key
  - Bob decrypts this message with Alice’s public key

Example (Simplified) – RSA (1)

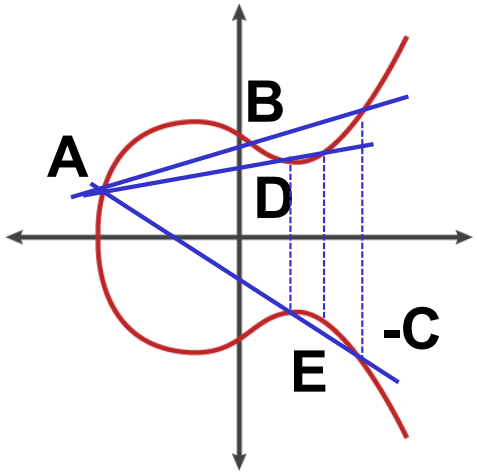
- 2 (large, very large) prime numbers p,q are selected
  - p=11, q=13
- The product N is calculated as N = p * q = 143
- Value e ist calculated to: 1 < e < (p-1) (q-1)
  - e = 23
- Value d ist calculated to: d*e mod (p-1)(q-1) = 1
  - d = 47
- Thus, the number pair (e, N) defines the public key:
  - K_{pub} = (23, 143)
- Thus, the number triple (p, q, d) defines the private key:
  - K_{priv} = (11, 13, 47)

Example (Simplified) – RSA (2)

- Text (plain text) to be decrypted: P = 75
- Calculation of the cipher text y as: $y = P^e \bmod (p * q)$
  - $y = 75^{23} \bmod 143$
  - $y = 69$
- Cipher text C = 69 will be transmitted as the message
- Calculation of the plain text x as: $x = C^d \bmod N$
  - $x = 69^{47} \bmod 143$
  - $x = 75 \rightarrow P$

Basic Elliptic Curve Cryptography (ECC)

- ECC generates a public-private key pair
  - A 256 bit key in ECC is at about the same security as a 3,072 bit key using RSA
  - ECC trapdoor function is similar to a mathematical “game of pool”



  - Knowing where starting point (A) is and how many hops are required to get to the ending point (E), it is “easy” to find the ending point

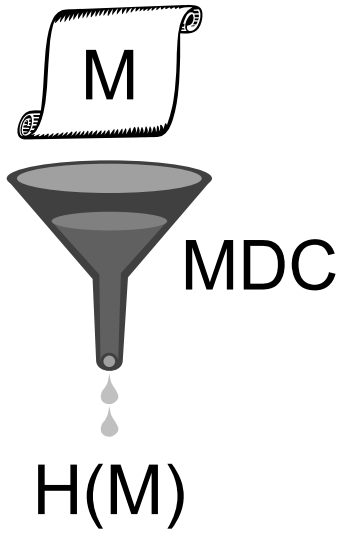
    But by knowing where the starting point/ending points are, it is nearly impossible to find how many hops it took to get there

- Public Key: Starting Point A, Ending Point E
- Private Key: Number of hops from A to E

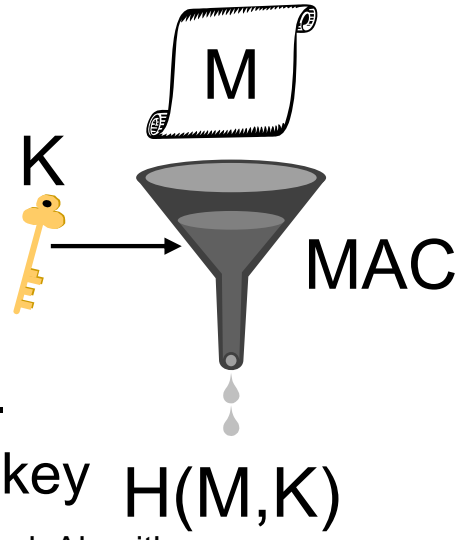
https://blog.goodaudience.com/very-basic-elliptic-curve-cryptography-16c4f6c349ed

Hash Functions and Hashes

- Message Digest Code, MDC
  - Message M (arbitrarily long) → Hash H(M)
    (hash with a minimum of 128 bit length)
  - Note: "One-way" feature of hash function:
    - Efficient generation
    - Very low collision possibility: M, M' with H(M)=H(M')
    - Examples: (MD5), RIPEMD-160, SHA-256



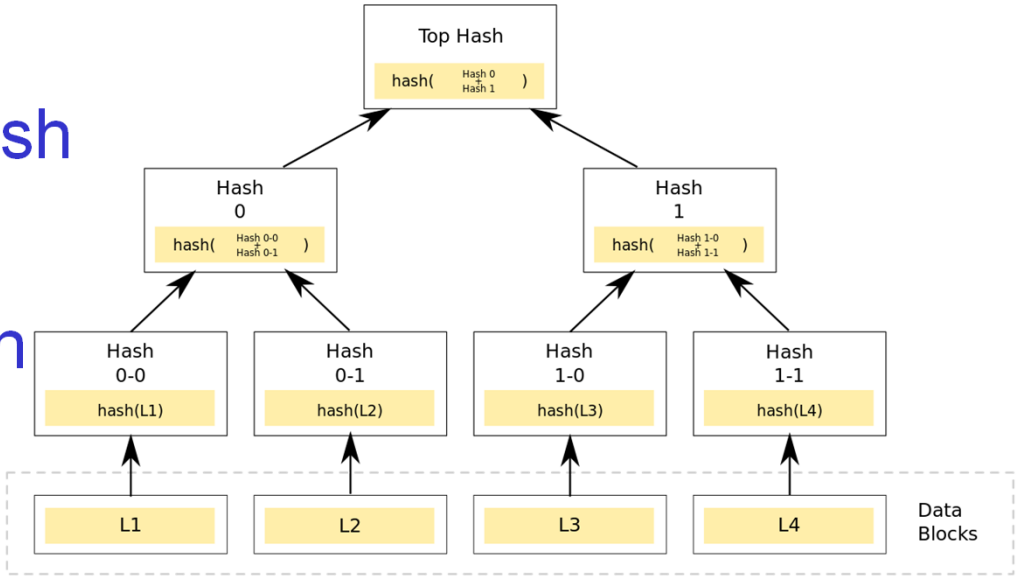
- Message Authentication Code, MAC
  - Message M, key K → Hash H(M, K)
  - May be constructed out of MDC
  - HMAC (RFC 2104), *e.g.*, HMAC-SH-256
  - Cryptographic strength of underlying hash function depending on size of its hash output, size, and quality of key



HMAC: Hash-based MAC, RIPEMD: RACE Integrity Primitives Evaluation Message Digest, SHA: Secure Hash Algorithm

Merkle Trees

- Merkle (or hash) tree
  - Every leaf node is labeled with the hash of a data block and
  - Every non-leaf node is labeled with the cryptographic hash of the labels of its child nodes.
  - Top of a tree there is a root hash
- Characteristics
  - Efficient and secure verification content of large data
  - For binary hash trees: computing hashes is proportional to the logarithm of the number of leaf nodes of the tree
  - Usually cryptographic hash functions used (*e.g.*, SHA-256)



https://en.wikipedia.org/wiki/Merkle_tree#/media/File:Hash_Tree.svg

Zero Knowledge Proofs (ZKP)

- A ZKP is a method by which one party (the prover) can prove to another party (the verifier) that they know a value x, without conveying any information apart from the fact that they know the value x (as of 2013).
  - Any third party reviewing the transcript created cannot be convinced that either the prover or the verifier knows the secret.
    - Ideal cryptographic hash function required
  - Originates from authentication systems, where one party wants to prove its identity to a second party via a secret
  - New: to guarantee transaction validity, despite the fact that information about the sender, the recipient, and other transaction details remain hidden

https://en.wikipedia.org/wiki/Zero-knowledge_proof

Nonce

https://csrc.nist.gov/glossary/term/nonce

- Nonce
  - Arbitrary number that can be used just once in a cryptographic communication
    - Often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks
  - A nonce may also include a timestamp to ensure exact timeliness
    - Prerequisite: clock synchronization between organizations
  - Example
    - Salt: random data that is used as an additional input to a one-way function hashing data (password or passphrase)

Trust

- General: Trust is the firm belief in the reliability (character, strength), truth, or ability of someone or something
- In Information Security: Computational trust is the generation of trusted authorities, trusted protocols, or user trust through cryptography
- Key trust facets
  - Initial level of trust needed as an incentive to participate
  - Theoretical aspects of trust and its quantification
  - Relationship between security and trust
  - Significance of trust in distributed network security

https://pdfs.semanticscholar.org/0567/12cf5bc0fba369eb60ebbc23b327d23b1ab8.pdf

Key Security Aspects to Remember

- Security mechanisms are inevitable for modern communication networks and distributed systems
  - Operational and technical security and user behavior
- Security is only as strong as its weakest component
- Security is costly and does affect the ease-of-use concept of a system and its interfaces
- Trust increases/decreases and evolves over time as a consequence of security mechanisms applied
  - All security aspects (mechanisms and use) in combination only can deliver the basis to generate trust between users and systems; trust exists between users/systems

Blockchains‘ Security Aspects

Blockchain Security – Basics

- BC security is based on existing (a) asymmetric cryptography protocols and (b) hashing schemes

The Principle is to create computationally expensive problems, *i.e.*, NP-hard problems that can be solved in exponential time only

1. Discrete logarithm problem: ECC, ECDSA, ECDH, ElGamal, …
2. Factorization of large prime numbers: RSA, Benaloh, ...

- In practice, Bitcoin and Ethereum use
  a) ECDSA to generate public keys
  b) SHA-256 (Bitcoin), Keccak (Ethereum)





ECDH: Elliptic-curve Diffie–Hellman
ECDSA: Elliptic Curve Digital Signature Algorithm

Simplified Payment Verification (SPV) (1)

- Key question
  - Do all nodes have to run as a full node (the full copy of the entire BC stored locally) to be as secure as possible?
- SPV can validate a node's transaction without the need to know about anybody else's transactions
  - Sec. 8 of Nakamoto's White Paper on Bitcoin
  - Saves users from hassles of maintaining a full node
- Two ingredients

  (a) It ensures the node's transactions to be in a block and

  (b) it provides confirmations (via PoW) that additional blocks are being added to the chain

https://medium.com/@jonaldfyookball/why-every-bitcoin-user-should-understand-spv-security-520d1d45e0b9

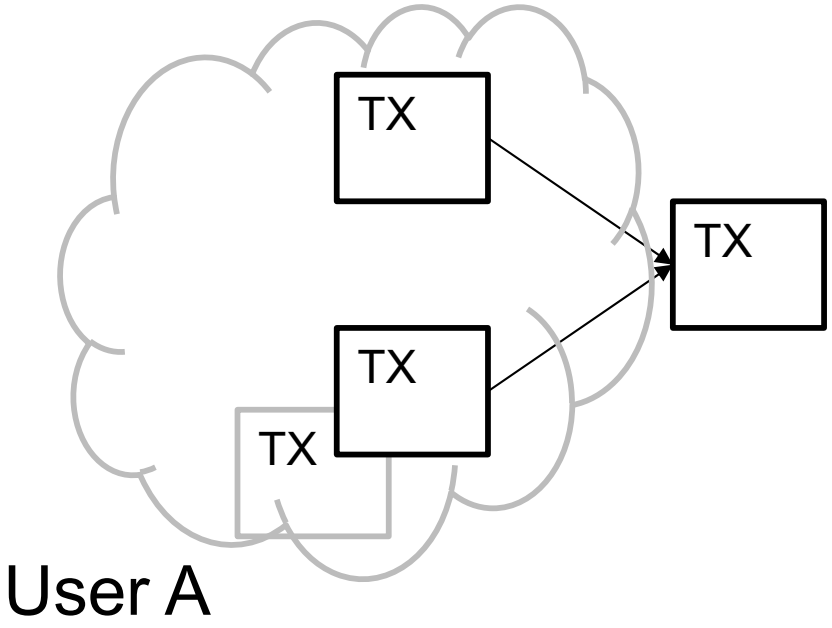
Simplified Payment Verification (SPV) (2)

- Analysis
  - SPV wallet does not check other transactions in the block
    - Thus, theoretically the block could be invalid
  - 99.99% of the time, this is not a problem (high probability)
    - Since the node can connect to many different nodes and can make sure that everyone agrees on the same chain of blocks!
  - Occasionally "unusual" situations happen (low probability)
    - A new consensus rule is adopted, and a larger group of nodes does not yet uses the new rule, since they forgot to upgrade
    - Non-compliant nodes are forked off the network soon
    - The SPV client could be temporarily fooled into thinking the non-upgraded nodes had the correct longest chain, but after a few more blocks, the longest valid chain would become apparent again

https://medium.com/@jonaldfyookball/why-every-bitcoin-user-should-understand-spv-security-520d1d45e0b9

Reaching Anonymity (1)

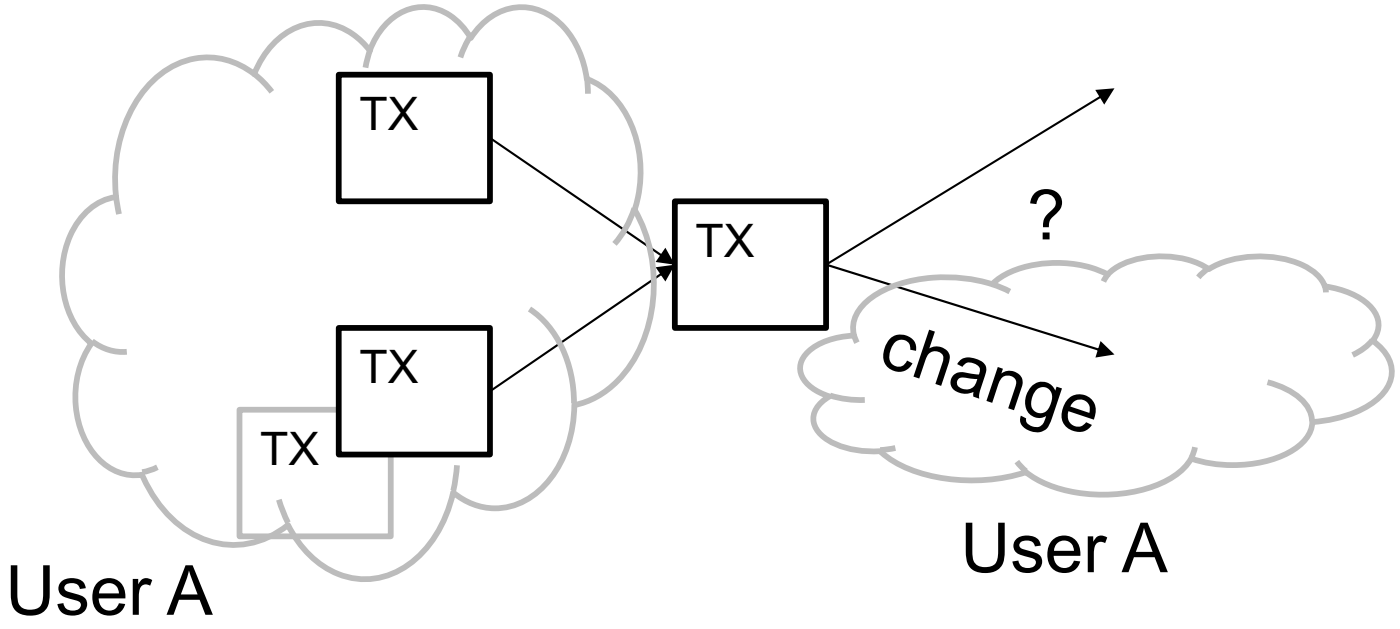
- BitcoinJ (library for the Bitcoin protocol)
  - Bloom filters for improved wallet performance and SPV security
- Heuristic 1
  - If two (or more) addresses are inputs to the same transaction, they are controlled by the same user, *i.e.*, for any transaction t all pk ∈ inputs(t) are controlled by the same user



Based on http://eprint.iacr.org/2014/763.pdf and others

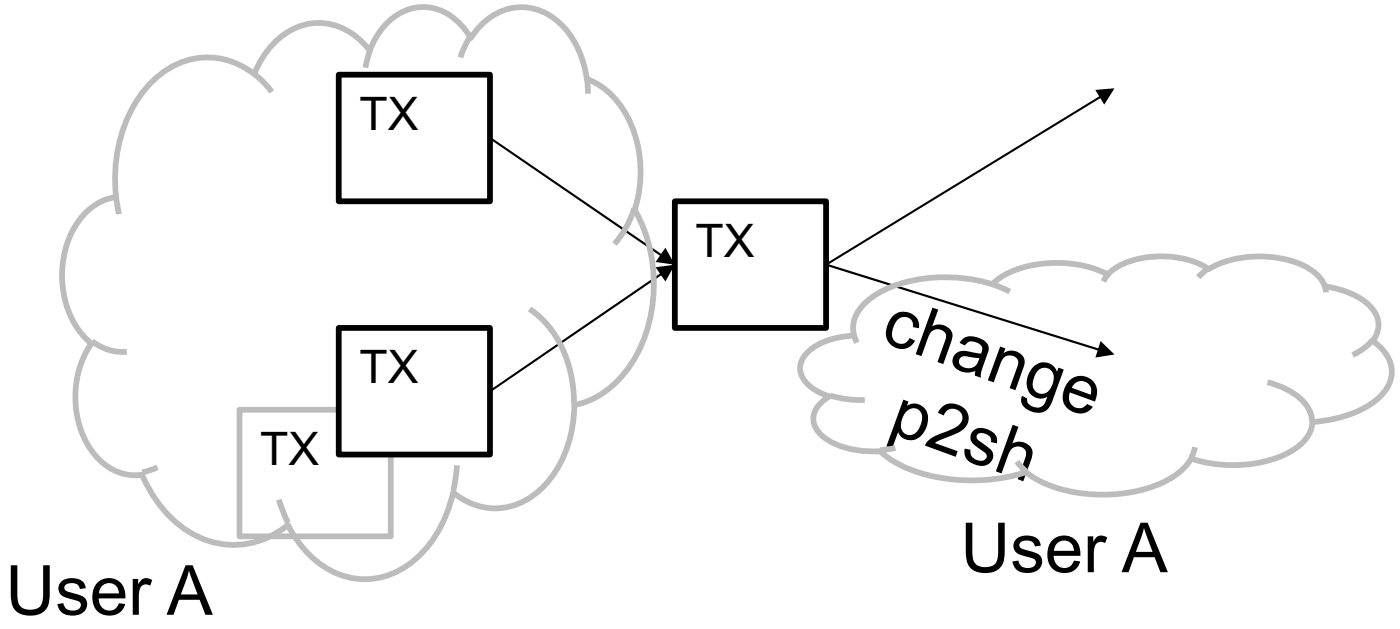
Reaching Anonymity (2)

- Heuristic 2
  - One-time change address (“remainder BTC”) is controlled by the same user as an input address, *i.e.*, for any transaction t the controller of inputs(t) *also* controls the one-time change address pk ∈ outputs(t) (if such an address exists)



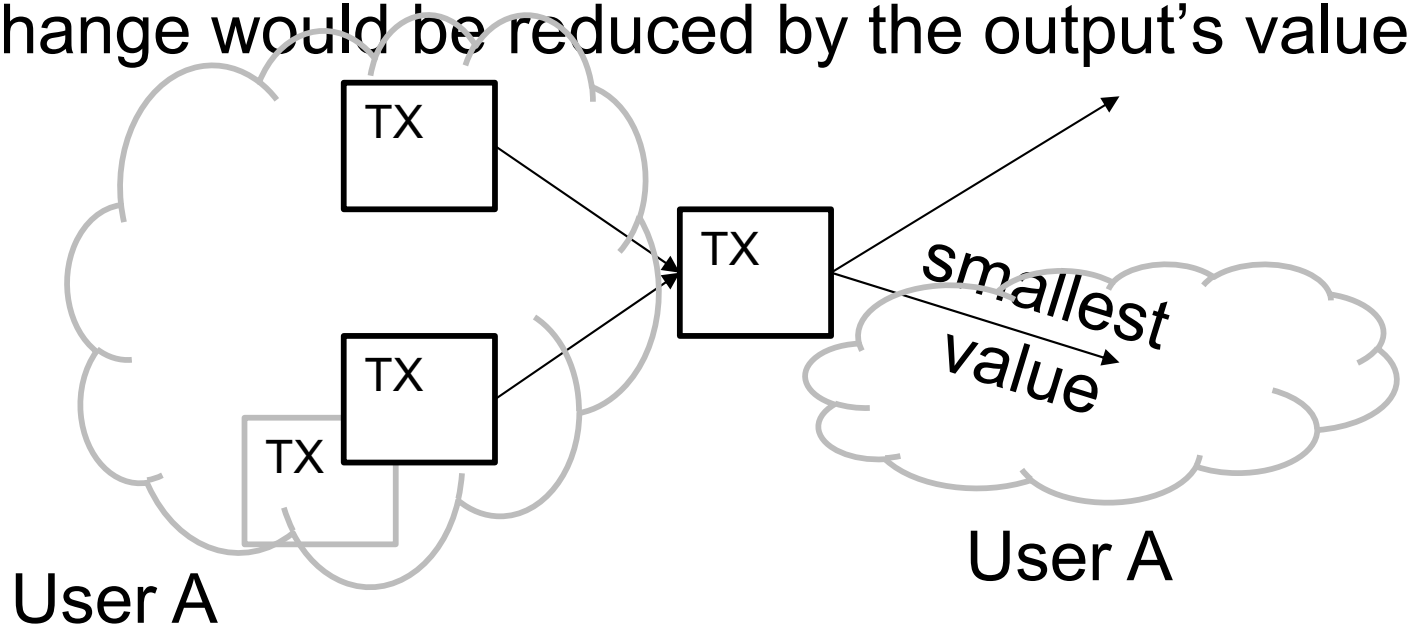
Reaching Anonymity (3)

- Heuristic 3
  - Multisig wallets usually use a P2SH change address (Pay To Script Hash Address), but the recipient rarely uses a P2SH address, which allows to determine the correct change output with a high probability



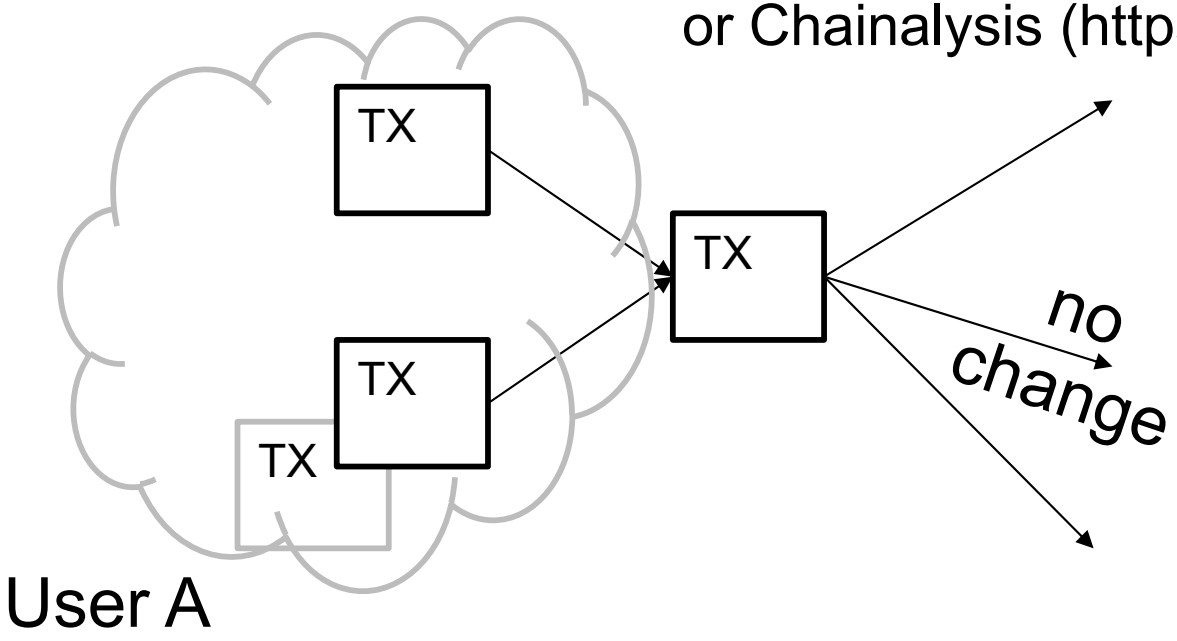
Reaching Anonymity (4)

- Optimal Change Heuristic
  - Assumption: wallet software does not spend outputs unnecessarily. Therefore, the change value is smaller than any of the spent outputs. Because if the change was larger than one output, then this output would be left out and the change would be reduced by the output's value.



Reaching Anonymity (5)

- Consumer Heuristic
  - Consumer wallets only create transactions with two outputs
  - Thus, if an output is spent by a transaction with 3 outputs it is no change
    - Example analysis: WalletExplorer (https://www.walletexplorer.com/) or Chainalysis (https://www.chainalysis.com/)



Security Concerns – Theft (1)

- Bitcoinica was an on-line service enabling leveraged speculation, based on Linode (cloud provider)
  - Gaining root access using the service provider's systems
  - 46k BTC were stolen (in 2012 ~ 228,000 US$, today …)
  - 2 month later: 19k BTC were stolen
    - "Hacking Web server (Rackspace) by resetting a password, most likely through an automated e-mail"
- Bitfloor, a US-based Bitcoin exchange site
  - 24k BTC stolen (in 2012 ~ 250,000 US$, today …)
  - Attacker gained accesses to an unencrypted backup of the wallet keys

Security Concerns – Theft (2)

- Bitcoin-Central (currency exchange: https://bitcoin-central.com/)
  - Some hundreds Bitcoins stolen in early days
  - Password reset – issue with OVH (Webhosting, Cloud, Virtual Private Server und Dedicated Server) in March 2021?
- BitInstant (service for quickly and easily transferring funds into an online exchange account without delay, 2011-14)
  - 2012: 333 BTC stolen ~ 12,000 US$ (today …)
    - Attack: Domain registrar Site5: redirected DNS
  - "Armed with knowledge of my place of birth and mother's maider name alone (both facts easy to locate on the public record) they convinced Site5 staff to add their email address to the account and make it the primary login."

Security Concerns – Theft (3)

- 2012: ~ 9,222 BTC theft (today worth …)

  https://www.20min.ch/finance/news/story/ETH-Doktorand-um-Millionen-erleichtert-22056755

- March 23, 2022: ~ 173,600 ETH and 25,5 m USDC

  Ronin Network is an Ethereum sidechain built specifically for Axie Infinity, which is a crypto game that rewards its players with crypto tokens. 2 signatures were compromised (while 4 existed).

  https://www.nzz.ch/technologie/hacker-stehlen-kryptowaehrungen-im-wert-von-ueber-600-millionen-dollar-was-genau-ist-passiert-ld.1677129

- List of major Bitcoin heists

  USDS: digital stablecoin on US$

  – Including thefts, hacks, scams, and losses

  https://bitcointalk.org/index.php?topic=576337

- Outsourcing infrastructure to the cloud
  – Outsourcing the risk
  – *E.g.*, Bitcoin-Central and Bitcoinica formally served a "secure" system during the time the theft had happened!

Security Concerns – Exchange Example

- Mt. Gox
  - Short for “Magic: The Gathering Online Exchange”
  - July 2010 reused domain name, started Bitcoin trading
- April 2013: handled 70% of Bitcoin trades
- November 2013: users reported payout delays
  - Exchange rates different from other trading platforms
- February 7, 2014: all BTC withdrawals suspended
  - “Bug in the Bitcoin system”: transaction malleability
- February 20, 2014: all withdrawals suspended
- February 24, 2014: all trading suspended, blank page
- March 9, 2014: filed for bankruptcy

Security Concerns – DoS and Availability

- DDoS against Mt. Gox – "Layer 7" April 2013
  - Trading engine lag ~ 75 min
  - Panic sells – market prices dropped

https://www.theregister.co.uk/2013/04/22/mtgox_ddos/

- Mt. Gox closed (February 2014), Bitfloo closed (April 2013), Bitcoin24 closed (April 2013), and others
  - Note these DDoS "Successes" did not harm BTC and its protocols and they did not exploit bitcoins as such!
- 45% of Bitcoin exchanges ended up closing
  - 40 Bitcoin exchange sites analyzed over 3 years, 18 closed
  - 5 closed exchange sites did not provide reimbursements

http://lyle.smu.edu/~tylerm/fc13.pdf

Security Concerns – Malleability

- Tx malleability

https://www.newsbtc.com/2017/03/11/bitclub-performs-successful-malleability-attack-bitcoin-network/

  - Transactions are identified by the SHA256 hash
  - Malleability = change data in transaction to change tx hash, but transaction remains valid (signature still verifiable)
    - Known since May 2011

- Signature Malleability

ASN.1: Abstract Syntax Notation
ECDSA: Elliptic Curve Digital Signature Algorithm
DER: Distinguished Encoding Rules
SSL: Secure Socket Layer

  - Signature is DER-encoded ASN.1 (done by OpenSSL)
  - OpenSSL parser is not strict (like an HTML browser)
  - ECDSA signature (r,s) = (r, -s (mod N)) → valid signature

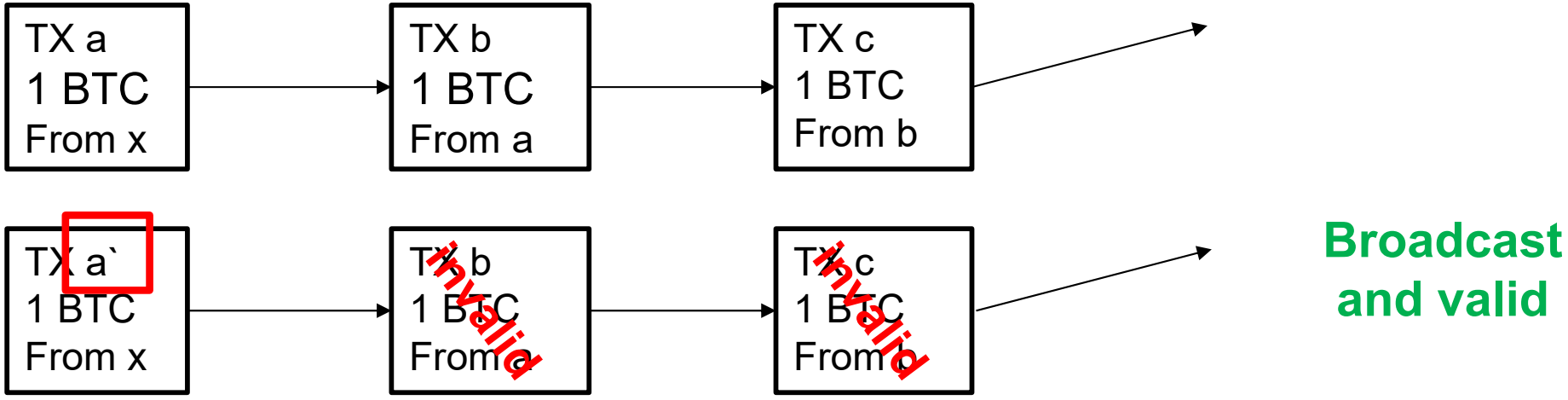
- scriptSig Malleability
  - Signature may not sign any data structure containing itself
    - Use OP_0, replace later. Script may be encoded in several different ways

Transaction Malleability in Detail

- Chained transactions (not broadcasted yet)
  - Tx with hash and index

https://en.bitcoin.it/wiki/Transaction

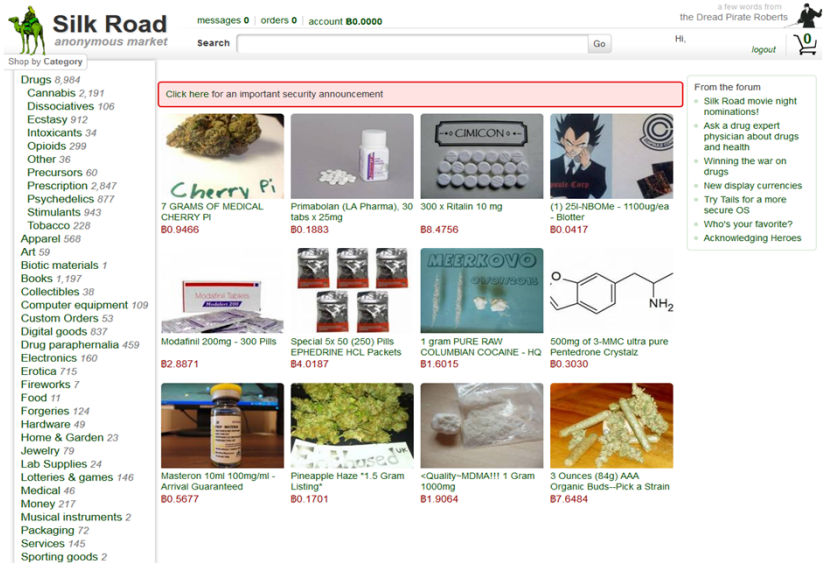
    - "A transaction is a transfer of a Bitcoin value that is broadcast to the network and collected into blocks. A transaction typically references previous transaction outputs as new transaction inputs and dedicates all input Bitcoin values to new outputs. Transactions are not encrypted, so it is possible to browse and view every transaction ever collected into a block."



Security Concerns – Silk Road (1)

- Founded in February 2011 by “Dread Pirate Roberts“
  - Mainly illegal goods (“eBay for drugs”)
  - Connection through Tor hidden service, payment in Bitcoins
  - 1.2 US$ revenue/month,
    92k US$ commissions/month

    https://arxiv.org/abs/1207.7139v2
  - Up to 150,000 active customers
    - ~ 4,000 vendors
    - 900, 000 registered users
  - Could link a forum account of Silk Road to a Gmail account
    - Logged in to his account
    - “FriendlyChemist” demanded 500,000 US$ to prevent release of identities



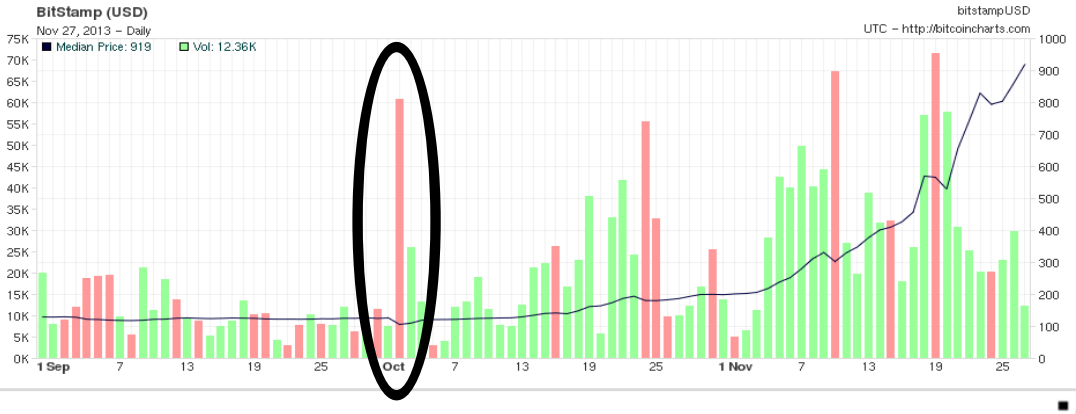
https://edition.cnn.com/2013/10/04/world/americas/silk-road-ross-ulbricht/

Security Concerns – Silk Road (2)

- Taken down on October 2, 2013
- Assets seized by FBI
  - Over 144,000 BTC from Ulbricht
  - 5 million US$ from MtGox frozen
  - *"In fact, the 174,000 or so Bitcoins that the FBI controls now account for about 1.5% of all Bitcoins in circulation."*
- Successors already in place
  - Silk Road 2.0,
    Silk Road 3.0,
    OpenBazaar?
- Bitcoin exchange rate
  - Not much happened



https://www.forbes.com/sites/andygreenberg/2013/10/25/fbi-says-its-seized-20-million-in-bitcoins-from-ross-ulbricht-alleged-owner-of-silk-road/#54e1a08f2765



Blockchains and Quantum Security

Traditional vs. Quantum Computing

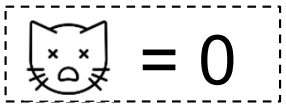
Traditional

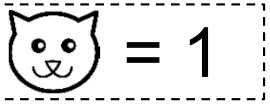
- Based on Bits: stream of electrical/optical pulses represented as 0's or 1's
  - A Bit is either 0 or 1
  - Complex structures are translated in binary

Quantum

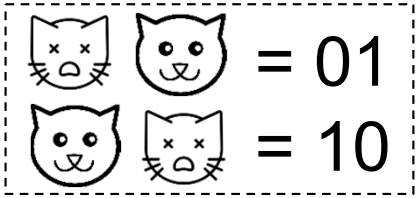
- Based on Qubits: use Quantum Mechanics Properties (QMP) superposition and entanglement to store combinations of 0's or 1's

Schrödinger's cat is either dead or alive but not both

Bit  = 0

Bit  = 1

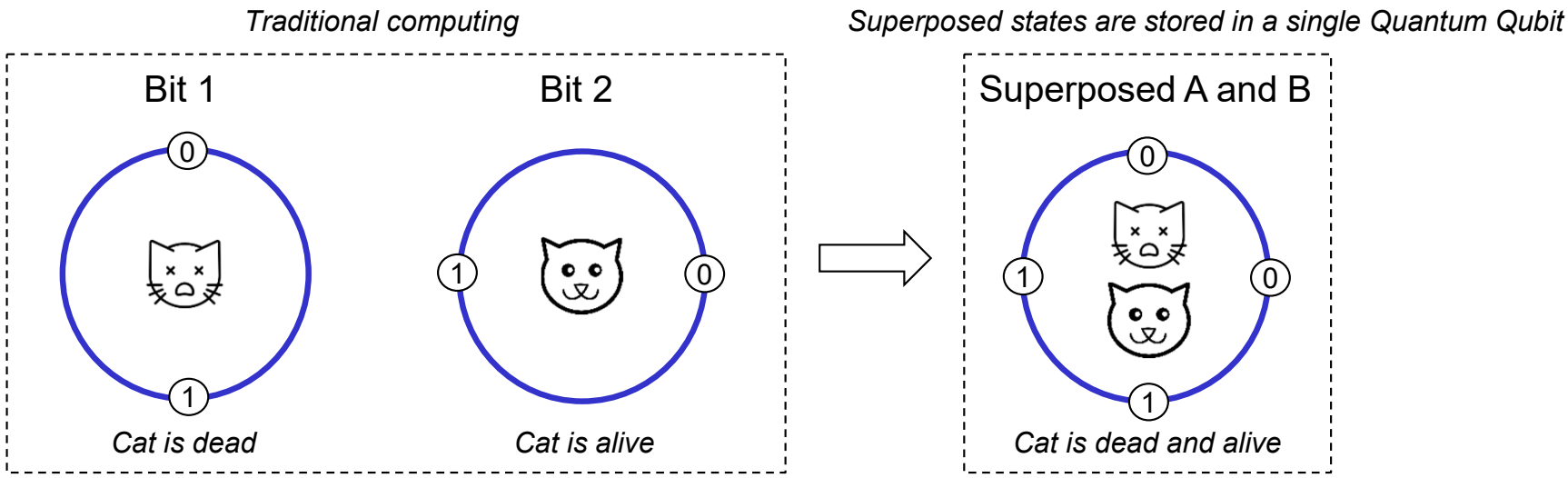
Schrödinger's cat can be in any combination of dead and alive at the same time

Single Qubit  = 01

= 10

QMP – Superposition

- Two or more states can be combined (superposed) resulting in another valid quantum state

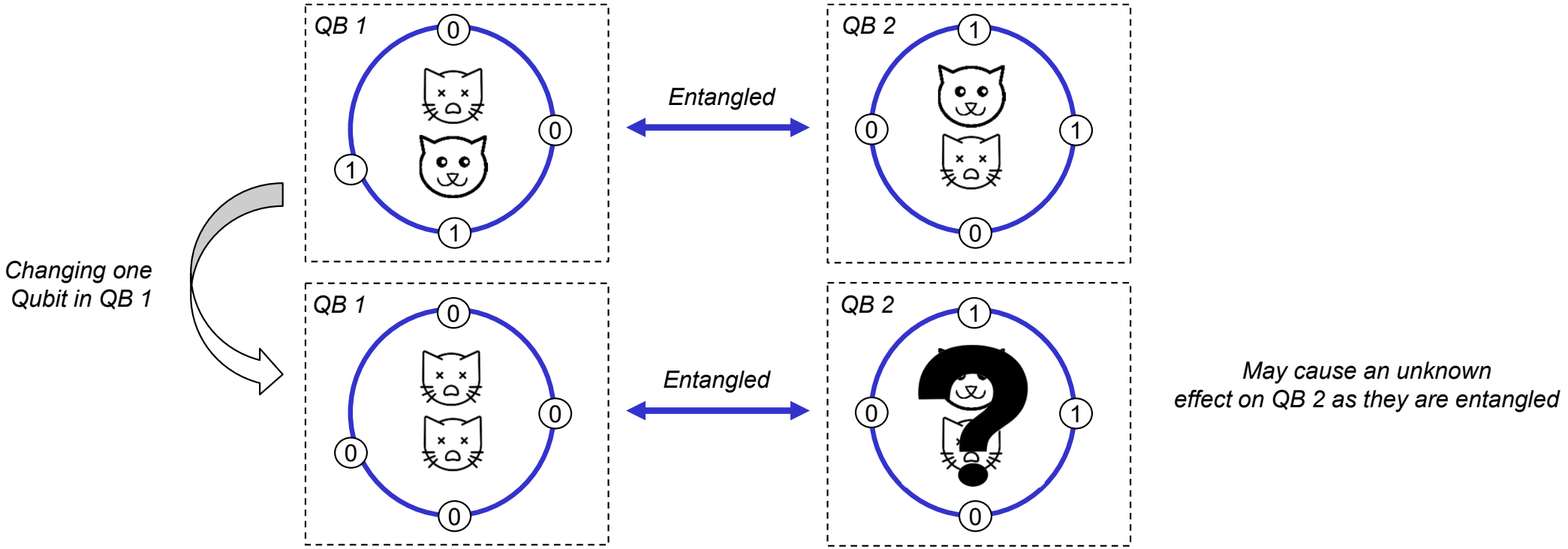


  - Whereas, a bit in traditional computer would only store 0 or 1, so a Cat would be *dead* or *alive*

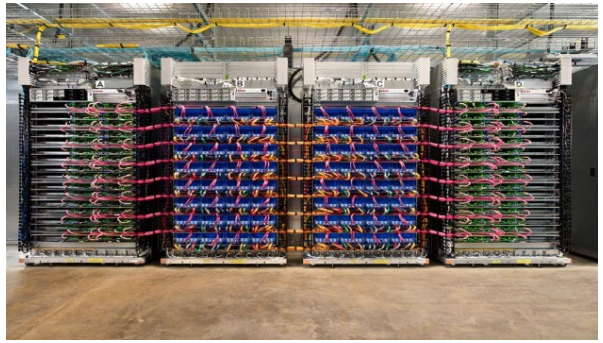
QMP: Quantum Mechanics Properties

QMP – Entanglement

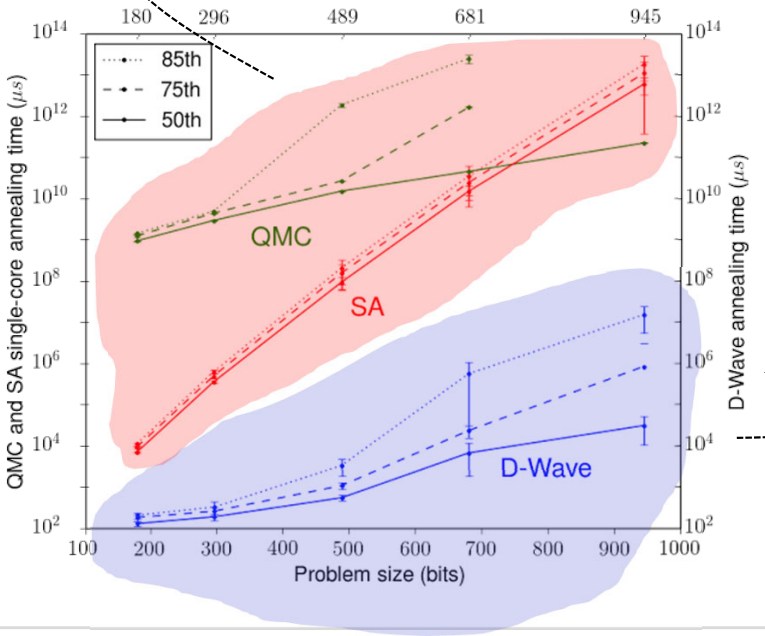
- Pairs of Qubits (QB) can exist in a way that their quantum state cannot be described independently
  - If the Schrödinger's cat is dead-alive, then the cat is alive-dead.



Contrasting the Performance Difference



Simulated methods (QMC and SA) running on a traditional Google supercomputer



Lower = F

Google's D-Wave Quantum Computer

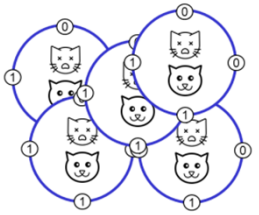


More than **108 times** faster than classical counterparts to solve a combinatorial problem than SA (Simulated Annealing) and QMC (Quantum Monte Carlo)

https://ai.googleblog.com/2015/12/when-can-quantum-annealing-win.html

Impacts on Asymmetric Cryptography

- In theory, existing asymetric protocols would be broken
  - A quantum computer would solve existing exponential problems in polynomial time based on the *Shor's algorithm (1994)*:
    - A theoretical proof that problems involving discrete logarithms (ECDSA) and factorization of prime numbers (RSA) would change from exponential to polynomial time using a quantum computer
  - In theory, this would make it easier to steal BTC or ETH by deriving private keys from public keys



- In practice, a quantum computer is infeasible to achieve a sufficient scale of Qubits as of today

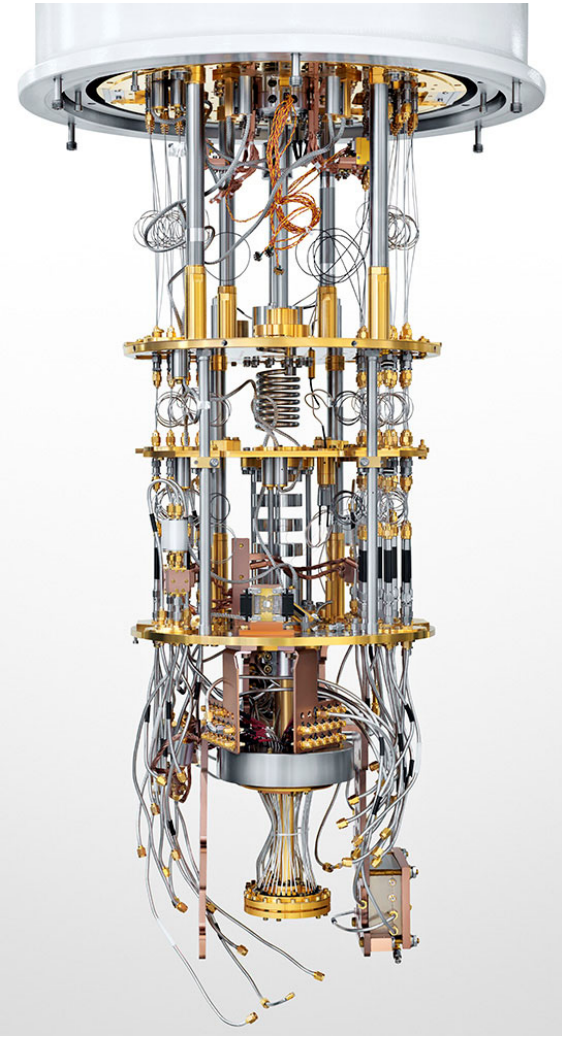
Impacts on Hash Functions

- In theory, selected existing protocols can survive quantum computers, *i.e.,* they are quantum-resistant
  - The *Groover's algorithm (1996)* can be used to find hash collisions
    - Quantum algorithm to find the probability of an input to produce a particular output
  - A Quantum computer would always win the mining race based on existing output sizes of SHA-256 and Keccak
- In practice, SHA-256 (SHA2) or Keccak (SHA3) would would resist by
  - Increasing the output hash size
  - Raising the mining difficulty

If and only if a quantum computer with sufficient Qubits scale is available

Advantages and Disadvantages

- Main quantum computing advantage
  - Greater Performance
    - Superposition: Qubits store more values of 0's and 1's simultaneosly
    - Entanglement: Scale exponentially as more Qubits are added (entangled)
    - Result: quantum computers are exponentially faster than traditional computers
- Main disadvantage
  - Lack of reliability
    - Decoherence: Qubits are extremely sensible to environment conditions (vibrations or temperature changes), thus, they disturb superpositions



Most powerful Quantum computer as of 2018
Rigetti Computing - 128 Qubits

General Quantum Mechanics Impacts

- Quantum mechanics as of today are a theoretical threat
  - Existing asymmetric cryptography protocols are, in theory, broken
  - However, the entire Internet is based on AES, RSA, ECC 
    - Unsafe Blockchains would **not be the largest problem** at all

... and in reality

  - Large scale Quantum computers are still not reliable 

- Future steps and concerns (BC-related and general IT)
  - Theory is also advancing on post-quantum cryptography
    - But these will not work on traditional computers
  - Transition from traditional to quantum computing not smooth

Blockchain Interoperability and Standardization



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Interoperability Demand
  - Mechanism Overview
  - State-of-the-Art
- Dedicated Interoperability Approaches
- Open API (Application Programming Interface)
  - Goals and Requirements
  - Application: Blockchains for Coldchains (BC4CC)
  - Workflow, Implementation, and Adapters
  - Comparison
- BC Standardization Efforts
  - Current Activities
  - Overall Judgment

© 2022 UZH, CSG@IfI

Interoperability Demand

Need for Interoperability

- Each blockchain (18,600+) has its pros and cons
  - Transparency vs. privacy
  - Transaction fees, speed of transaction confirmation

As of https://coinmarketcap.com/ in terms of cryptocurrencies March 31, 2022

- Interoperable chains enable several opportunities
  - Move assets between different platforms
  - Access information from one chain inside other chain
  - Different payment schemes
  - Allowing to choose the best alternative BC possible

- V. Buterin: "In the longer term, building interoperable applications is something that should be done on a basis of responding to actual need; while the underlying technology can certainly be developed in the abstract, it would likely to be a waste of resources to try to build applications that connect chains without a clear understanding on the reasons driving the use case."

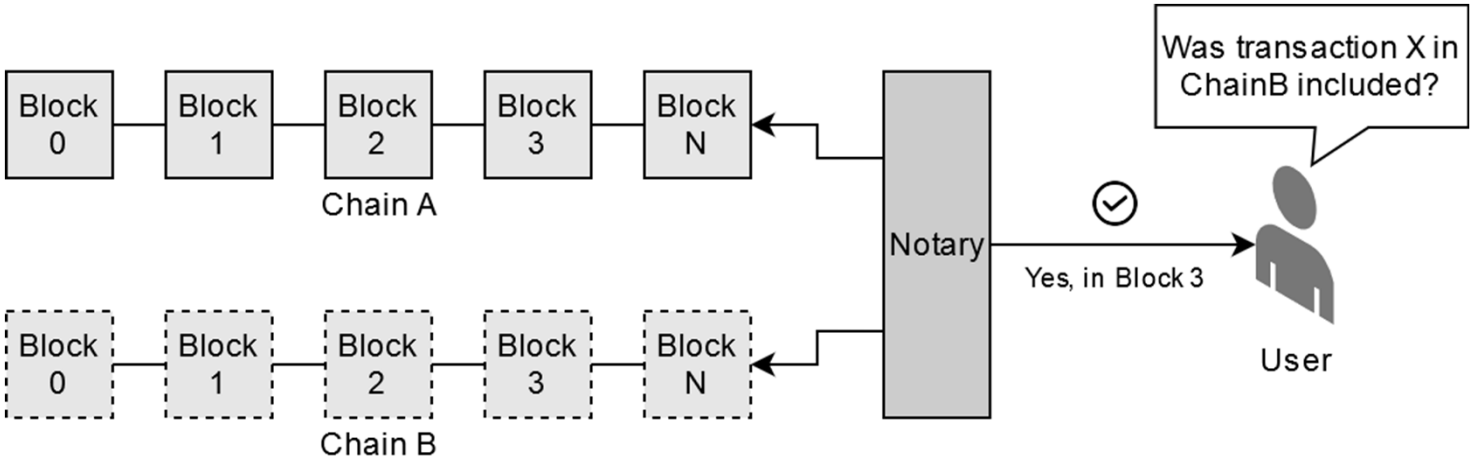
Types of Interoperable Chains

- Notary schemes
  - One or more trusted parties validate events on another chain
  - Ease of implementation: medium
- Sidechains/relays
  - The chain validate events on external chains as part of validating local events
  - Ease of implementation: hard
- Hash-locking Approaches
  - Events on different chains are unlocked using a secret value
  - Ease of implementation: easy

V. Buterin, Chain Interoperability, September 2016

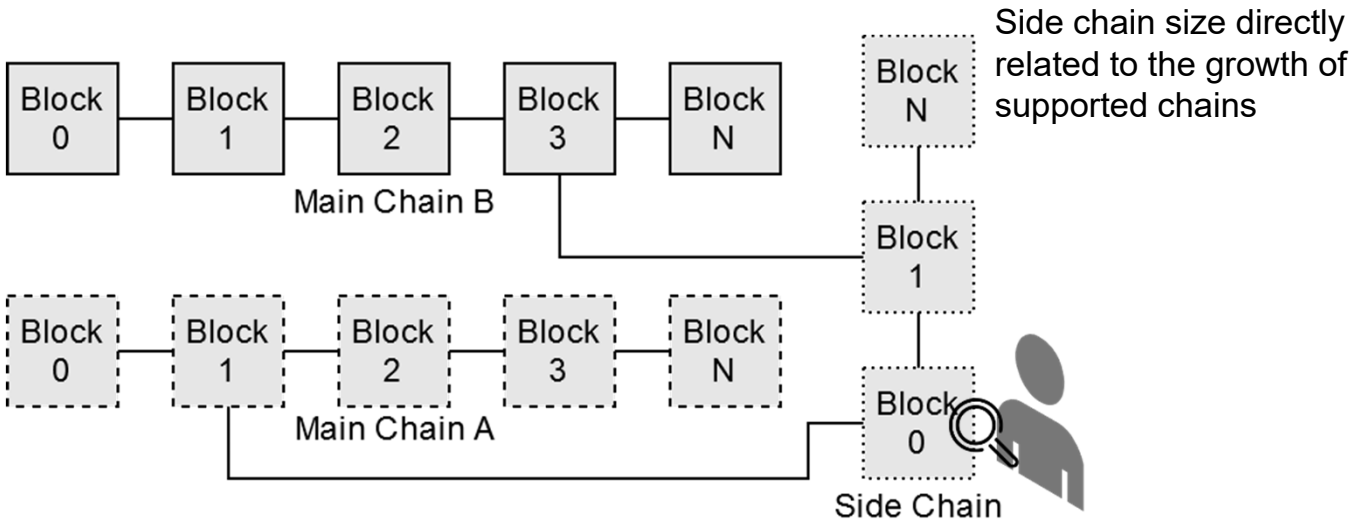
Notary Scheme

- (Re)Introduction of a Trusted Third Party (TTP)
  - Mainly centralized approach
- Notary has “connectors” to different chains
  - Able to send, retrieve, and verify transactions
- Flexibility over security
  - Security relies on the honesty of the TTP



Sidechain/Relay

- Validates and gather information from other chains
  - Mainly decentralized approach
- Requires the same mechanisms as BC/DLT
  - Consensus, miners, full and light nodes
- User and applications verify one side chain for trusted information from several chains



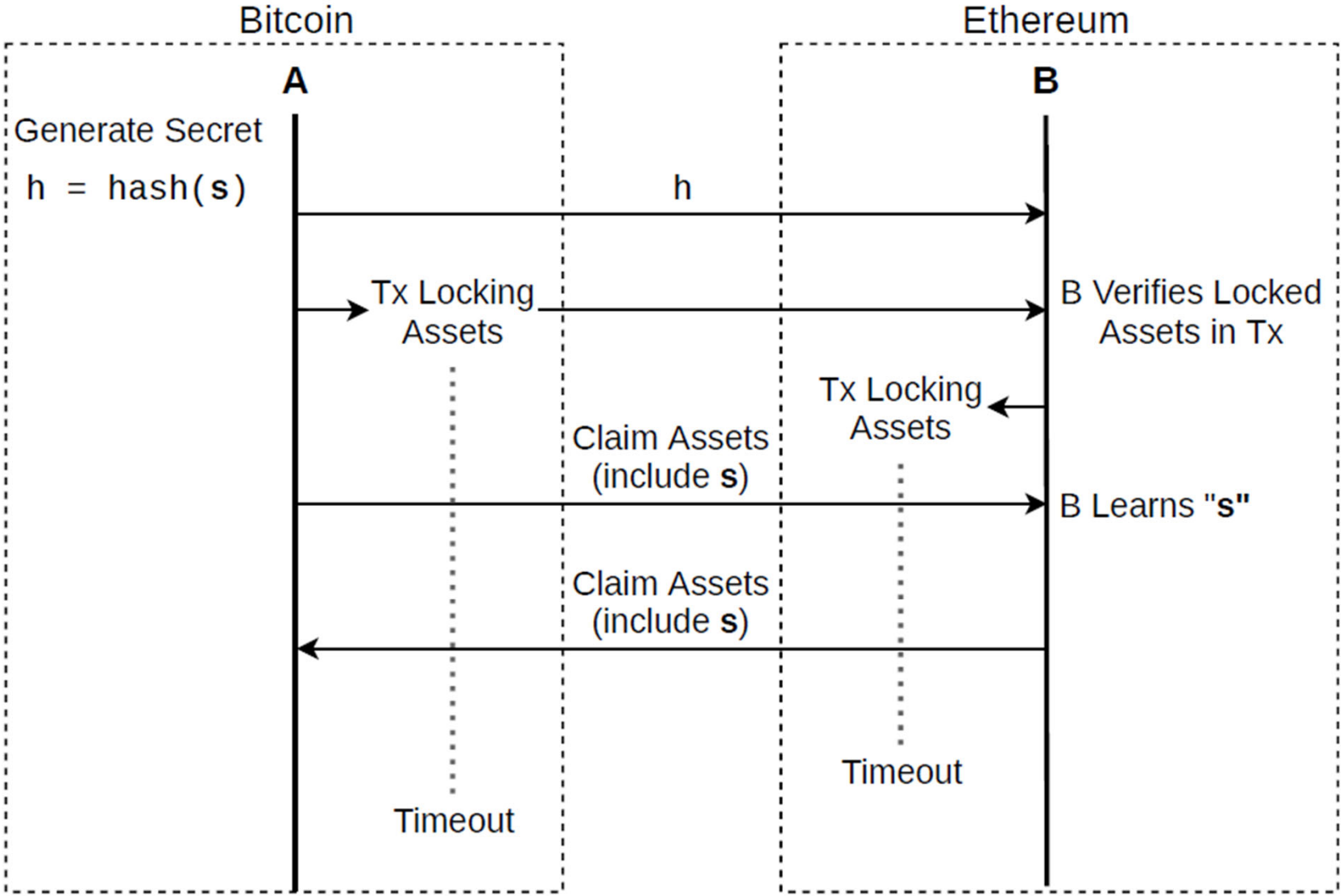
Hash Locking

- Allows the exchange of coins from different chains without TTPs
  - Decentralized and atomic scheme
- Must be supported by both chains
- Scheme:

1. A generates a secret `s` and computes the hash `H = h(s)`, which is sent to B
2. A locks their funds into a SC, which can verify if an `s` being input outputs the same `H`, then B locks theirs funds with `H`
3. On A's side, if the secret `s` is provided within `2X` seconds, then the asset is transferred to B, otherwise it is sent back to A
4. On B's side, if the correct secret (*i.e.,* the value whose hash is `H`) is provided within `X` seconds, then the asset is transferred to A, otherwise it is sent back to B
5. A reveals the `s` within `X` seconds in order to claim the asset from B's contract. However, this also ensures that B learns the secret allowing B to claim the asset from A's contract

V. Buterin, Chain Interoperability, September 2016

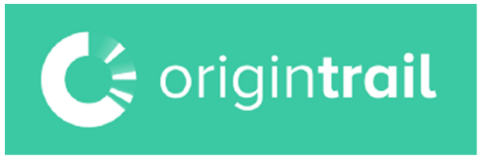
Hash Locking Sequence Diagram (Modified)



Based on V. Buterin, Chain Interoperability, September 2016

State-of-the-art

- Projects that use or provide a platform for interoperable chains include:





















BC4CC
Interoperability API

...

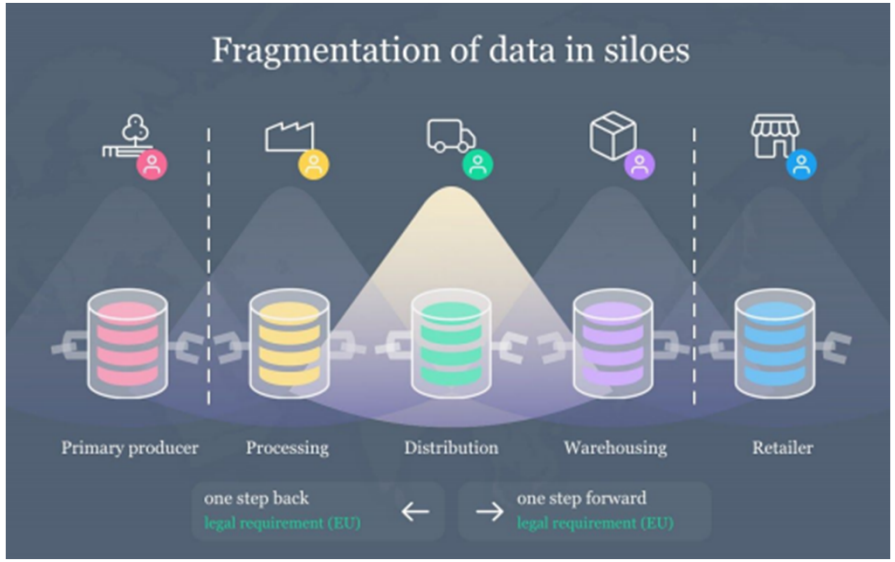
Dedicated Interoperability Approaches

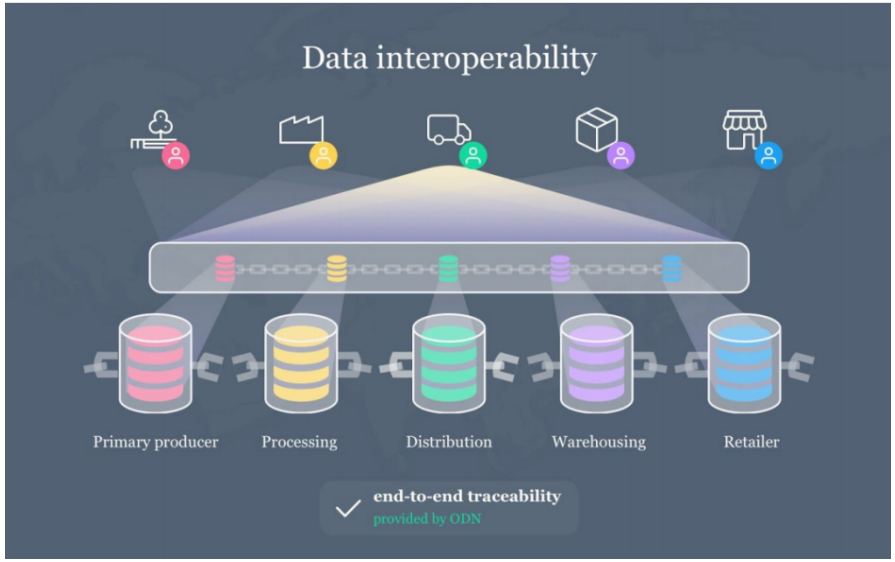
Origintrail.io

- Origintrail.io
  - Type: Sidechain/Relay
  - Description

In order to provide the optimal solution we implement the OriginTrail protocol that runs on an off-chain decentralized peer to peer network, called the OriginTrail Decentralised Network (ODN). It enables peers on the network to negotiate services, transfer, process and retrieve data, verify it's integrity and availability and reimburse the provider nodes. This solution minimizes the amount of data stored on the blockchain in order to reduce cost and inefficiency.

    - Blockchain-agnostic protocol for asset tracking in supply-chain
    - Provide a cost-effective solution to store sensitive data in a secure manner through a blockchain-agnostic approach





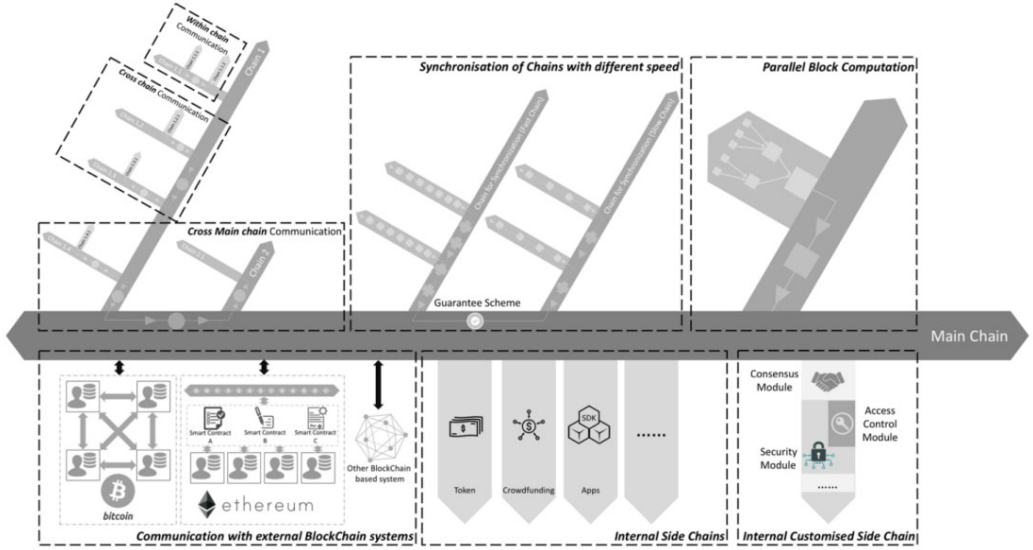
ælf (ELF)

- ælf (ELF)
  - Type: Sidechain/Relay
  - Description

4.1.2. Side Chain Dynamic Indexing

aelf is a dynamic system where all Side Chains are attached to the Main Chain. The Main Chain contains the index of the system boundaries and records where the Side Chains are. They interact with each other via the Main Chain in the form of Merkle tree verification through external information input. As such, Side Chains do not interact directly, allowing Side Chains to be added to or excluded from the aelf system.

- Crosschain blockchain protocol
- Aelf will interact with Bitcoin, Ethereum and other Blockchain systems. Cross-chain interaction with mainstream chains will be realized via messaging.

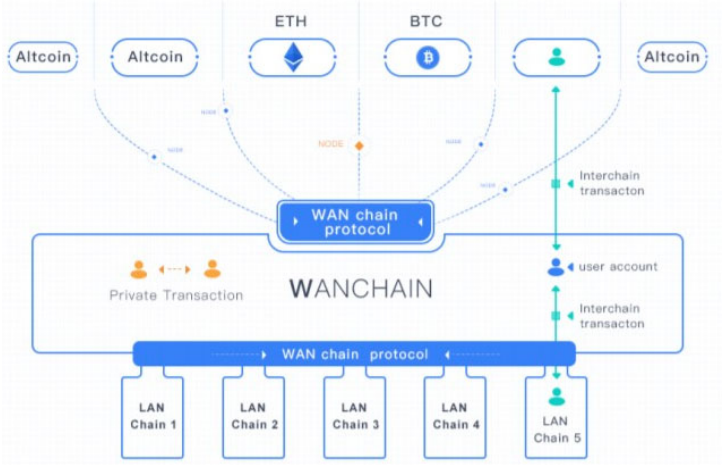


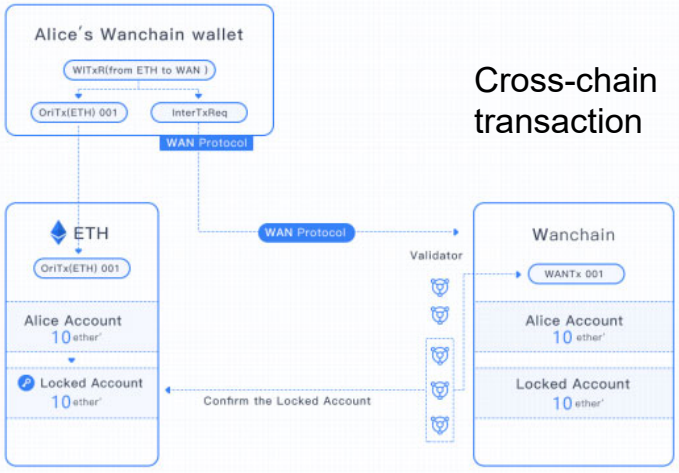
WanChain

- WanChain
  - Type: Hash-locking
  - Description
    - An Ethereum-based cryptocurrency that uses a crosschain blockchain protocol
    - Build a distributed infrastructure of digital assets connecting and exchanging value between different blockchain ledgers

For cross-chain transactions, we use secure multi-party computing and threshold secret-sharing joint anchoring schemes to achieve minimal-cost integration through the cross-chain communication protocol without

wanchain



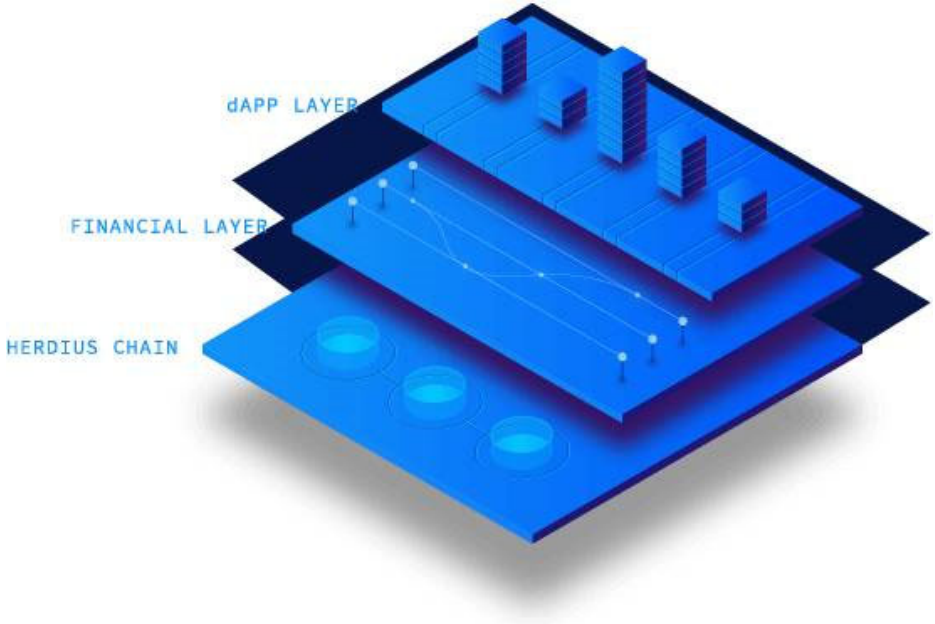


Herdius

- Herdius
  - Type: Notary-scheme
  - Description
    - Introduces “supervisors” on higher layers
    - Build a highly performing decentralized financial platform enabling users to transact assets between different chains

verification: *Supervisors* will have two minutes to inspect child blocks for traces of misconduct from validators and, if necessary, to invalidate those transactions. If validators create invalid transactions, supervisors are responsible for reporting it.

HERDIUS





© 2022 UZH, CSG@IfI

Cosmos

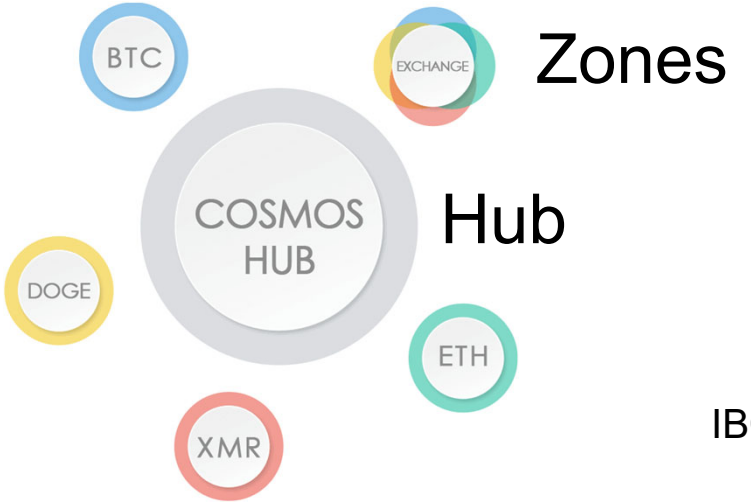
- Cosmos
  - Type: Sidechain/Relay
  - Description

Inter-blockchain Communication (IBC)

Now we look at how the Hub and zones communicate with each other. For example, if there are three blockchains, "Zone1", "Zone2", and "Hub", and we wish for "Zone1" to produce a packet destined for "Zone2" going through "Hub". To move a packet from one blockchain to another, a proof is posted on the receiving chain. The proof states that the sending chain published a packet for the alleged destination. For the receiving chain to check this proof, it must be able keep up with the sender's block headers. This mechanism is similar to that used by sidechains, which requires two interacting chains to be aware of one another via a bidirectional stream of proof-of-existence datagrams (transactions).

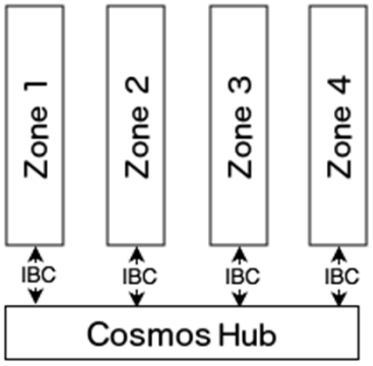
COSMOS
INTERNET OF BLOCKCHAINS

- Tendermint-based proof-of-stake BC interoperable with any other BC
- Uses the token Atom, which can be used for transaction fees on the main Cosmos hub (Zones are allowed to collect fees in any currency or combination of currencies)



Zones

Hub



IBC: Inter-blockchain Communication

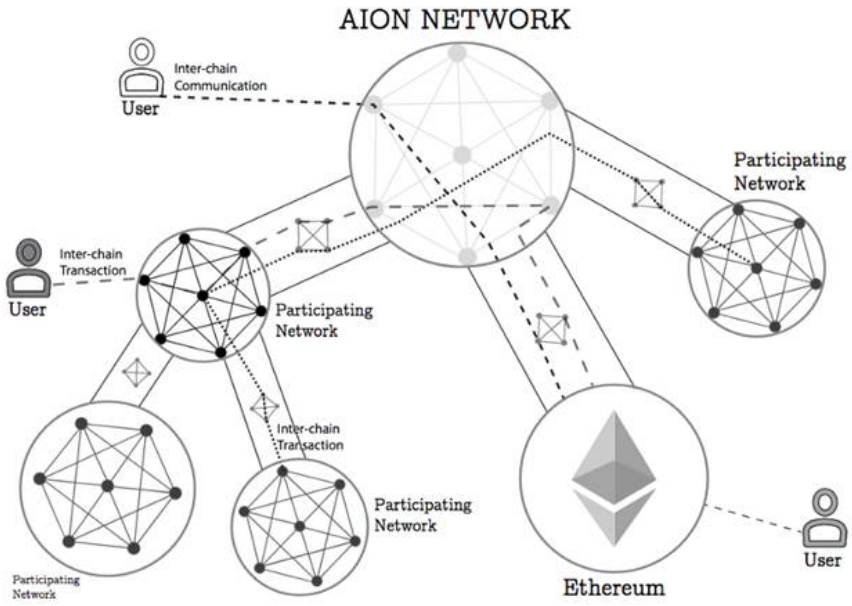
Aion

- Aion
  - Type: Sidechain/Relay
  - Description

The Third Generation Blockchain Network

Interchain transactions are initially created on a source blockchain and then processed and forwarded by bridges and connecting networks before finally reaching the target blockchain. As stated previously, the creator of an interchain transaction must pay a transaction fee for the communication cost using AION tokens, thereby incentivizing all the participants in each junction of the route.

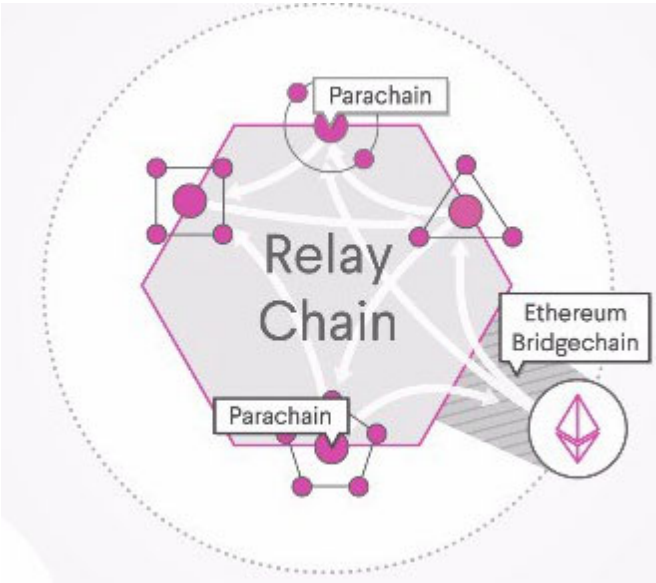
- A multi-tier blockchain system designed to address unsolved questions of scalability, privacy, and interoperability in blockchain networks



- Connecting networks: facilitate interchain communication and interchain transactions between multiple private or public blockchain networks
- Inter-chain transaction: is a trust-free message between blockchains, which allows any connected blockchain to exchange information
  - Defined format, routing, states
- Bridges: communication protocol or API responsible for registering connected chains
- Participating networks: any AION-compliant network, can be purpose-specific blockchains, private networks, or consortium blockchains representing collections of entities

Polkadot

- Polkadot
  - Type: Sidechain/Relay
  - Description
    - Heterogeneous multi-chain technology which allows new designs of blockchains to communicate and pool their security while still allowing them to have entirely arbitrary state-transition functions



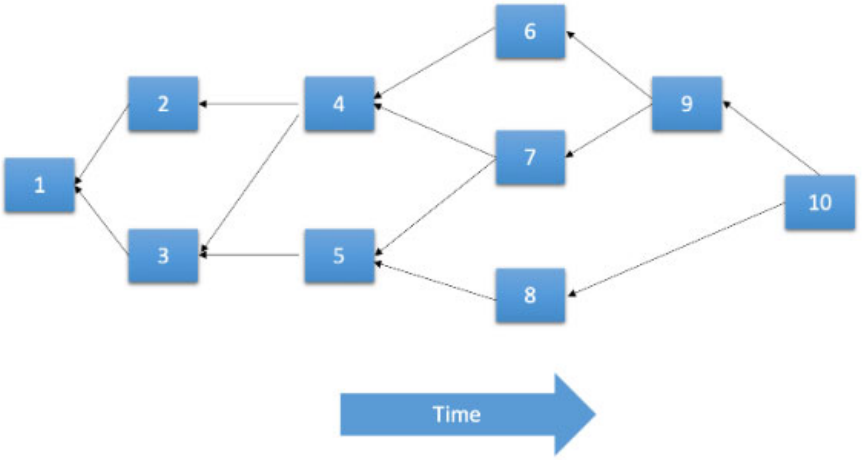
- Relay-chain: cordinates consensus and transaction delivery between chains
- Parachains: constituent blockchains which gather and process transactions
- Bridges: link to blockchains with their own consensus as such ethereum

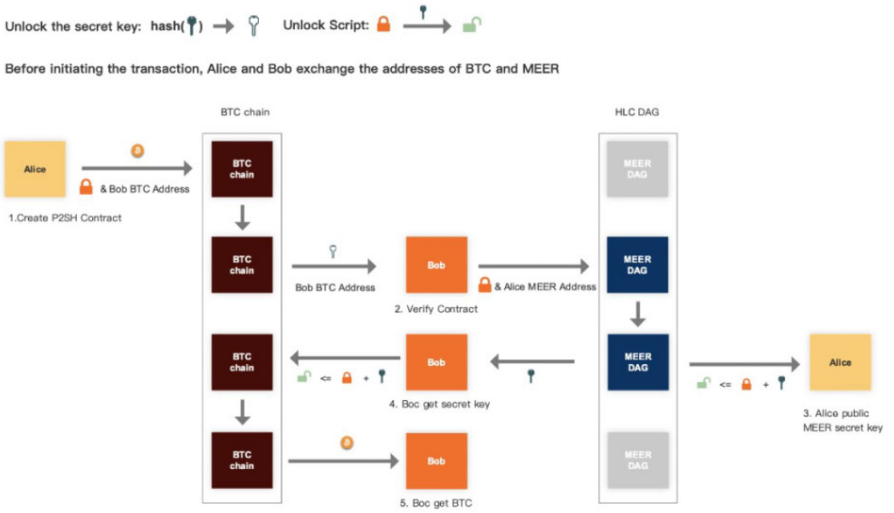
Qitmeer

- qitmeer.io
  - Type: Hack-locking
  - Description

Currently, Qitmeer has supported P2SH script contracts and cross-chain functions through hash-locking.

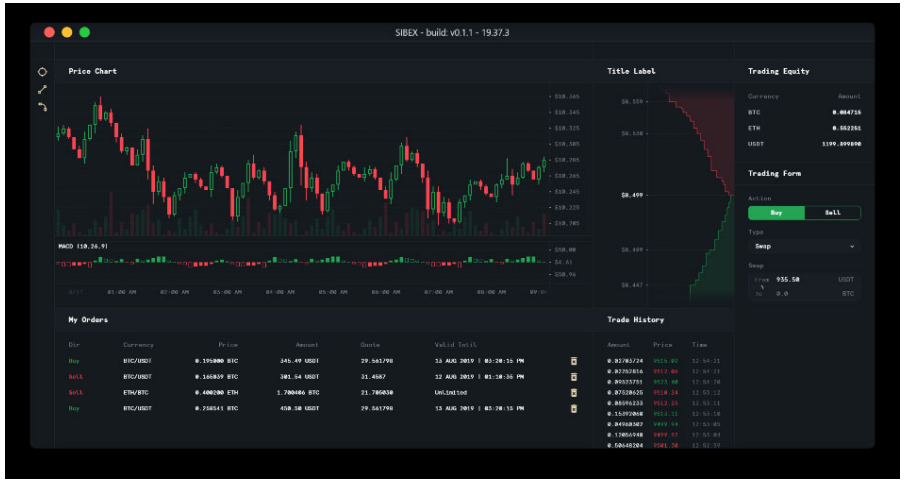
- Qitmeer is the next generation public chain based on BlockDAG which is dedicated to serving the ecosystem of Islamic Finance, ethical finance, and socially responsible investment, thereby enhancing financial inclusion and creating social impact.

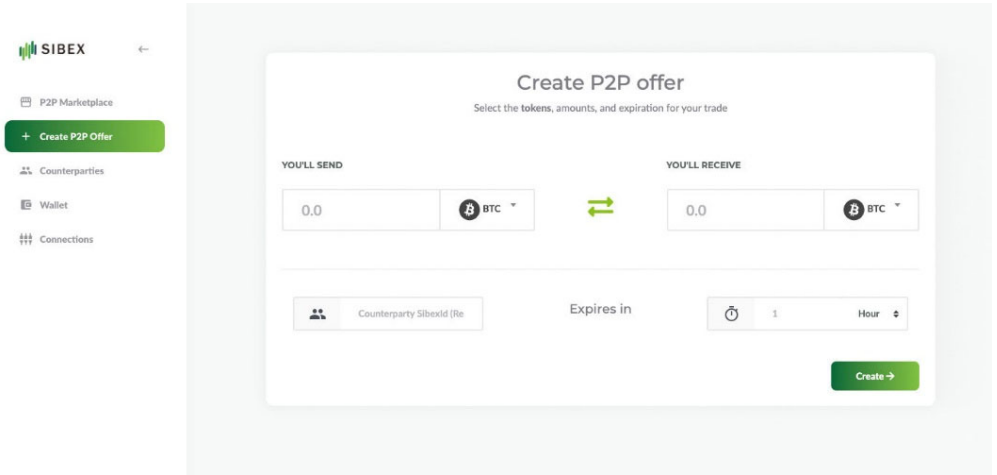




SIBEX

- sibex.io
  - Type: Hack-locking
  - Description
    - SIBEX software helps you trade Bitcoin and Ethereum using hashed time-locked contracts (HTLCs). The software helps you discover peers, communicate with them, create special Bitcoin and Ethereum transactions, and validate incoming transactions. Your private keys are always, and only, in your control.



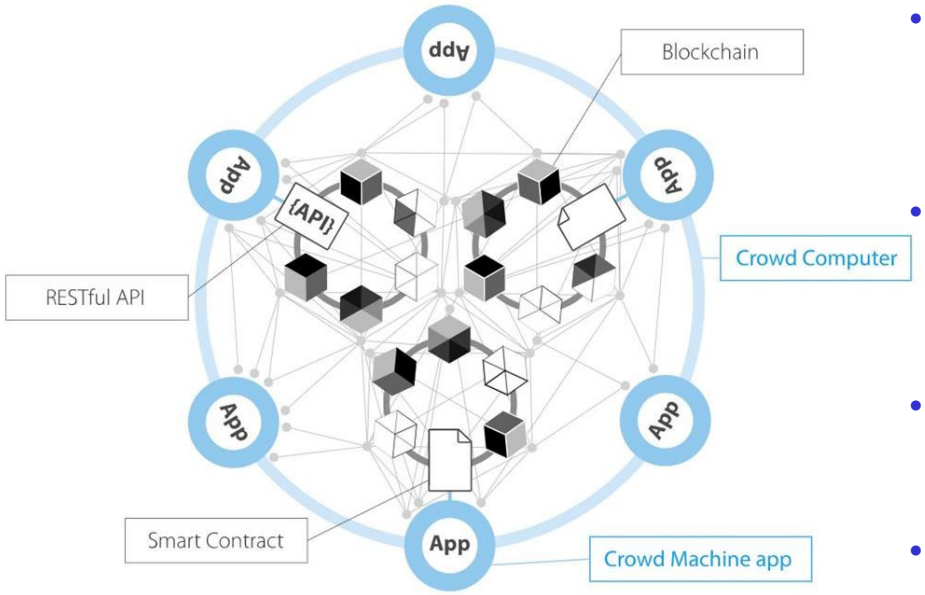


Crowdmachine

- Crowdmachine
  - Type: Sidechain/Relay
  - Goals

1. Enable any mobile device or computer owner to allow the spare capacity of their device to function as a member of a global network to run apps, and be rewarded for the use of their device; and
2. Allow developers and non-developers to create apps, without writing a line of code, that run on that global network and be rewarded every time the app is used.

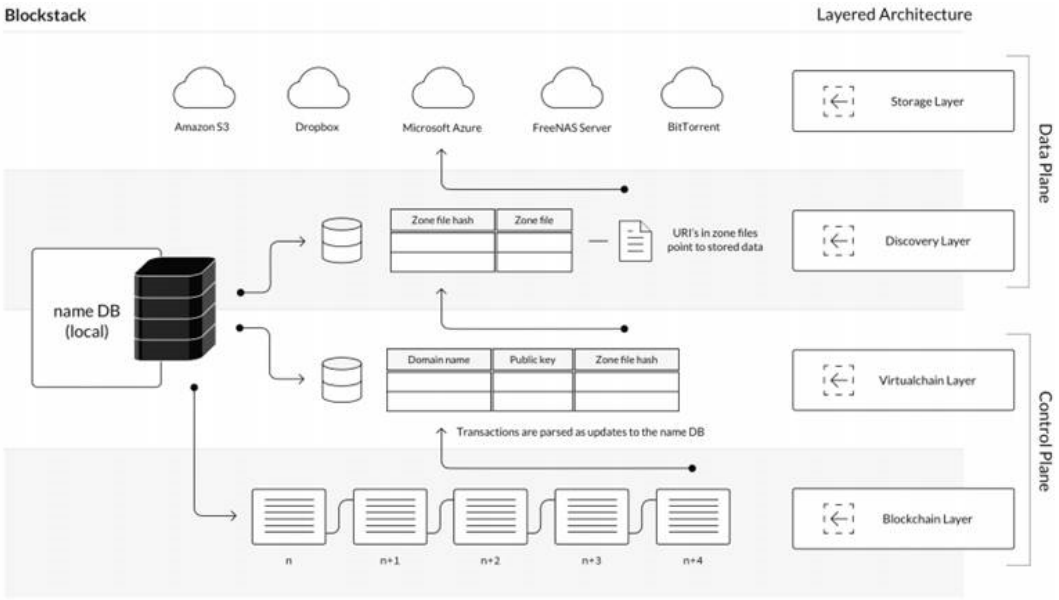
Confusing description! Distributed Cloud Computing?



- Crowd Machine: consists of the Crowd Computer, Crowd App Studio and Crowd Share, being designed to execute blockchain smart contracts and decentralized apps that meet any requirement.
- Crowd Computer: consists of a peer-to-peer network of Crowd Virtual Machines ("CVM") that run on the peer devices. Device owners are compensated for the use of their surplus processing power to run the CVM.
- Crowd App Studio: allows App feature and function to be drawn as logic diagrams, making it faster and easier to create apps than current development approaches.
- Crowd Share: a GitHub-like source repository enabling the commercialization of source code by the developer community

Blockstack

- Blockstack
  - Type: Sidechain/Relay
  - Description
    - Digital Identity service. Instead of relying on remote, third party servers for access to data and the Internet, Blockstack aims to remove those trust points and give them back to the users via blockchains.



- Storage layer: includes the actual data values and represents a part of the data plane.
- Discovery layer: Blockstack uses zone files to store routing information. The format of zone files resembles that of DNS zone files.
- Virtualchain layer: designates new operations without having to modify the underlying blockchain, which are not aware of the virtual layer
- Blockchain layer: provides the medium for storage of operations and it also provides consensus regarding the order of writing of operations

Open API (Application Programming Interface): Blockchains for Coldchains (BC4CC)

BC4CC Interoperability API

- Goal 1: enabling interaction with multiple blockchains
  - Supporting different logistic chains



- Goal 2: Providing general blockchain interoperability
  - Relying in a Notary Scheme
  - Central solution managing transactions



- Goal 3: Allowing for transparent blockchain interaction
  - Technical details hidden from users

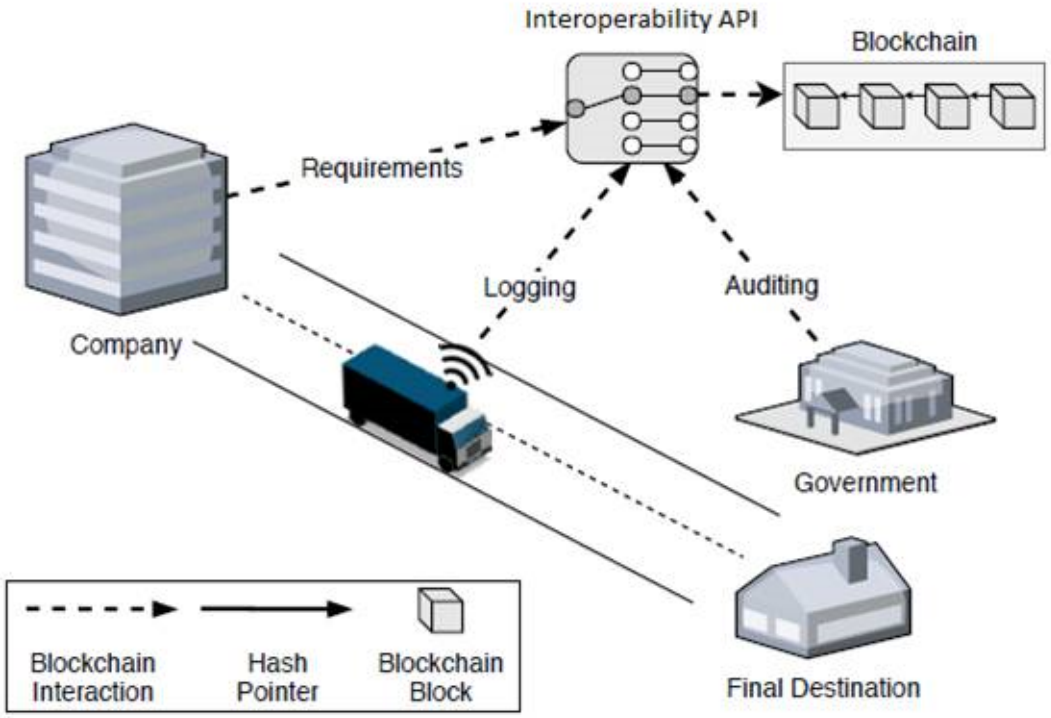


- Requirement: Storage of arbitrary data
  - *E.g.*, character strings



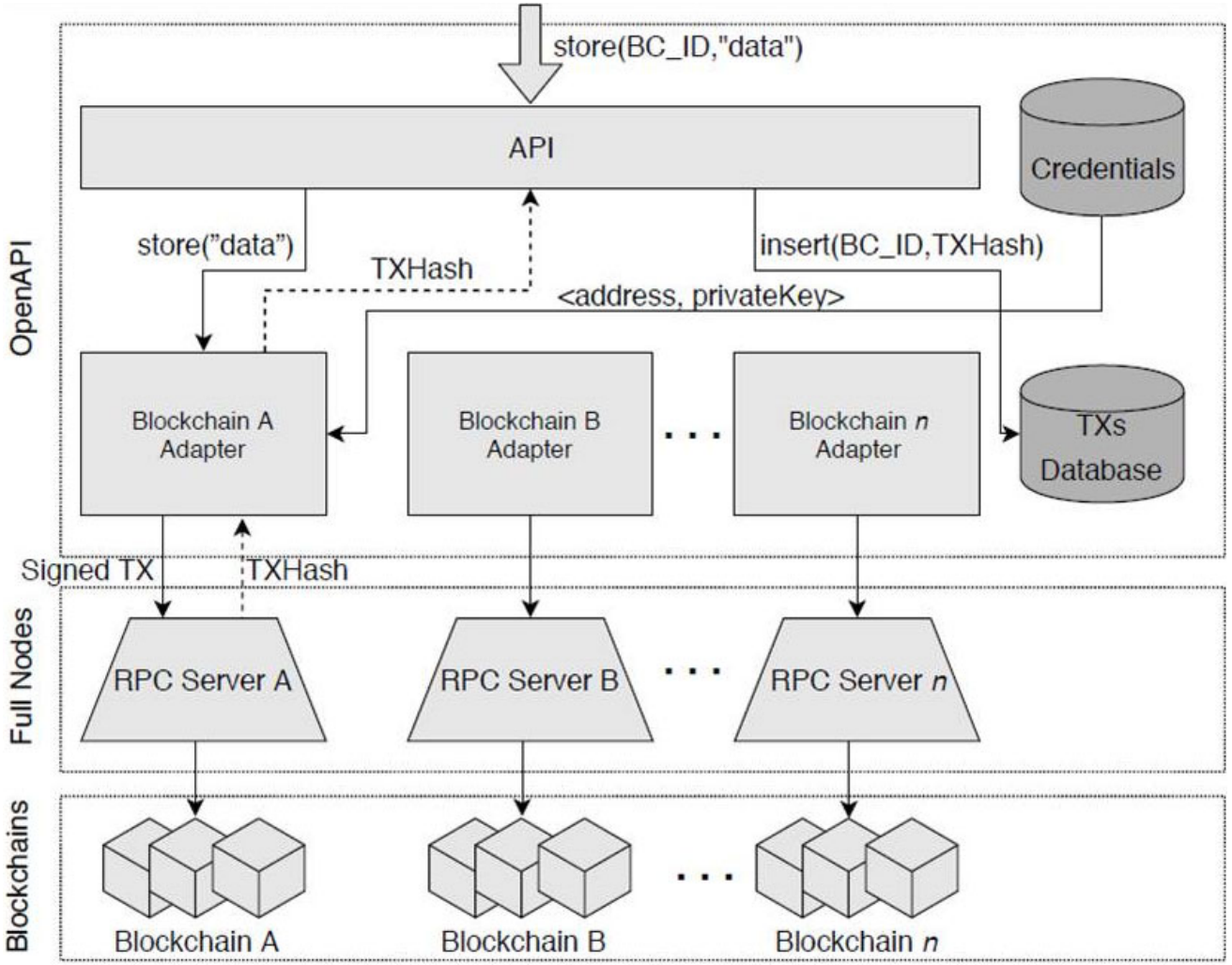
HYPERLEDGER

Application Scenario – Coldchain



- Medical drug companies
  - Standards compliance
  - *Store*: drug quality
- Transportation service
  - Temperature monitoring
  - *Store*: sensor readings
- Final destination and authorities
  - Standards verification
  - Data auditing
  - *Retrieve*: quality and temperature readings

Interoperability API – Workflow



- API Entry Point
  - No technical details
  - Implementation hidden
- Credentials DB
  - Pub/Private key
- Transactions DB
  - Hash
  - Timestamp
  - Blockchain ID
- Specialized Adapters
  - BC-specific functions

API Implementation

- User Entry-Point Example
- Interfaces implemented in the API
  - Store data interface

```
def store(text, blockchain):
    adapter = Adapter[blockchain]
    tx_hash = adapter.store(text)
    return tx_hash
```

```
tx = create_transaction("data")
signed_tx = sign_transaction(tx)
transaction_hash = send_raw_transaction(signed_tx)
add_transaction_to_database(tx_hash)
return tx_hash
```

  - Retrieve data interface

```
def retrieve(transaction_hash):
    blockchain = database.find_blockchain(tx_hash)
    adapter = Adapter[blockchain]
    text = adapter.retrieve(tx_hash)
    return text
```

```
tx = get_transaction(tx_hash)
data = extract_data(tx)
return to_text(data)
```

- Functions implemented in each blockchain adapter

Impact of Blockchain Heterogeneity

- UTXO and Account-based blockchains
  - Bitcoin Transactions != Ethereum Transactions
  - Each BTC transactions must have an input transaction to verify if the address has funds or not (unspent)
  - Whereas, Ethereum is Account-based, *i.e.,* an account holds the funds, and is updated on sending/receiving of coins
- Many other technical differences
  - Address generation mechanisms, fees, data structure…
  - RPC IP, port number, communication protocol, authentication…
- Need for specialized blockchain adapters!

Blockchain Adapters

- Specialized adapters must implement
  - Transaction creation function
  - Transaction signing function
  - Communication with RPC function
- Currently implemented

Blockchain	Python Library Name
Bitcoin	python-bitcoinrpc
Ethereum	web3.py
Stellar	py-stellar-base
IOTA	PyOTA
EOS	eosjs_python
Hyperledger	sawtooth_sdk
Multichain	python-bitcoinrpc

Specialized Adapter Implementation (1)

Bitcoin Adapter – Transaction Creation [1]

```
def create_transaction(cls, text):
    input_transaction_hash = database.find_latest_transaction(Blockchain.BITCOIN)
    inputs = [{'txid': input_transaction_hash, 'vout': 0}]
    data_hex = cls.to_hex(text)
    output = cls.create_transaction_output(data_hex, input_transaction_hash)
    # Necessary so that the address is the first output of the TX
    output = collections.OrderedDict(sorted(output.items()))
    transaction_hex = cls.client.createrawtransaction(inputs, output)
    return transaction_hex
```

- Input transaction
- Unspent input
- Data encoded in hexa
- Fee estimation and change

Transaction Signing

```
def sign_transaction(cls, transaction_hex):
    parent_outputs = []
    signed = cls.client.signrawtransactionwithkey(
        transaction_hex,
        [cls.key]
    )
    assert signed['complete']
    return signed['hex']
```

- Recently created transaction
- Private key from address
- Verify if signing was completed

https://gitlab.ifi.uzh.ch/scheid/bc4cc/

Specialized Adapter Implementation (2)

Ethereum Adapter – Transaction Creation [1]

```
def create_transaction(cls, text):
    transaction = {
        'from': cls.address,
        'to': cls.address,
        'gasPrice': cls.client.gasPrice,
        'value': 0,
        'data': bytes(text, 'utf-8'),
        'nonce': cls.get_transaction_count()
    }
    transaction['gas'] = cls.estimate_gas(transaction)
    return transaction
```

- 'from' and 'to': Equal
- 'value': 0 — No funds transfer
- 'data': bytes(text, 'utf-8') — Data encoded in bytes
- transaction['gas'] = cls.estimate_gas(transaction) — Transaction fee estimation

Transaction Signing

```
def sign_transaction(cls, transaction):
    signed = cls.client.account.signTransaction(transaction, cls.key)
    return signed.rawTransaction
```

- cls.key: Private key from address
- transaction: Recently created transaction

https://gitlab.ifi.uzh.ch/scheid/bc4cc/

Transaction Confirmation

- Transactions (tx) might not be processed
  - Refused by the network due to low fees
  - May remain in the tx pool for unknown duration
- Tx finality
  - Number of blocks that a tx is considered immutable
  - *I.e.*, the probability of a 51% attack is close to zero

I. Weber *et al.*, "On Availability for Blockchain-Based Systems," *IEEE 36th Symposium on Reliable Distributed Systems (SRDS 2017)*, September 2017, Hong Kong, China, pp. 64-73.

- Solution
  - Wait for X amount of time before confirming tx
    - Value of X depends on the blockchain in use
      - *E.g.*, Bitcoin requires 6 blocks to ensure finality (3,600 s)
  - Replace tx with increased tx fees
    - Tx re-broadcast to increase the system's robustness

Data Size

- Blockchains not designed as a database!
  - Data size is limited (*i.e.*, non-transaction related data)
  - Possible solution: rely on off-chain storage, *e.g.*, IPFS

Blockchain	Maximum String Size
Bitcoin	80 Byte
Ethereum	46 kByte
Stellar	28 Byte
EOS	256 Byte
IOTA	1300 Byte
HyperLedger	20 Byte
Multichain	80 Byte

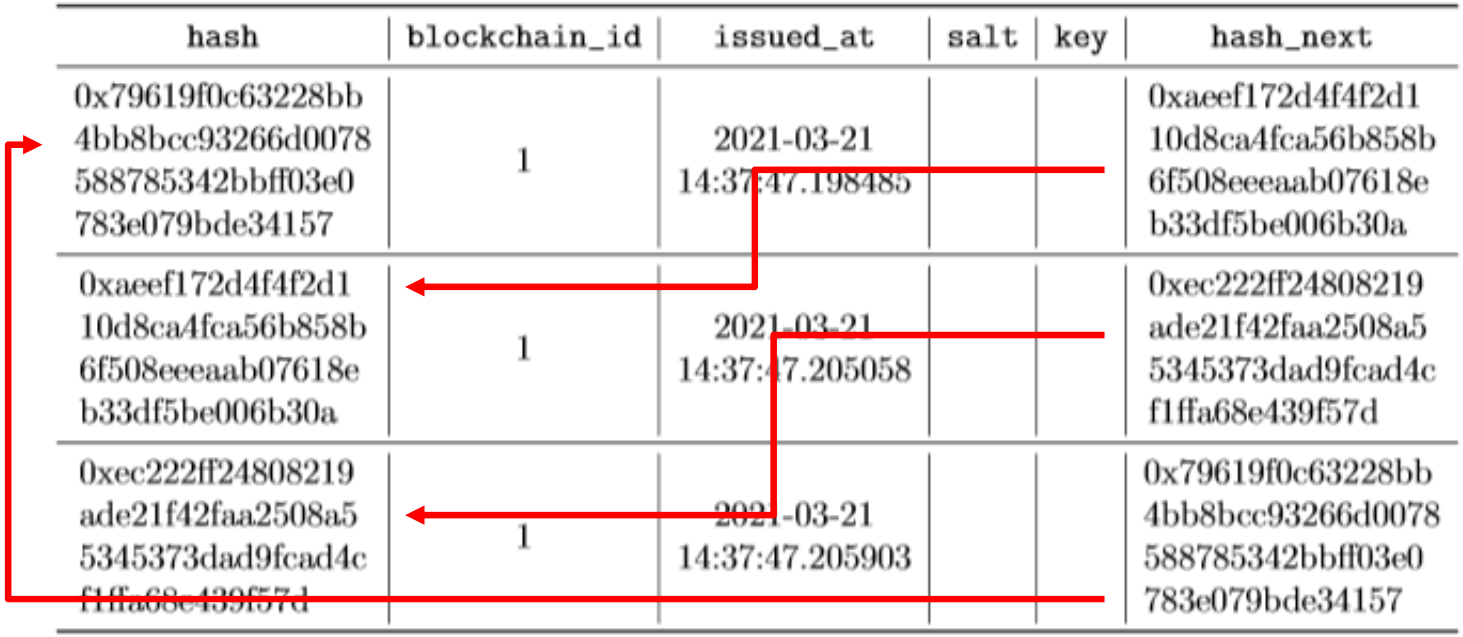


Cannot store a **SHA-256** hash (32 Byte)

IPFS: Inter-planetary File System

Data Size – Alternative Solution

- Split data in multiple transactions
  - Still requires off-chain management and is costly
  - *E.g.*, using regular database
- Store function



hash	blockchain_id	issued_at	salt	key	hash_next
0x79619f0c63228bb 4bb8bcc93266d0078 588785342bbff03e0 783e079bde34157	1	2021-03-21 14:37:47.198485			0xaeef172d4f4f2d1 10d8ca4fca56b858b 6f508eeeaab07618e b33df5be006b30a
0xaeef172d4f4f2d1 10d8ca4fca56b858b 6f508eeeaab07618e b33df5be006b30a	1	2021-03-21 14:37:47.205058			0xec222ff24808219 ade21f42faa2508a5 5345373dad9fcad4c f1ffa68e439f57d
0xec222ff24808219 ade21f42faa2508a5 5345373dad9fcad4c f1ffa68e439f57d	1	2021-03-21 14:37:47.205903			0x79619f0c63228bb 4bb8bcc93266d0078 588785342bbff03e0 783e079bde34157

Private Keys

- Blockchains based on public/private key cryptography
- Private key required to sign a transaction
  - Different address for each BC → several private keys
- Private key never leaves the server
  - RPC node receives a signed raw transaction
- However, potential attacker's target, but only option now
  - If private key stored as plain text in the DB → no encryption
  - Incentivization of invasions
    - *E.g.*, steal/ransom address funds

Basic Comparison

Blockchain	Area	Interoperability Type
Origintrail.io	Supply-chain	Sidechain/Relay
Ælf (ELF)	General purpose	Sidechain/Relay
Wanchain	Finance	Hash-locking
Herdius	Finance	Notary-scheme
Cosmos	General purpose	Sidechain/Relay
Aion	General purpose	Sidechain/Relay
Polkadot	General purpose	Sidechain/Relay
Qitmeer	Finance	Hash-locking
SIBEX	Finance	Hash-locking
Blockstack	Identity	Sidechain/Relay
BC4CC	Supply-chain	Notary-scheme

Sidechain is the most employed scheme. Hash-locking used for financial area only.

Discussed in detail

Blockchain Standardization Efforts

Necessity for Standardization

- BC Standardization → Enables interoperability
  - Networks operate the TCP/IP Model
    - Windows ↔ Linux ↔ MAC
  - BCs do not follow a defined protocol
  - No communication with the "outside" world
- However, there is no "BC protocol"
  - UTXO vs. account-based
  - Non-complete vs. Turing-complete SC
- No established standards
  - No BC "handbook"
  - No "reference" implementation/algorithm, *e.g.*, such as a Distributed Hash Table (DHT) for P2P networks



TCP: Transmission Control Protocol
UDP: User Datagram Protocol
IP: Internet Protocol

Standard Types

- Current standards can be divided in two categories
  - *De Jure* and *De Facto*
- *De Jure*
  - Originate from official standardization organizations
  - E.g., IETF's RFC 791 defining the Internet Protocol (IP)
- *De Facto*
  - Existing in reality, but are not formally recognized
  - E.g., QWERTY keyboard or HTML as Markup language
- Within the BC context *De Facto* standards include
  - ERC-20 for tokens, ERC-721 for NFTs, and BIP 39 for Mnemonic Seed Phrase

IRTF: Internet Research Task Force
RFC: Request for Comments
ERC: Ethereum Request for Comments
BIP: Bitcoin Improvement Proposal



https://learnmeabitcoin.com/technical/mnemonic
https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

Selected *De Jure* Standards

ISO: International Organization for Standardization
ETSI: European Telecommunications Standards Institute
ISG: Industry Specification Group
ITU-T: International Telecommunications Union
OAIS: Open Archival Information System
ISITC: International Securities Association for Institutional Trade Communication

- ISO/TC 307
  - Structured in 8 Working Groups (WG)
- IRTF/Decentralized Internet Research Group (DINRG)
  - Investigates BC applications and open issues
- ETSI ISG on Permissioned Distributed Ledger (PDL)
  - Focus on DLs for industry and governmental applications
- OAIS/ISITC Europe BC Working Group (WG)
  - Definition and verification of technical BC standards
- ITU-T Focus Group DLT
  - Trust-based data management frameworks
  - Standards and challenges

M. Belotti, N. Božić, G. Pujolle, S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," in *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 4, pp 3796-3838, Fourth Quarter 2019.

ISO/TC 307

https://www.iso.org/committee/6266604.html

Distributed Ledger Technologies (DLT)

- Active since 2016
- 4 Published ISO Standards (Blockchain and DLTs)
  - ISO/TR 23455:2019: Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems
  - ISO/TR 23576:2020: Security management of digital asset custodians
  - ISO/TR 23244:2020: Privacy and personally identifiable information protection considerations
  - ISO 22739:2020: Vocabulary
- 11 ISO Standards under development
  - *E.g.*, use-cases, data flow model, reference architecture

IRTF DINRG

https://datatracker.ietf.org/rg/dinrg/about/

- Active since 2017
- Research Challenges (non exhaustive)
  - Scalability - what are the problems that prevent decentralized infrastructure services from achieving global scale?
  - Trust management in decentralized communication settings
  - Applicability of distributed ledger and related technologies to different use cases and environments
  - Consensus algorithms for specific scenarios with a focus on Internet infrastructure services
  - Identification of common requirements and properties
- Until now, no formal document
  - Request For Comment or Draft

ETSI ISG on PDL

https://www.etsi.org/committee/1467-pdl

- Active since 2018
- Analyze and provide foundations for PDL operation
  - Goal: Create an open ecosystem for industrial solutions to foster the application of PDLs in different sectors
- Recent documents published in 2021
  - Requirements for SC's architecture and security
  - Research and Innovation Landscape
  - Federated Data Management
  - SCs: System Architecture and Functional Specification
- Other publications include
  - Application Scenarios, Landscape of Standards and Technologies, and Proof of Concepts Framework

W3C BC Community Group

https://www.w3.org/community/blockchain/

- Last meeting November 2017
- Mission
  - *Generate message format standards of Blockchain based on ISO20022 and to generate guidelines for usage of storage including torrent, public blockchain, private blockchain, side chain and CDN. This group will study and evaluate new technologies related to blockchain, and use cases such as interbank communications.*
- ISO20022
  - Standard for data interchange between financial institutions
  - Payment transactions, settlement information, credit and debit card transactions and other financial information

OASIS/ISITC Europe BC WG

https://isitc-europe.com/isitc-europe-blockchain-working-group

- Active since 2016
- Definition and verification of standards
  - Resilience, Scalability, Security, Latency, Data, Governance, Legal, Regulatory, Software and Network

- Goal of “benchmarking” Blockchain DLT to valuable tool to assist the industry to make sound and informed decisions
- Introduction of standards that firms can apply in their particular Blockchain DLT projects help them make better decisions and make implementation easier

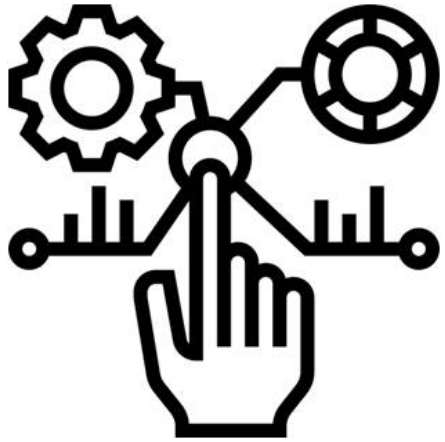
ITU-T

https://www.itu.int/en/ITU-T/focusgroups/dlt/

- Created in 2017
- Completed in 2019 with 8 deliverables:
  - DLT Terms and Definitions
  - DLT Overview, Concepts, Ecosystem
  - DLT Standardization Landscape
  - DLT Use Cases
  - DLT Reference Architecture
  - Assessment Criteria for DLT Platforms
  - DLT Regulatory Framework
  - Outlook on DLTs
- Focus on DLTs and on ITU's Telecommunication Standardization Sector (ITU-T)

Further Research Directions

- BC interaction standardization
  - Which "minimal" functions should a BC platform implement?
  - *Senders*
    - Funds, raw transaction, and address creation
  - *Getters*
    - Transactions, blocks, balances, tx inclusion, and data in tx
  - Smart Contracts (SC)
    - Call SC function, deploy SC, and events
- Data types and terminology to be used
    - Miners or validators?
    - Account type and fields (*e.g.*, to store data)
- Transaction lifecycle definition



Overall Judgment

- Fairly recent efforts (2016 onward)
- Majority of *De Jure* focus on permissioned DL standards and *De Facto* standardize BC current trends
- Standards are key for BCs and DLTs
  - Ensuring conformity → Better decision-making
  - Defined protocols → Quantification of security
  - Common interfaces → Interoperability
  - Implementation guidelines → Errors easily identifiable
- (Interoperable) standards are crucial to the adoption of BCs and DLTs in different application areas
- Need for joint work between industry and academia

Blockchain-based Remote Electronic Voting



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

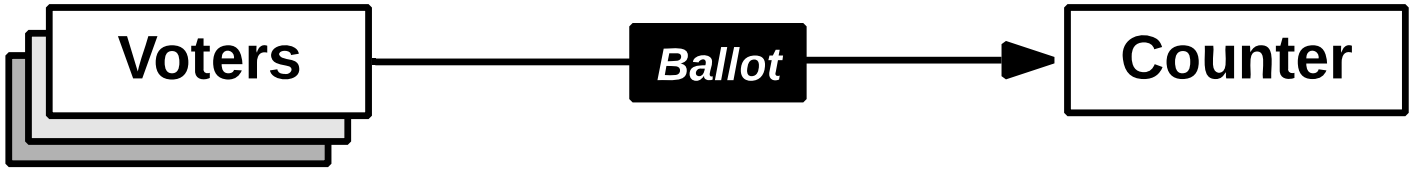
- Electronic Voting (EV)
  - Definitions and Notions with a Global Overview on EV
  - Focus on Remote Postal Voting (RPV) in Switzerland
- Remote Electronic Voting (REV)
  - REV Fundamentals & Protocol
  - General REV System Requirements & Building Blocks
    - Homomorphic Encryption, ZKP, PBB
- BC-based REV
  - Provotum 2.0 Voting Protocol and Evaluation
  - Evolution toward ProvotuMn 3.0 and Architecture
- Summary

© 2022 UZH, CSG@IfI

Electronic Voting (EV)

Definitions and Notions
with a Global Overview on EV

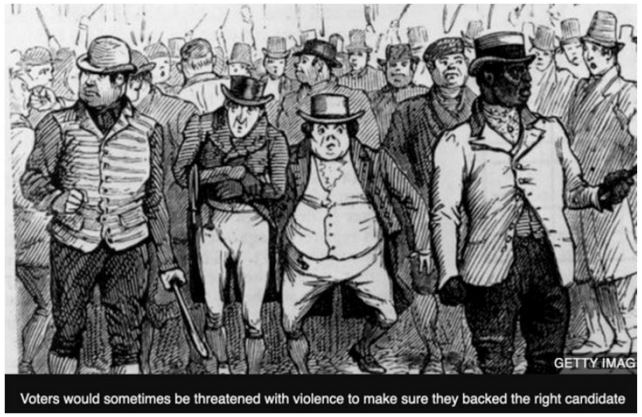
Aren‘t elections just counting?



– The need for evidence
  - Any observer should be able to verify the election data
– The secret ballot
  - Voters should not be able to prove how they voted to anyone, even if they wish to do so

The Secret Ballot

- Mid 19th century, the “Australian ballot” was introduced as a means of deterring coercion
- Today, society takes for granted this process of voting privately within a public environment, where observers can enforce privacy, but great innovation in its time
- In the early 1980s, cryptographers began to propose new election designs with new properties and integrity guarantees



Voters would sometimes be threatened with violence to make sure they backed the right candidate

https://www.bbc.com/news/uk-england-leeds-31630588



Forms of Voting in the OSCE Area

Medium	Use in a **Controlled Environment**	Use in an **Uncontrolled Environment**	Use in **Mixed Environments**
Voting with Paper Ballots	Voting with paper ballots in polling stations	Remote Postal Voting / Postal Voting	Mobile ballot box voting
Voting with NVT	Direct Recording Electronic (DRE) Voting Systems **E2**	Remote Electronic Voting / Internet Voting	Hybrid NVT: DRE Systems using Internet Voting Technology
Paper ballots and electronic counting	Ballot Scanner **E1**		Centrally Counted postal votes using ballot scanners

E1

The VotingWorks Marking Device (BMD) and Ballot Scanner run on commercial off-the-shelf (COTS) hardware. The machines were tested again and then used countywide during the general election.

https://verifiedvoting.org/



E2



The ES&S ExpressTouch DRE voting device. Currently used to provide a curbside voting option in 8 Texas counties.

https://verifiedvoting.org/

Handbook For the Observation of New Voting Technologies, ISBN: 9789292348694

E-Voting World Map

https://e-voting.cc

ABC

US voting systems Dominion sues Fox News over false claims the company rigged the 2020 election

Some Fox News employees elevated false charges that Dominion changed votes through algorithms in its voting machines. The lawsuit is part of ...

BBC Home News Sport Reel More

NEWS

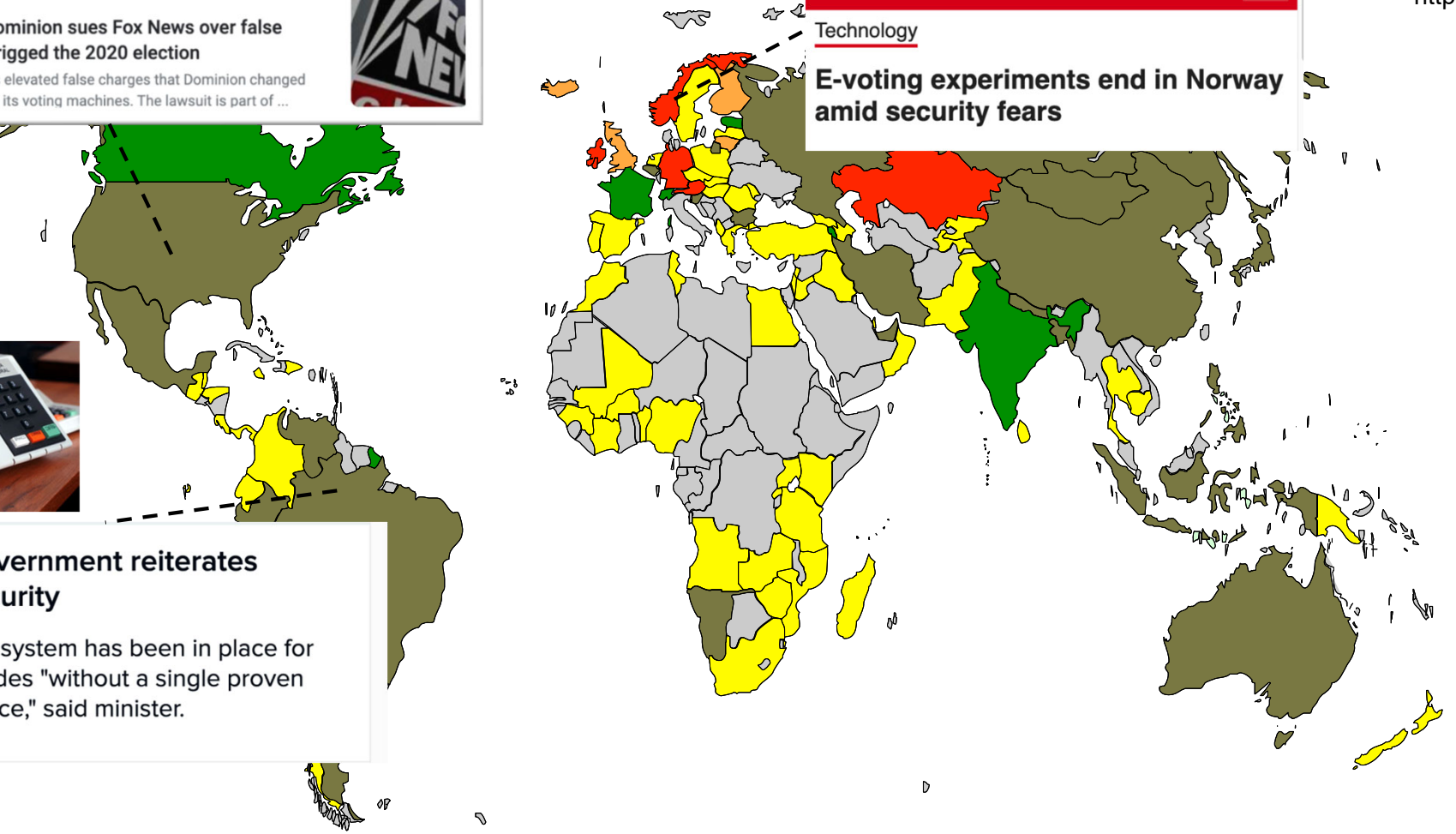
Technology

E-voting experiments end in Norway amid security fears



Brazilian government reiterates e-voting security

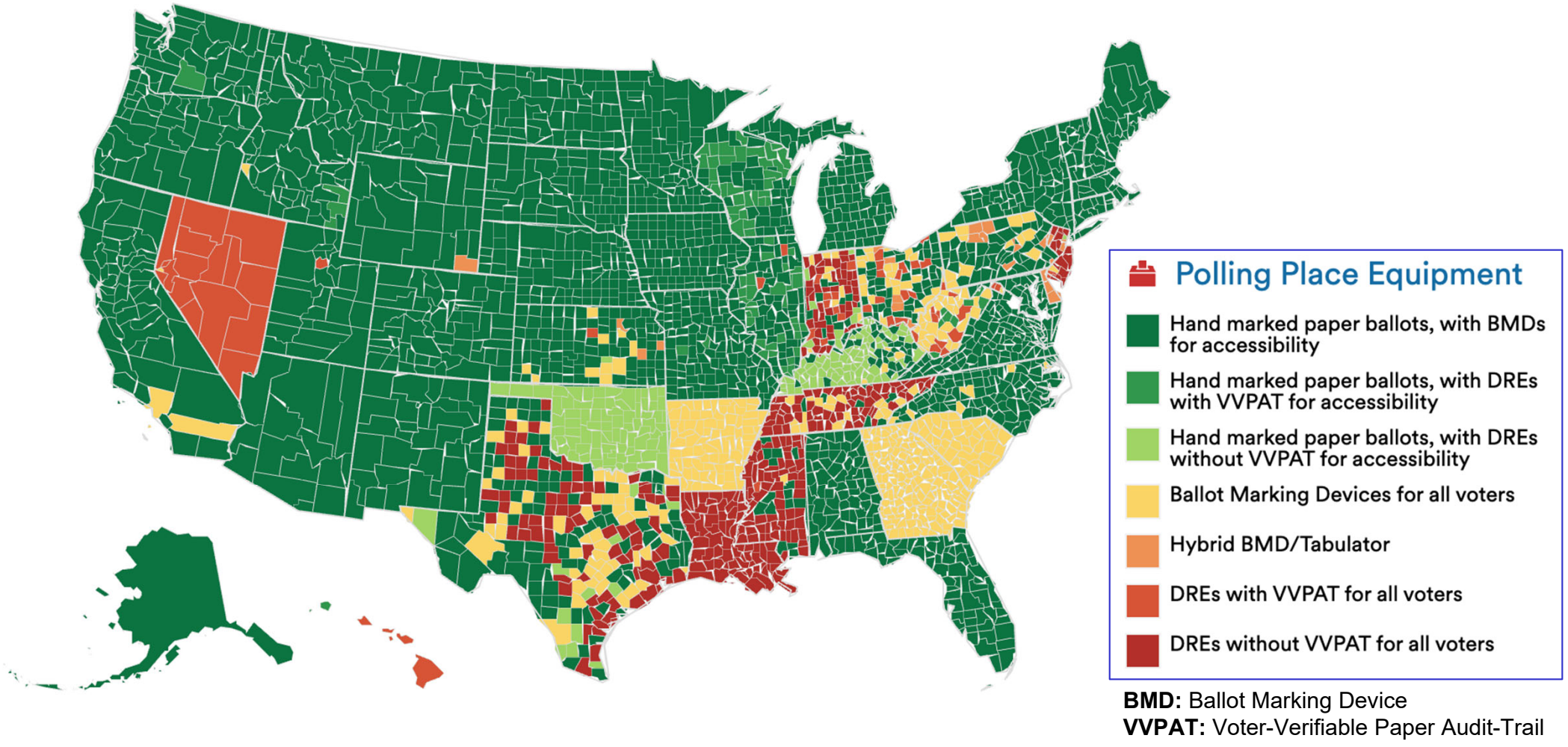
The electronic system has been in place for over two decades "without a single proven fraud occurrence," said minister.



- No plans
- Discussion with concrete plans for Internet voting
- Internet voting (legally binding) with other voting technologies
- Stopped use of voting technologies
- Discussion and/or voting technology pilots
- Ballot scanners and/or Electronic Voting Machines (legally binding)

E-Voting in the United States of America

https://verifiedvoting.org/

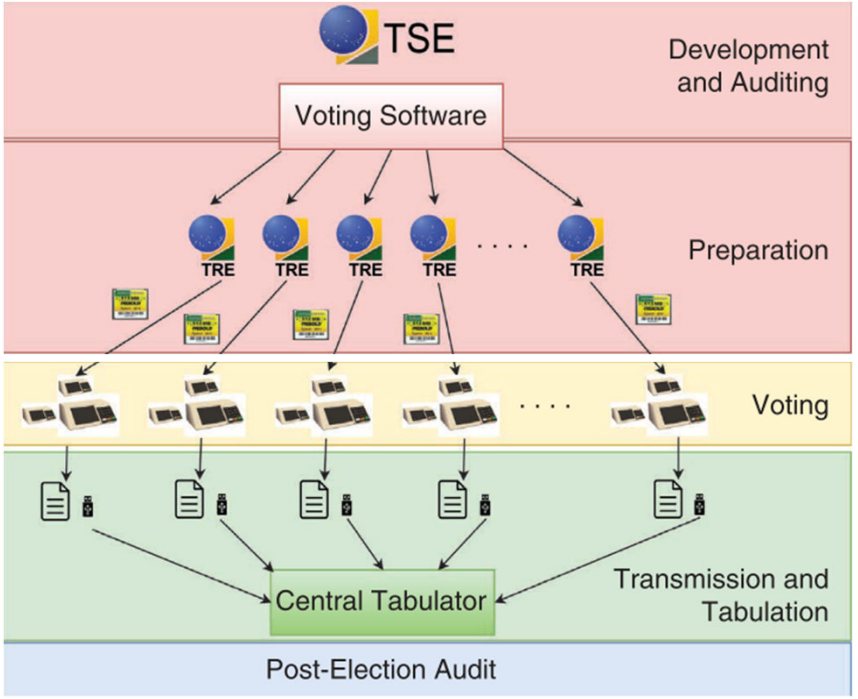


BMD: Ballot Marking Device
VVPAT: Voter-Verifiable Paper Audit-Trail

Pioneer in E-Voting: Brazil

- Brazil pioneered EV in 1996
  - Direct-Recording Electronic Voting (DRE) machines
    - Design intuition is to replicate a physical ballot box from paper-ballot voting systems.
  - Simplified voting phases
    - Development and Auditing
    - Preparation
    - Voting on DRE
    - Transmission (USB Sticks) & Tabulation
    - Post-Election Audit





https://*e-voting.cc*

Security-Critical | Less Critical | Not critical

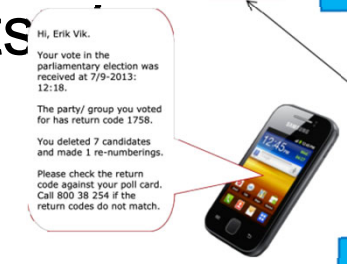
Internet Voting Trials in Norway

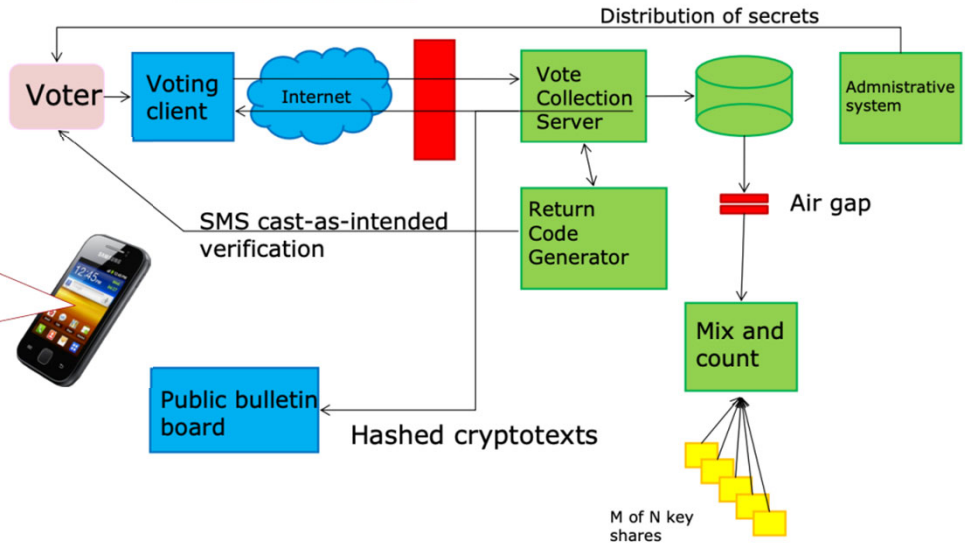
- Various NVT in use, including Internet Voting
  - Trials 2011 and 2013, discontinued in 2014
- System developed by "Scytl" and EDB Ergo Group
  - Distribution of Secrets / Postal Mail
  - Cast-as-Intended Verifiability via SMS

Medium	Use in an Controlled Environment	Use in an Uncontrolled Environment	Use in Mixed Environments
Voting with paper ballots	Voting with paper ballots in polling stations ✓	Postal voting ✓	Mobile ballot box voting ✓
Voting with NVT	Direct recording electronic (DRE) voting systems ✓ In 2015 (no VVPT)	Internet voting ✓	Hybrid NVT: DRE systems using Internet voting technology
Paper ballots and electronic counting	Ballot scanner ✓		Centrally counted postal votes using ballot scanners ✓



https://www.oas.org/es/sap/deco/seminarios/peru/pre/Henrik_Nore.pdf





Estonia‘s i-Voting REV System

i-Voting saves over

11,000

working days per election

44%

of Estonians use i-Voting

844

years of working time saved

From

2005

Estonia uses i-Voting

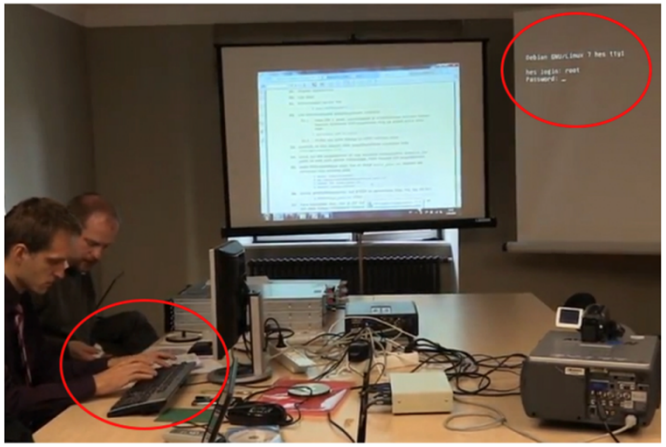


Figure 7: **Keystrokes reveal root passwords** — Videos posted by officials during the election show operators typing, inadvertently revealing root passwords for election servers.

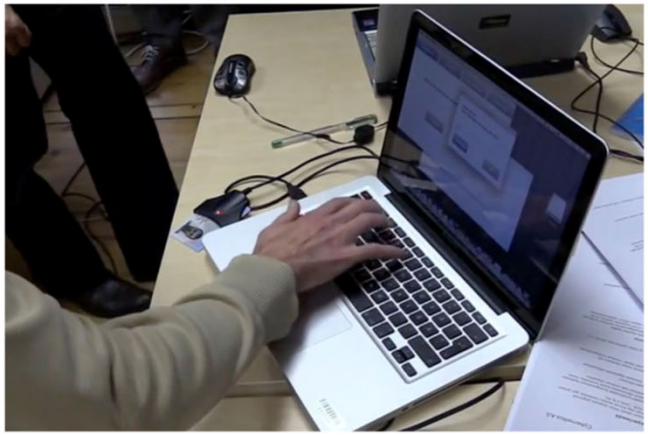


Figure 8: **Video shows national ID PINs** — During pre-election setup, someone types the secret PINs for their national ID card in full view of the official video camera.

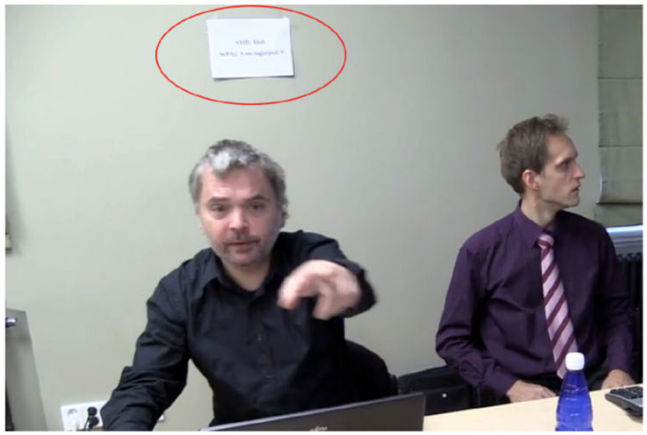


Figure 9: **Posted Wi-Fi credentials** — The official video of the pre-election process reveals credentials for the election officials' Wi-Fi network, which are posted on the wall.

Drew Springall et al.: Security Analysis of the Estonian Internet Voting System, November 2014

Electronic Voting (EV)

Focus on Remote Postal Voting (RPV)
in Switzerland

Advertisement on Public Initiative



Swiss public initiative on a "Secure and trusted democracy" in 2019

Proposed Law

Eidgenössische Volksinitiative «Für eine sichere und vertrauenswürdige Demokratie (E-Voting-Moratorium)»

Art. 197 Ziff. 12[2]

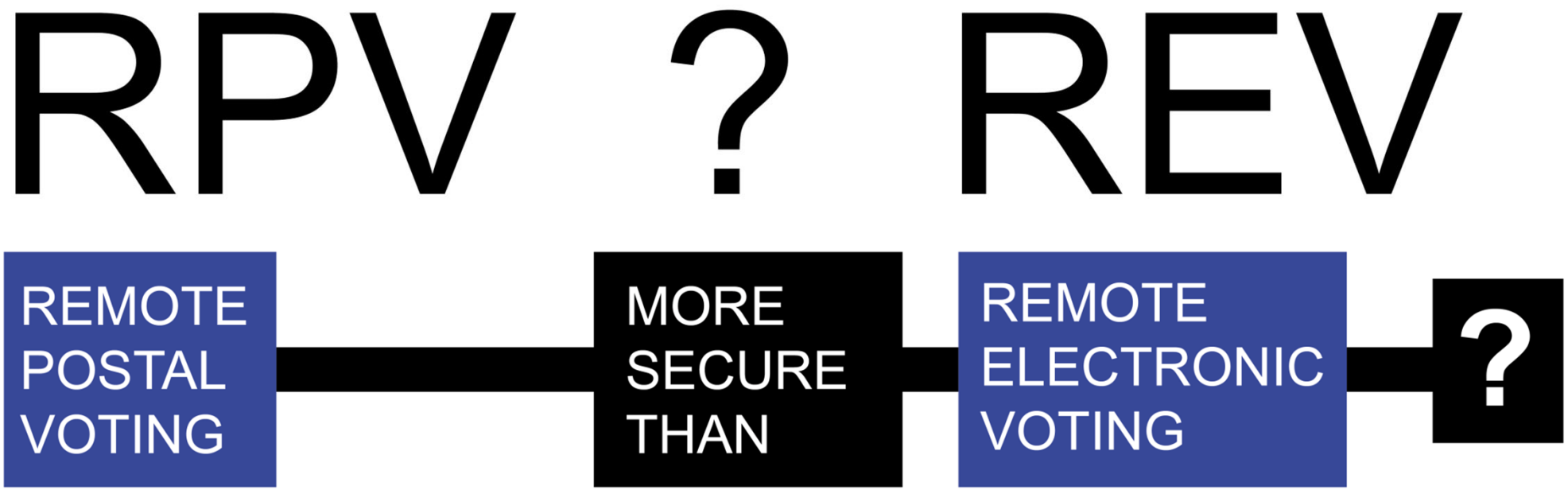
12. Übergangsbestimmung zu Art. 39 Abs. 1bis (Verwendung elektronischer Verfahren zur Stimmabgabe)

1 Artikel 39 Absatz 1bis tritt mit der Annahme durch Volk und Stände in Kraft; mit der Annahme sind sämtliche Bestimmungen des kantonalen Rechts und des Bundesrechts über elektronische Verfahren zur Stimmabgabe nicht mehr anwendbar.

2 Die Bundesversammlung kann das Verbot durch Bundesgesetz aufheben, wenn gewährleistet ist, dass mindestens die gleiche Sicherheit gegen Manipulationshandlungen wie bei der handschriftlichen Stimmabgabe besteht, namentlich wenn unter Wahrung des Stimmgeheimnisses:

*... if it is guaranteed that **at least the same security** **against manipulation exists as in the case of hand-written voting ...***

Comparing “Systems“



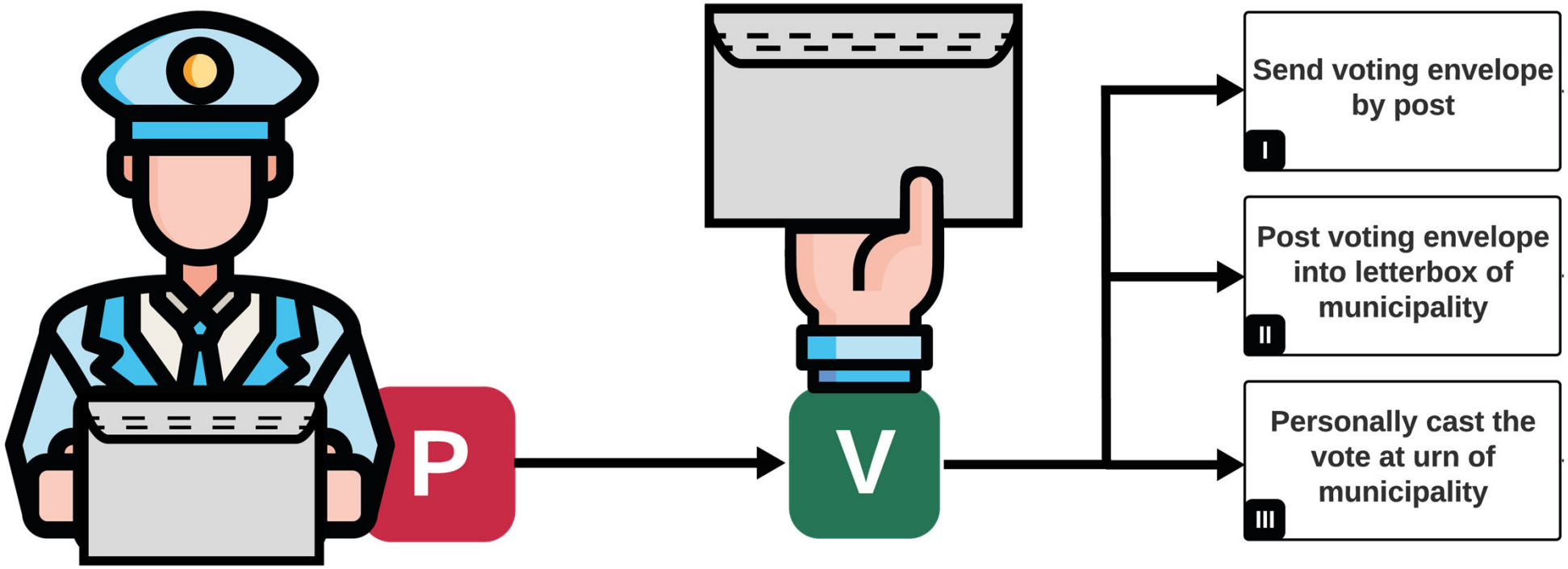
The Swiss RPV Case

- The Swiss RPV is **fragmented and difficult to generalize**, due to federalism in Switzerland, autonomy, and involvement of many external suppliers

- *Goal:*
  To identify weaknesses of RPV to allow for "hardening" of the RPV through security and risk assessment.
  - **Disclaimer:** Focus on generalization, may not cover all cantons and processes exactly, leaves room for exceptions.
  - Many exchanges with Swiss authorities and external suppliers

C. Killer, B. Stiller: The Swiss Postal Voting Process and Its System and Security Analysis, E-VOTE-ID 2019

RPV From a Voter's Perspective

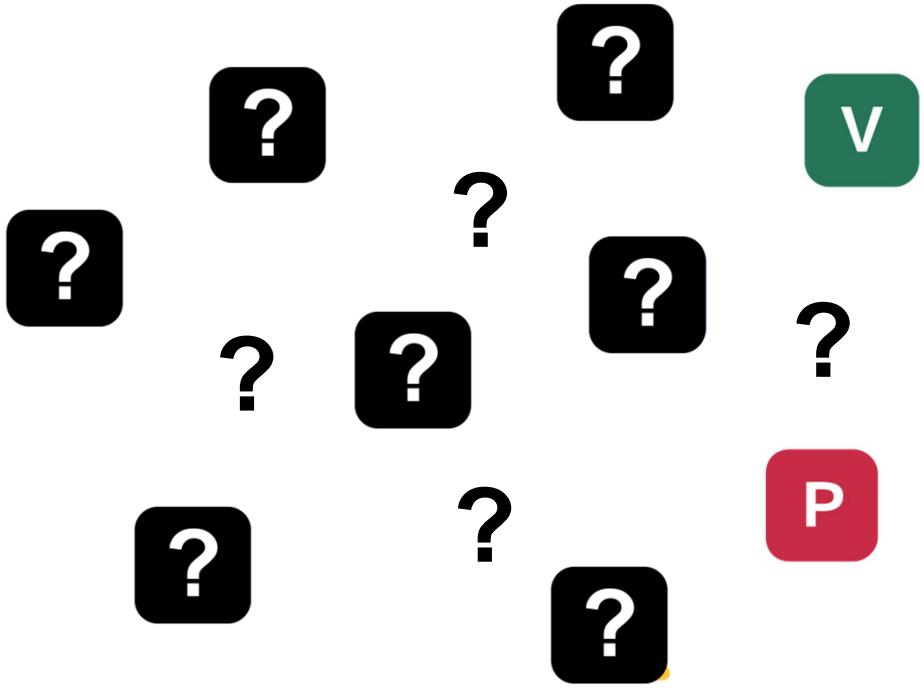


RPV: Remote Postal Voting

PVPF: Postal Voting Process Flow



Identification of Stakeholders

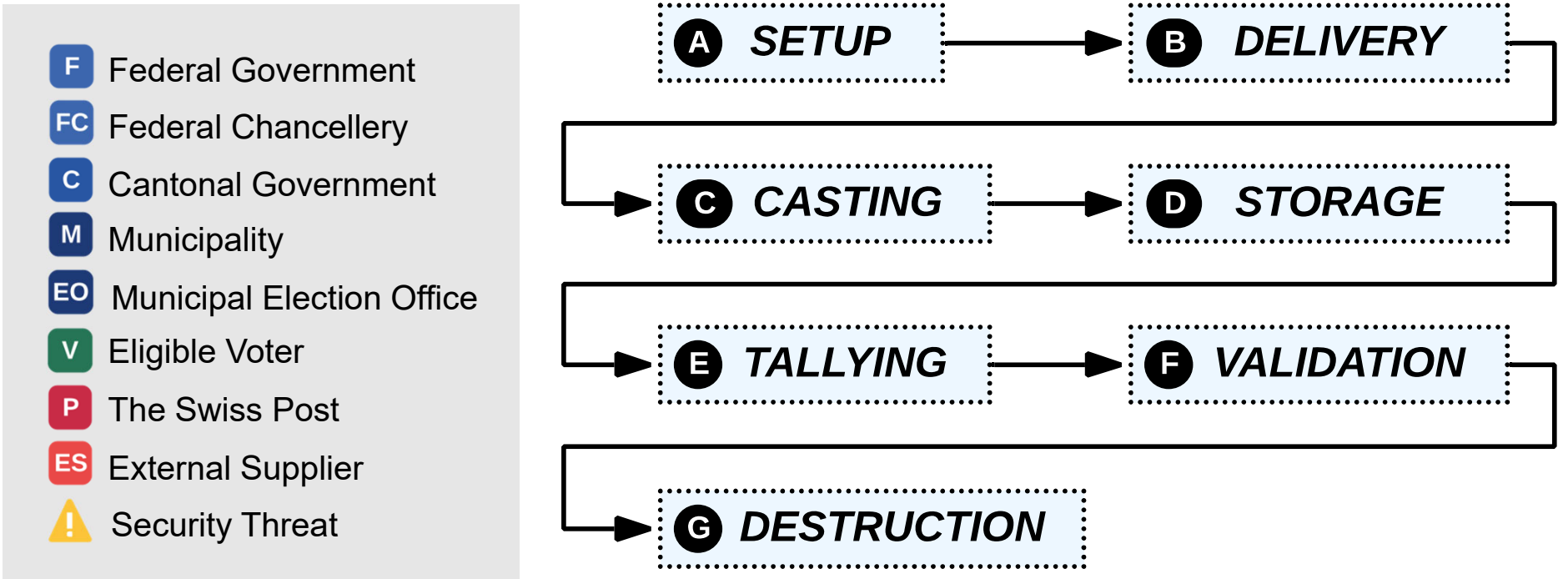


- Federal Government
- Federal Chancellery
- Cantonal Government
- Municipality
- Municipal Election Office
- Eligible Voter
- The Swiss Post
- External Supplier
- Security Threat

© 2022 UZH, CSG@IfI

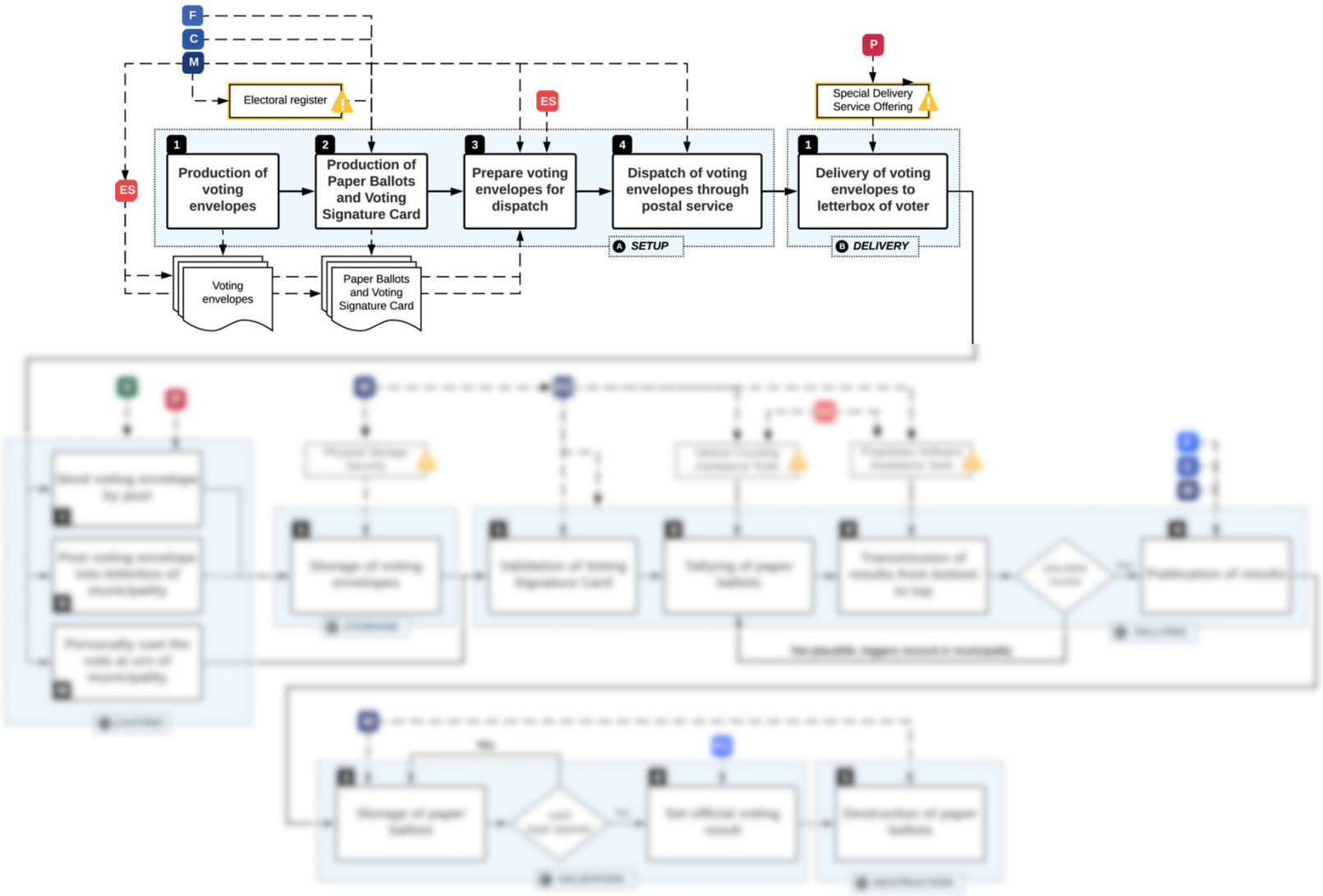
PVPF Phases

- Divided into phases A to G with various stakeholders



PVPF: Postal Voting Process Flow

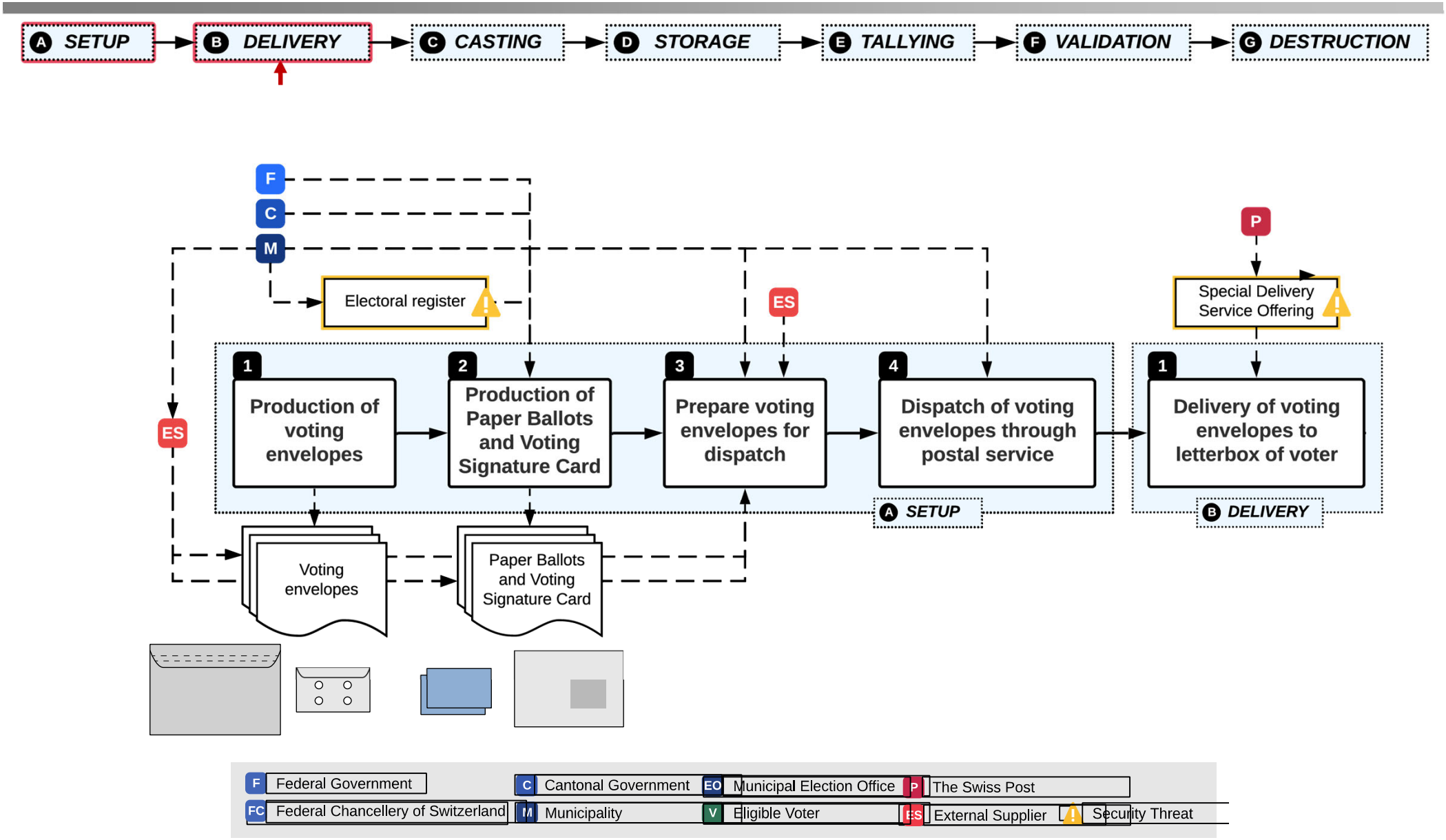
PVPF in Detail



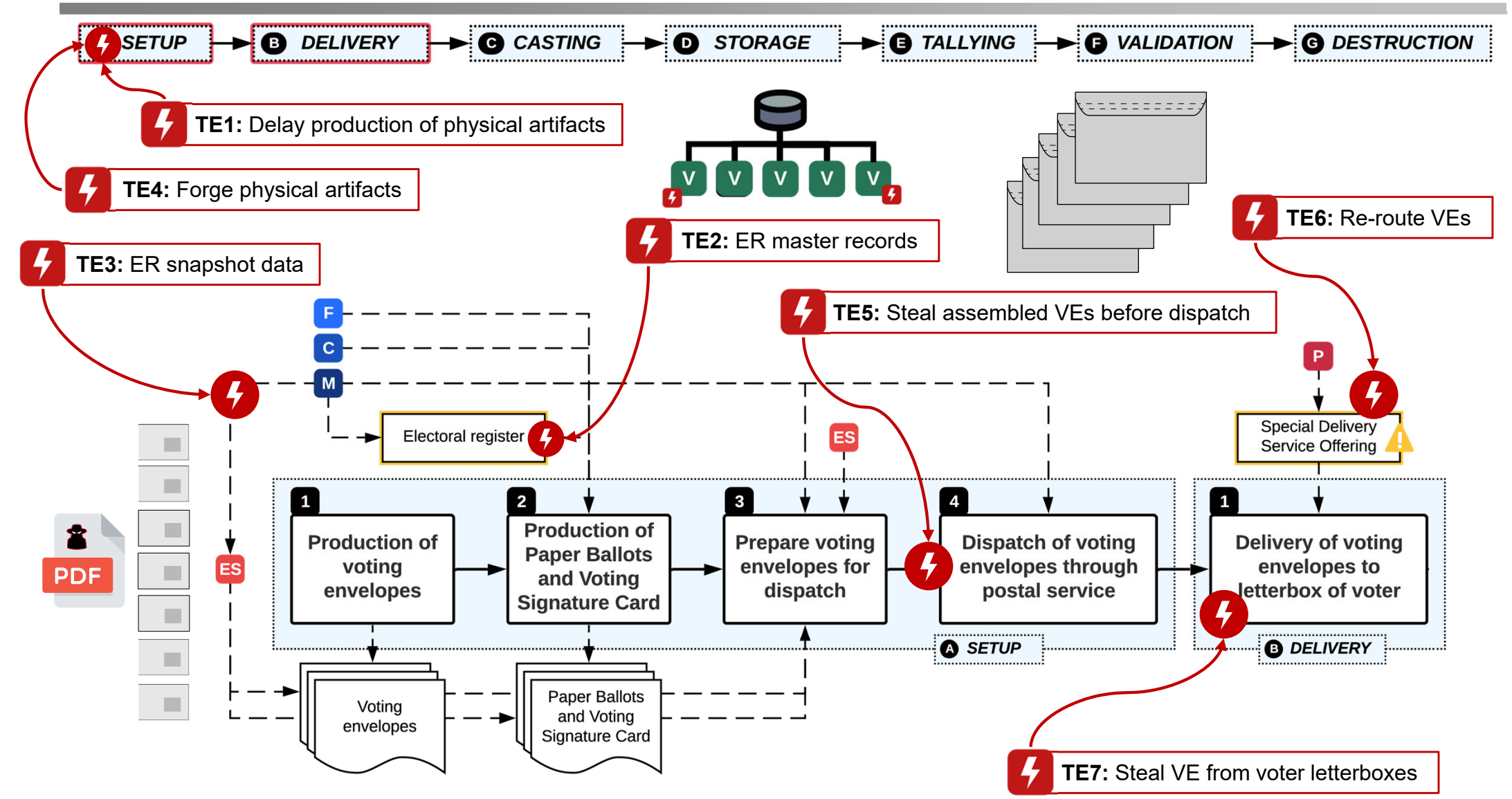
[KS19]

PVPF: Postal Voting Process Flow

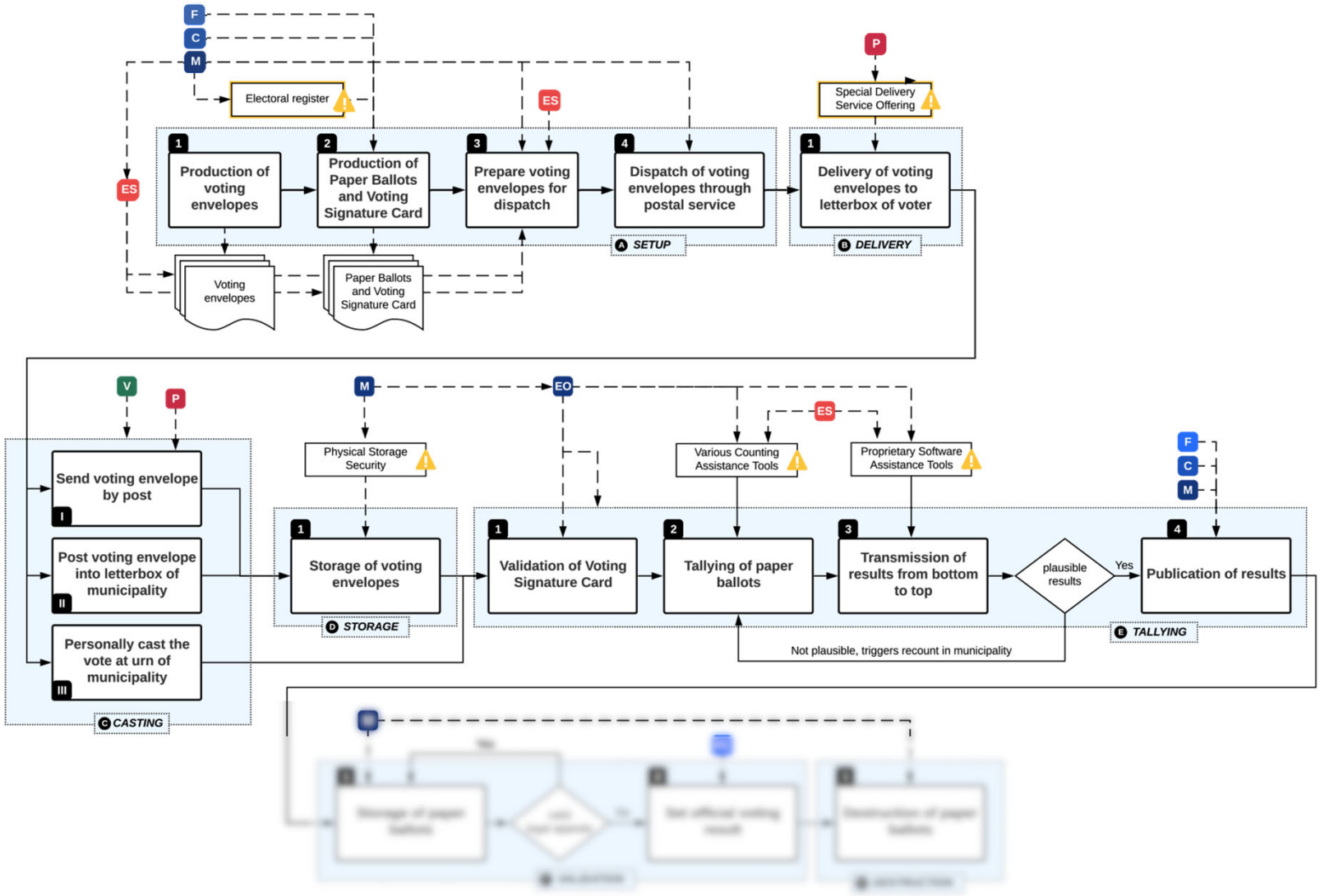
A: Setup, B: Delivery



A: Setup, B: Delivery



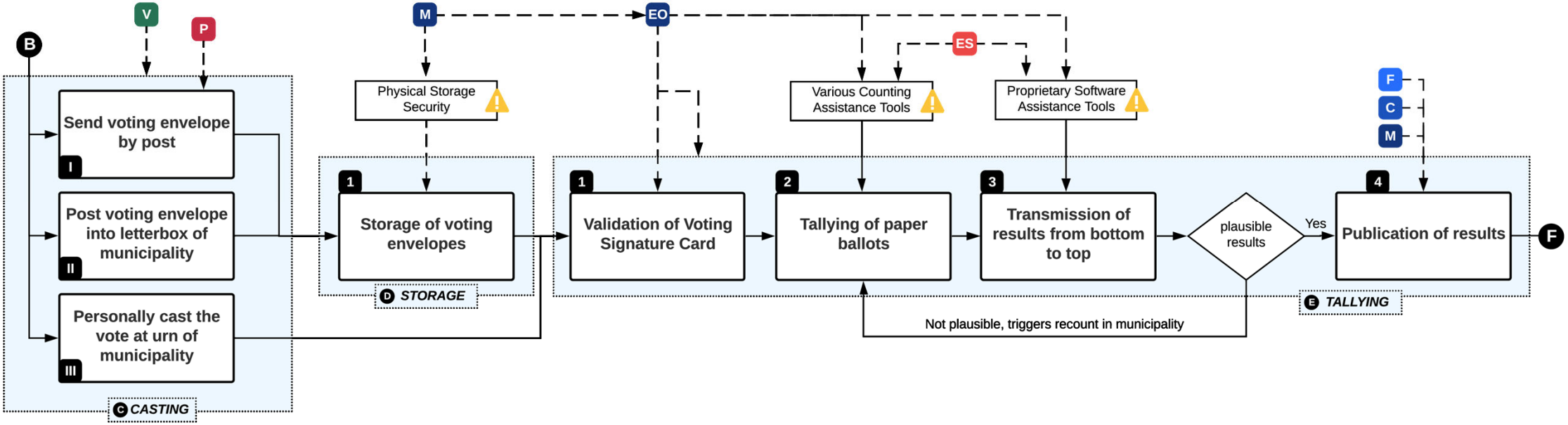
PVPF in Detail



PVPF: Postal Voting Process Flow

C: Casting, D: Storage, E: Tallying



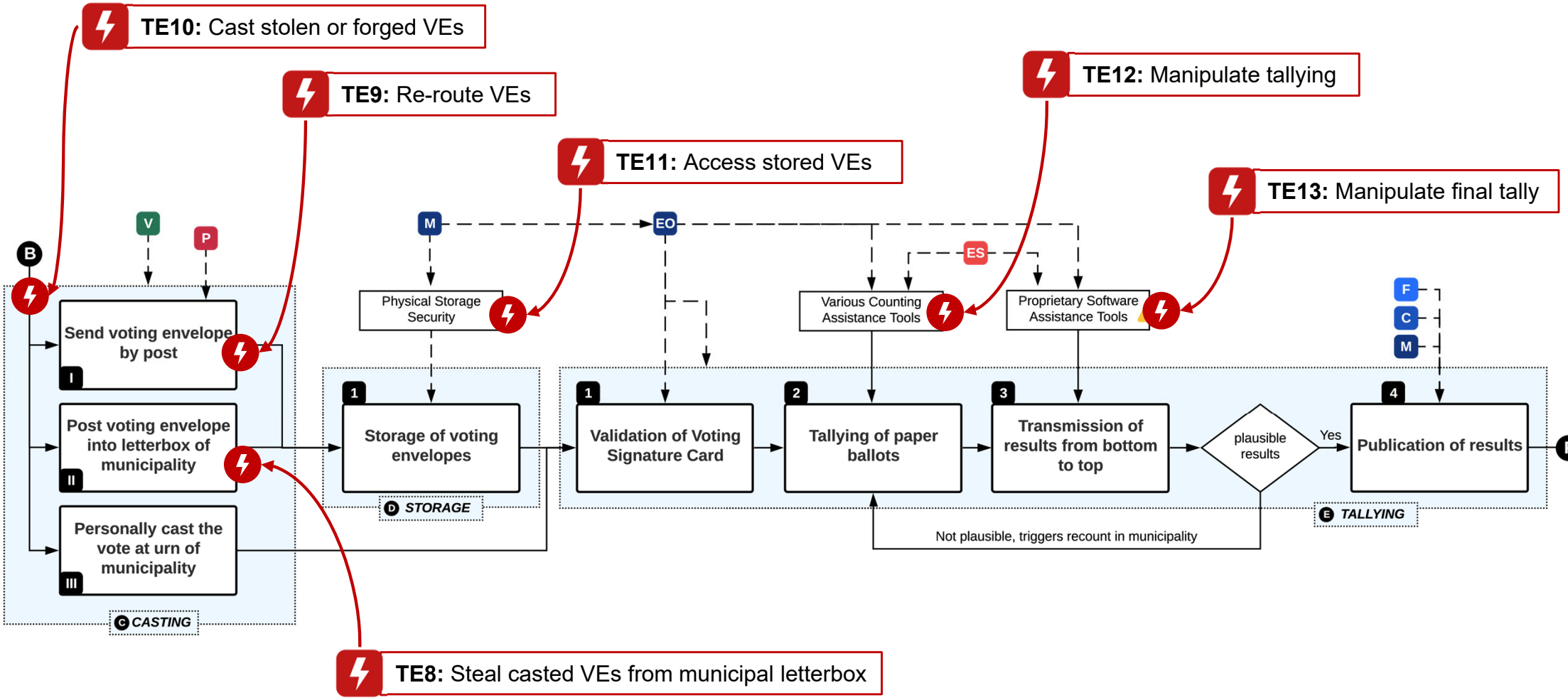




C: Casting, D: Storage, E: Tallying

THREAT EVENTS

A SETUP → B DELIVERY → C CASTING → D STORAGE → E TALLYING → F VALIDATION → G DESTRUCTION

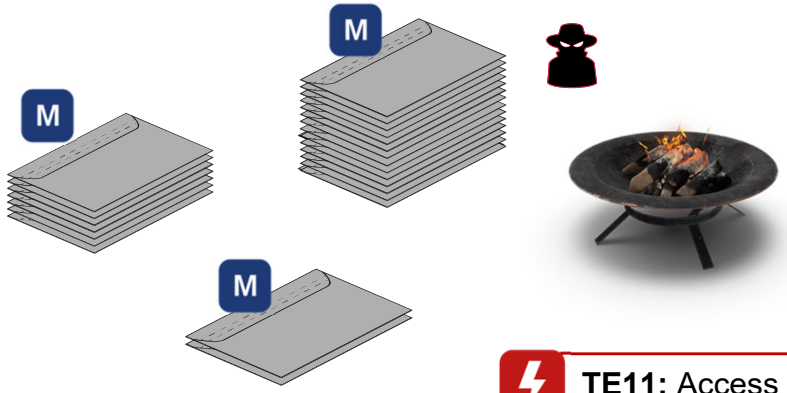


© 2022 UZH, CSG@IfI

C: Casting, D: Storage, E: Tallying

THREAT EVENTS

A SETUP → B DELIVERY → C CASTING → D STORAGE → E TALLYING → F VALIDATION → G DESTRUCTION



TE12: Manipulate tallying

TE11: Access stored VEs

TE13: Manipulate final tally

V P M EO ES F C M

B

Send voting envelope by post (I)

Post voting envelope into letterbox of municipality (II)

Personally cast the vote at urn of municipality (III)

C CASTING

Physical Storage Security

Various Counting Assistance Tools

Proprietary Software Assistance Tools

1 Storage of voting envelopes

D STORAGE

1 Validation of Voting Signature Card

2 Tallying of paper ballots

3 Transmission of results from bottom to top

plausible results

Yes

4 Publication of results

E TALLYING

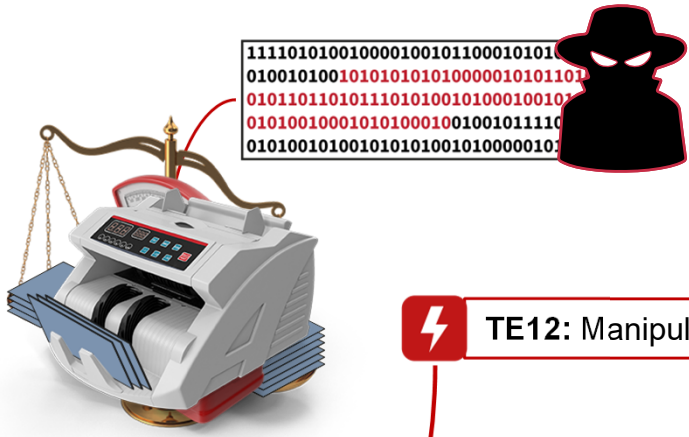
Not plausible, triggers recount in municipality

F

C: Casting, D: Storage, E: Tallying

THREAT EVENTS

A SETUP → B DELIVERY → C CASTING → D STORAGE → E TALLYING → F VALIDATION → G DESTRUCTION



TE12: Manipulate tallying

TE13: Manipulate final tally

B
V
P
Send voting envelope by post (I)
Post voting envelope into letterbox of municipality (II)
Personally cast the vote at urn of municipality (III)
C CASTING

M
Physical Storage Security
1 Storage of voting envelopes
D STORAGE

EO
ES
Various Counting Assistance Tools
Proprietary Software Assistance Tools
1 Validation of Voting Signature Card
2 Tallying of paper ballots
3 Transmission of results from bottom to top
plausible results
Yes
4 Publication of results
F
C
M
E TALLYING
Not plausible, triggers recount in municipality

F

Conclusions on the Swiss RPV

- The Swiss RPV system
  - Heterogeneous process with cantonal variations 😐
  - Substantial Trust in Third Parties ☹
  - Distribution of Trust and physical decentralization ☺

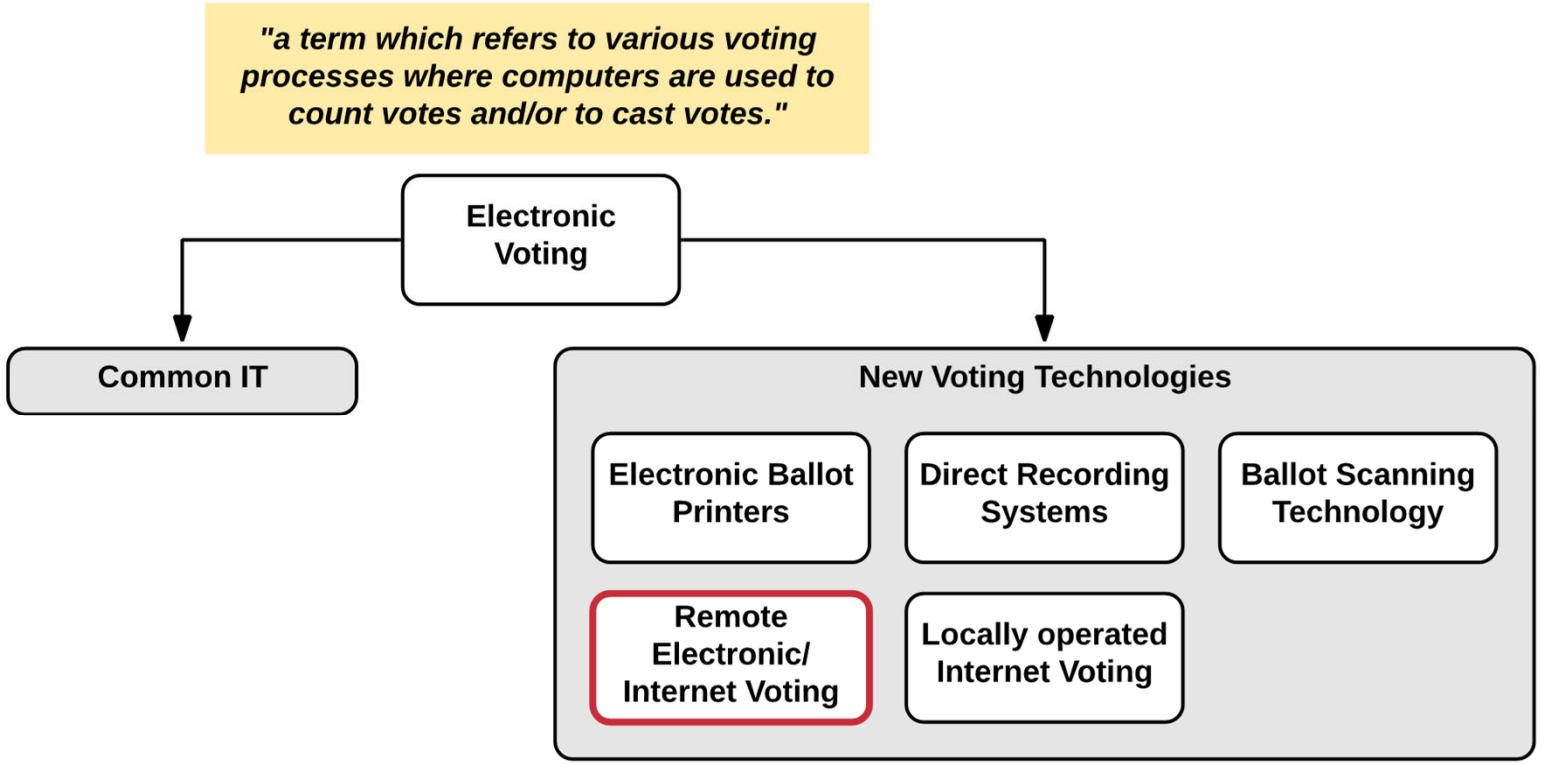
- *How can we distribute trust in the design of an electronic voting system?*
- *How can we design a fully decentralized voting system, without compromising privacy or verifiability?*

Remote Electronic Voting (REV)

REV Fundamentals
& Protocol Requirements

Terminology

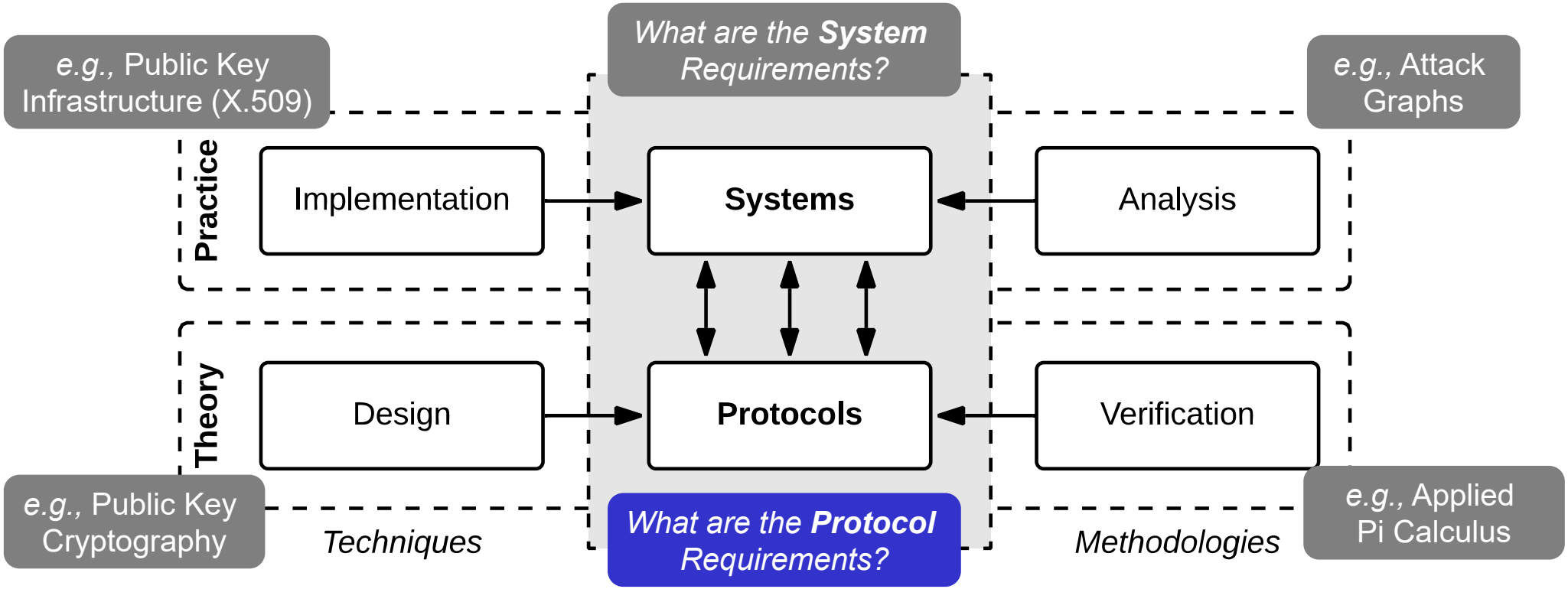
- **Remote Electronic Voting** (REV) or Internet Voting
  - Vote cast in a **remote, uncontrolled** environment and ***not*** in a voting booth.



https://www.bbc.com/news/uk-england-leeds-31630588

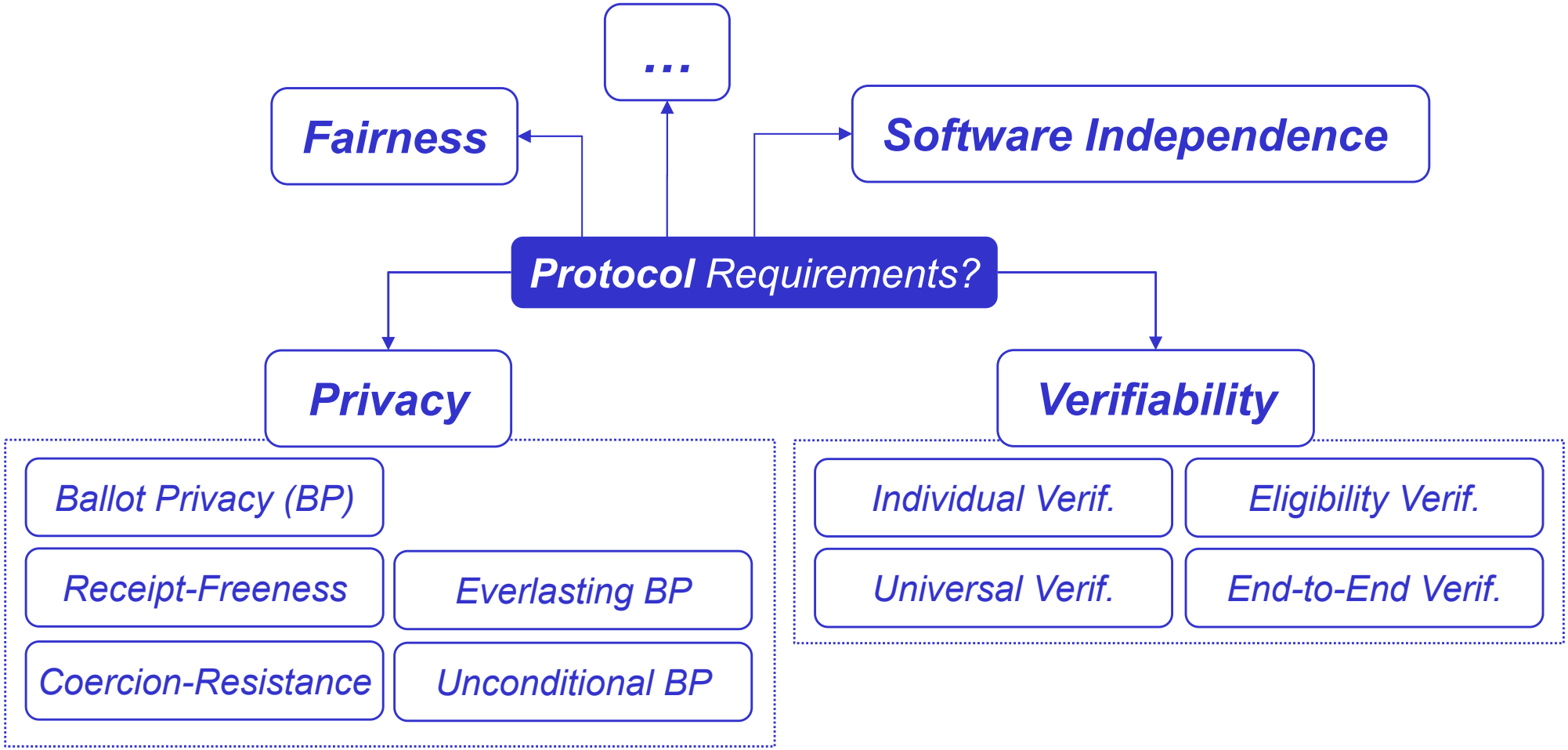
From Theory to Practice

- **From Protocols** (Theory) **to Systems** (Practice)



REV Protocol Requirements

- What are the requirements for a REV protocol?



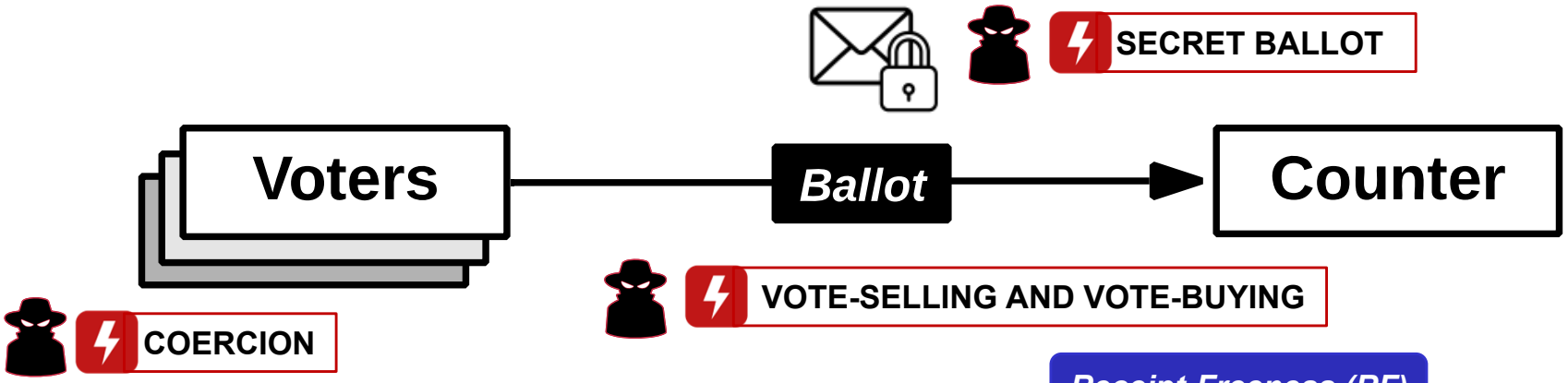
Privacy and Verifiability Trade-offs

- Privacy
  - A vote must not identify a voter and any traceability between the voter and its vote must be removed
- Verifiability
  - A voter should be able to verify if its vote was correctly recorded and accounted for in the final vote tally
- Absolute privacy (no information leaks) cannot coexist concurrently with verifiability (sufficient information leaked to verify the result)
  - Goal of voting system design is to find a way to enable as much privacy and verifiability simultaneously as possible

Privacy Properties

Ballot Privacy (BP)

*A voter's vote is **not revealed** to anyone*



Coercion-Resistance (CR)

*A voter cannot **interact with a coercer to gain information**, which can be used to prove how she voted*

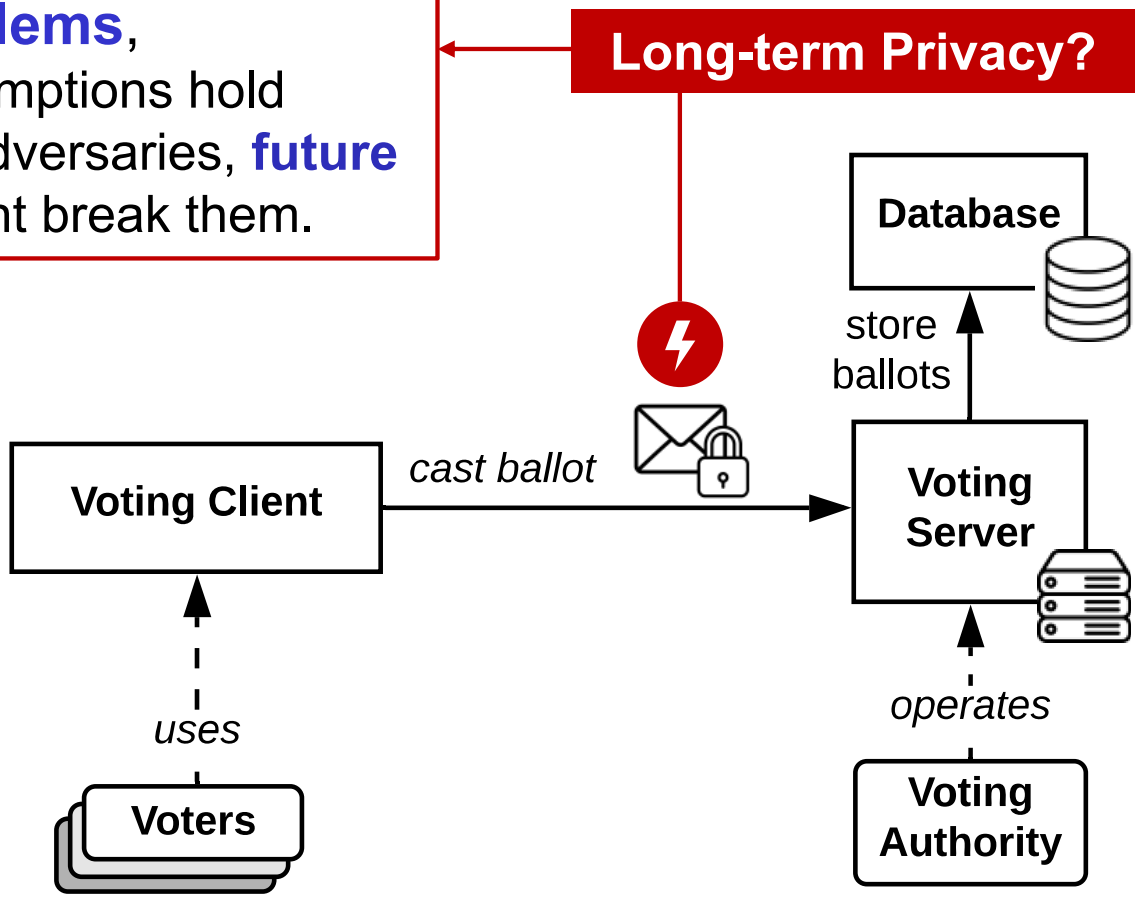
Receipt-Freeness (RF)

*A voter **cannot gain information** which can be used to prove, to a vote-buyer, how she voted*

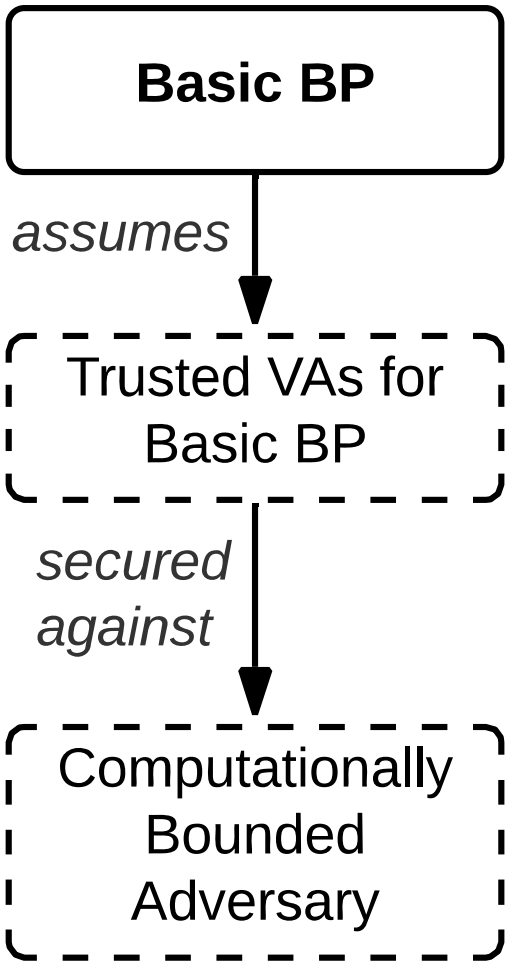
Long-Term Privacy

REV systems often rely on **cryptographic primitives which rely on hard problems**,

- While these assumptions hold against current adversaries, **future adversaries** might break them.

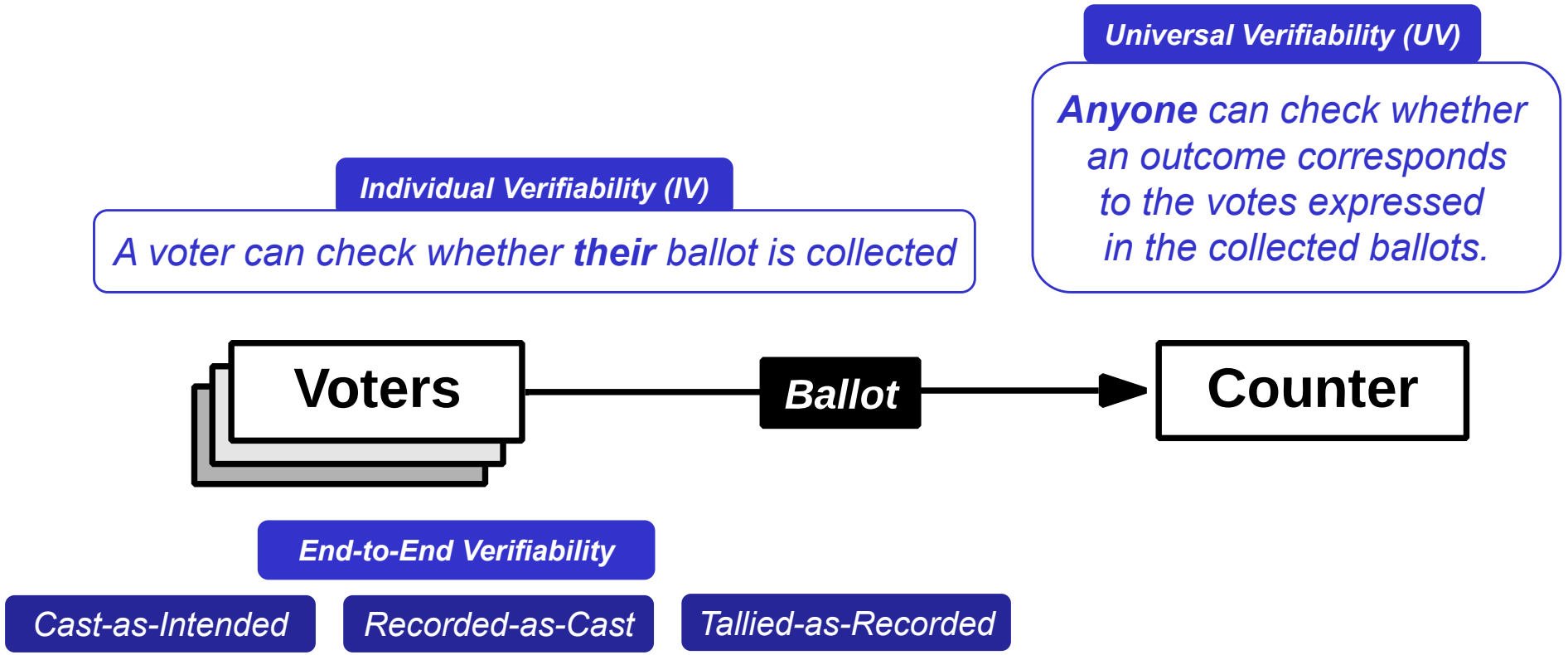


Basic to Unconditional Ballot Privacy

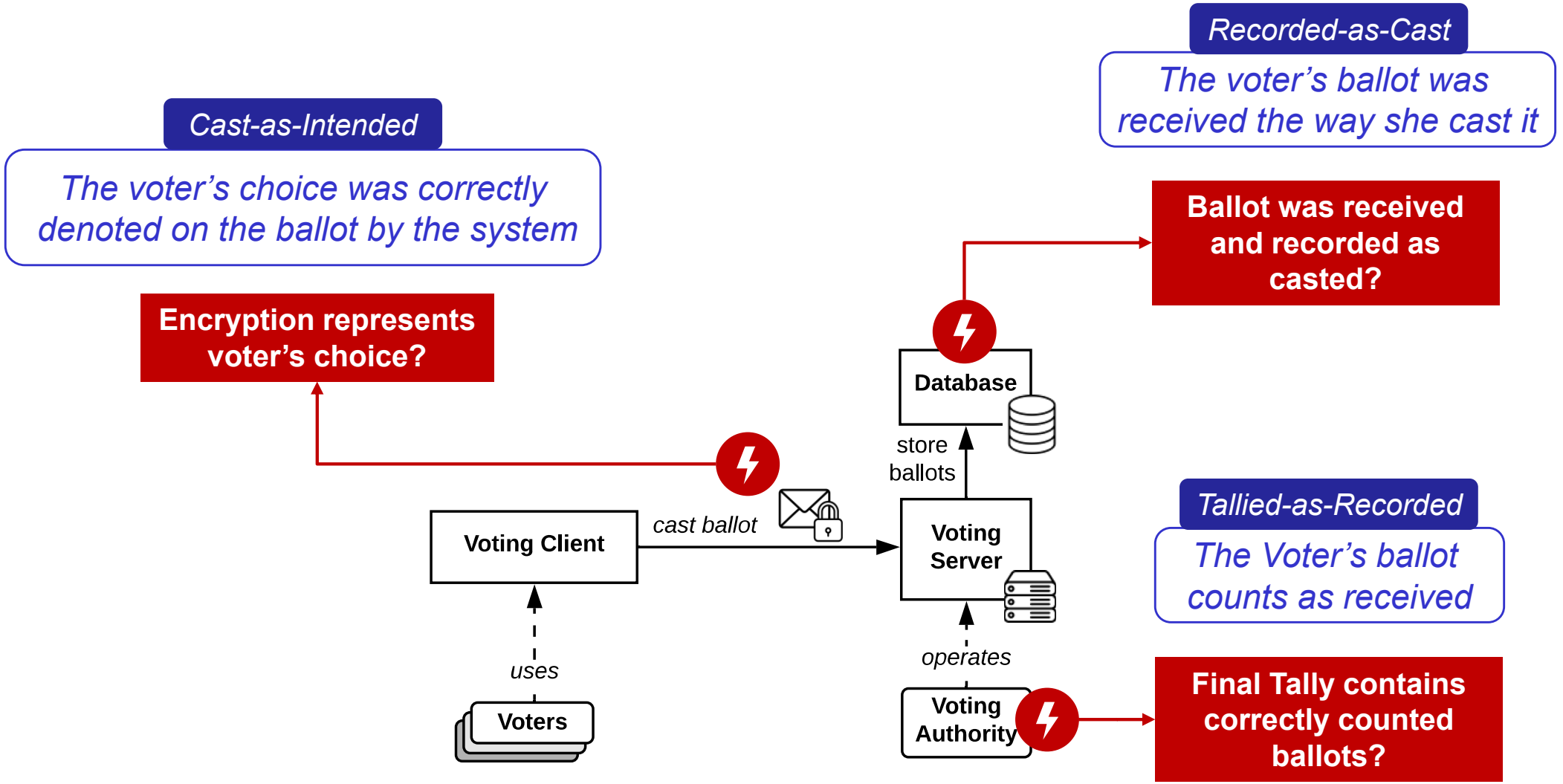


BP: Ballot Privacy

Verifiability Properties



End-to-End-Verifiability



Implications of Verifiability

- A verifiable REV system does not stop adversaries from attacking the integrity, but due to the verifiability mechanisms, it will be detected.
  - Removes trust assumptions in system operators

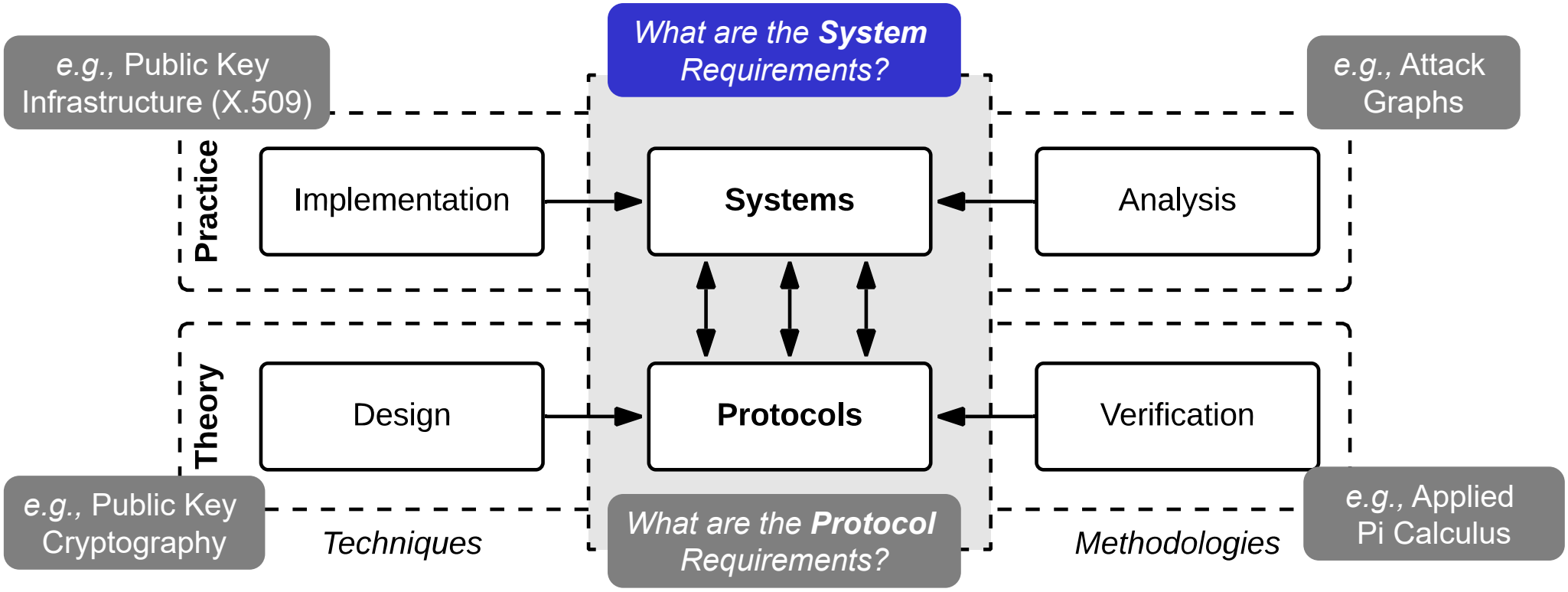
- Not every stakeholder (voter, candidate, …) is required to individually and universally verify
  - Since the adversary does not know who verifies his/her vote, the adversary cannot manipulate single votes without being detected with a very high probability
  - It is sufficient that one trustworthy entity verifies the tallying

Remote Electronic Voting (REV)

General REV System Requirements
& Building Blocks

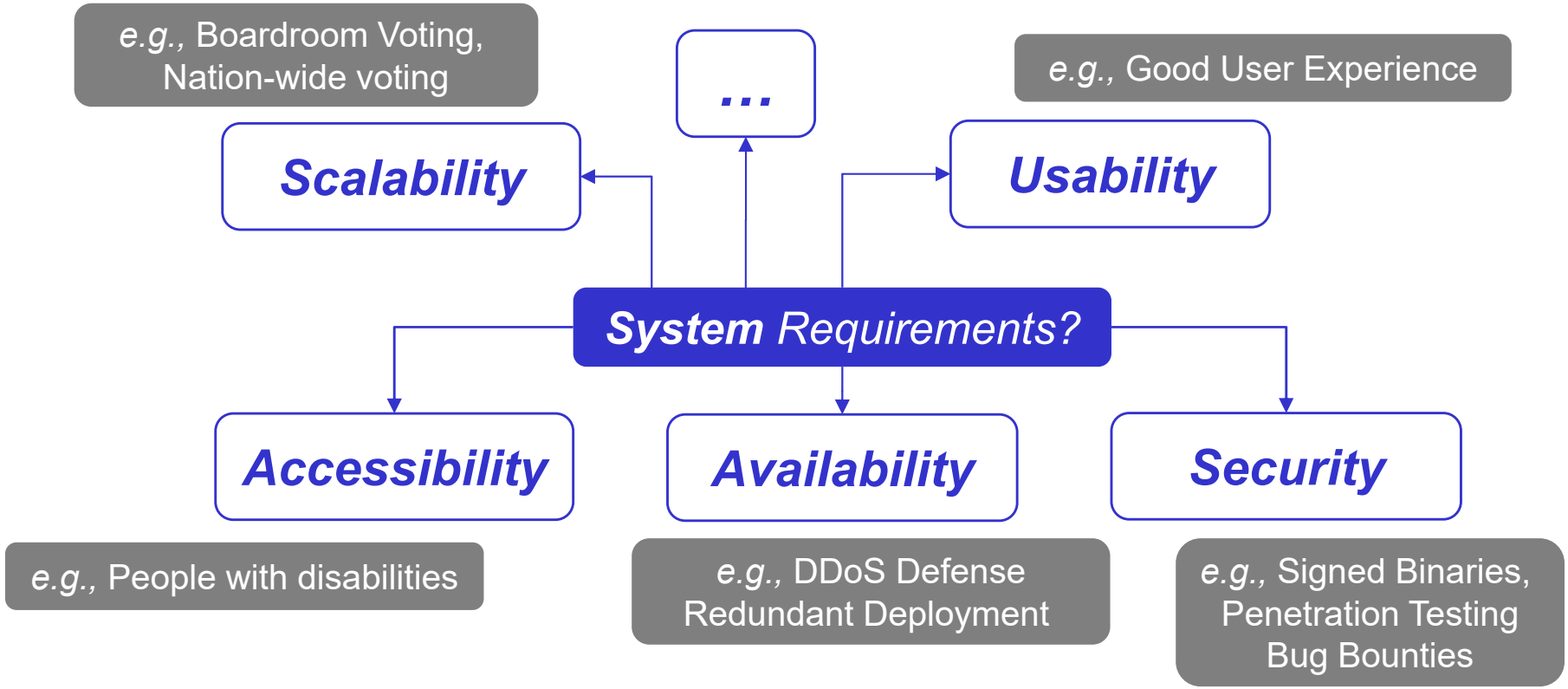
From Theory to Practice

- **From Protocols** (Theory) **to Systems** (Practice)



REV System Requirements

- What are the requirements for a REV System?



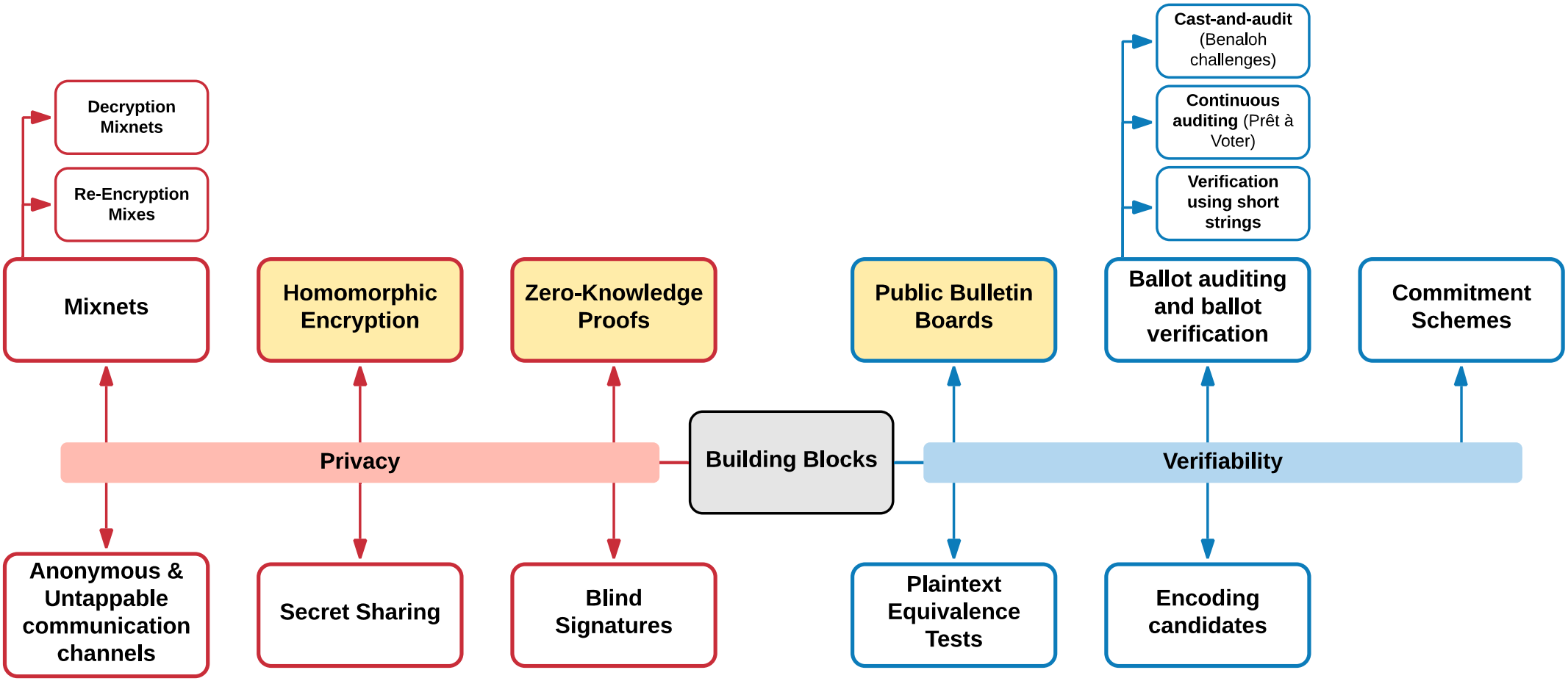
REV Building Blocks (1)

- From theory, we need design techniques, *i.e.,* voting protocols describing the process
  - *e.g.*, Secret Ballot Voting Scheme

- From practice, we require implementation techniques, i.e., system's engineering
  - *e.g.*, Public Bulletin Board *i.e.,* ballot box saving all results and logs of the procedures, making the process verifiable

- The combination of theory and practice results in a clearly defined set of building blocks

Building Blocks for REV Schemes



Secret Ballot Voting Schemes

scheme	privacy	verifiability	attacks	based on
FOO92 [32]	BP	UV	KR05 [44]	BS
CGS97 [25]	BP	UV	–	HE
BT94 [15]	RF	–	HS00 [37]	HE
NR94 [58]	RF	–	–	MC
SK95 [68]	RF	UV	MH96 [50]	MN
Oka96 [60]	RF	–	Oka97 [61]	BS
Oka97 [61]	RF	IV, UV	–	BS
HS00 [37]	RF	IV,UV	–	MN
LK00 [46]	RF	UV	LBD04 [45]	HE
BFP01 [8]	RF	UV	Hir01 [36],JCJ05 [42]	HE
MBC01 [48]	RF	–	LK02 [47], JCJ05 [42]	HE
LK02 [47]	RF	UV	–	HE
LBD03 [45]	RF	UV	–	MN
ALBD04 [4]	RF	IV, UV	–	MN
JCJ05 [42]	RF, CR	UV	–	MN
MN06 [52]	EBP*, RF	UV	–	CS
MN07 [54]	EBP*, RF	UV	–	CS, HE

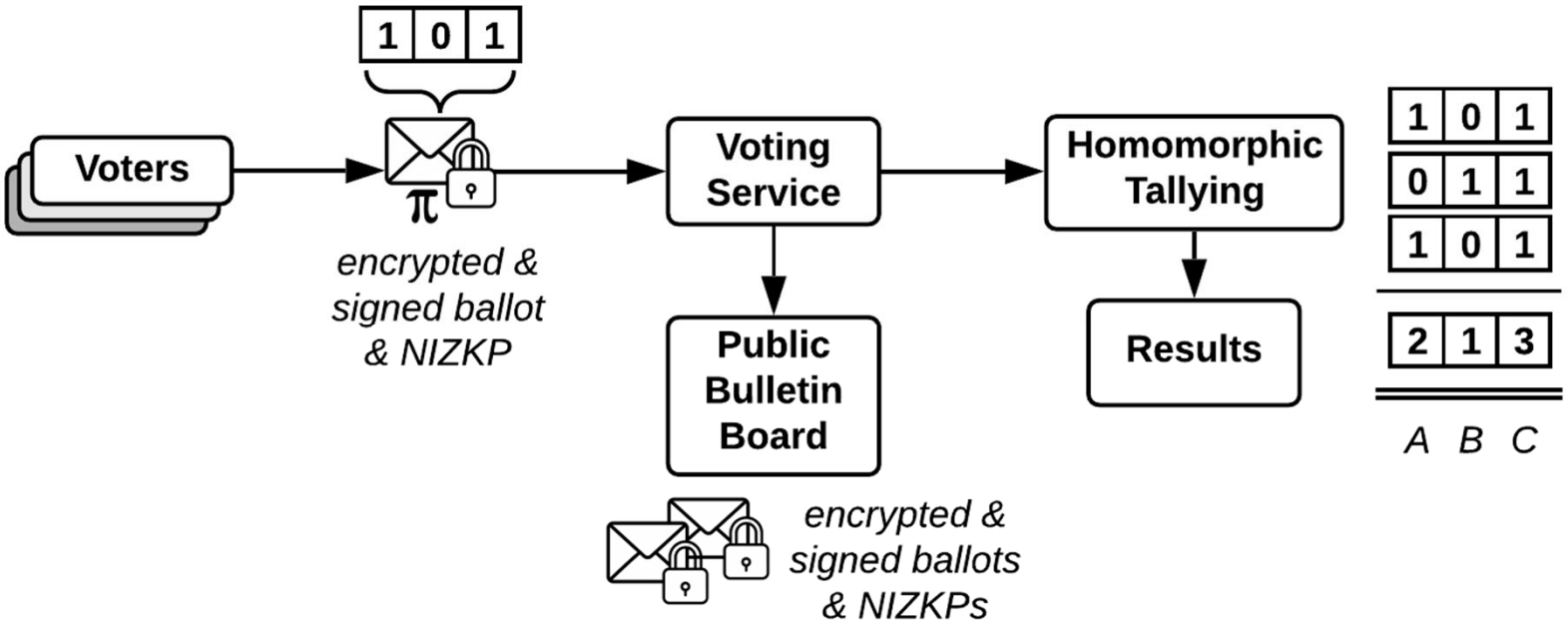
BP: Ballot Privacy
RF: Receipt-Freeness
CR: Coercion-Resistance
EBP: Everlasting Ballot-Privacy

BS: Blind Signatures
HE: Homomorphic Encryption
MC: Multiparty Computation
MN: Mixnet
CS: Commitment Scheme

Based on „Privacy and Verifiability in Voting Systems" by H. Jonker 2013

Homomorphic Tallying Approach

- Instead of decrypting each individual ballot, we use homomorphic tallying based on HE



NIZKP: Non-Interactive Zero-Knowledge Proof

Homomorphic Encryption in REV

- **In REV, HE used to tally votes**
- Every eligible voter V can encrypt his or her $vote$ using the election public key pk, resulting in the encrypted vote: $enc_{pk}\,(vote_V)$.
- No individual votes need to be decrypted, because the votes can be added together homomorphically.
- Thus, the final, encrypted tally is

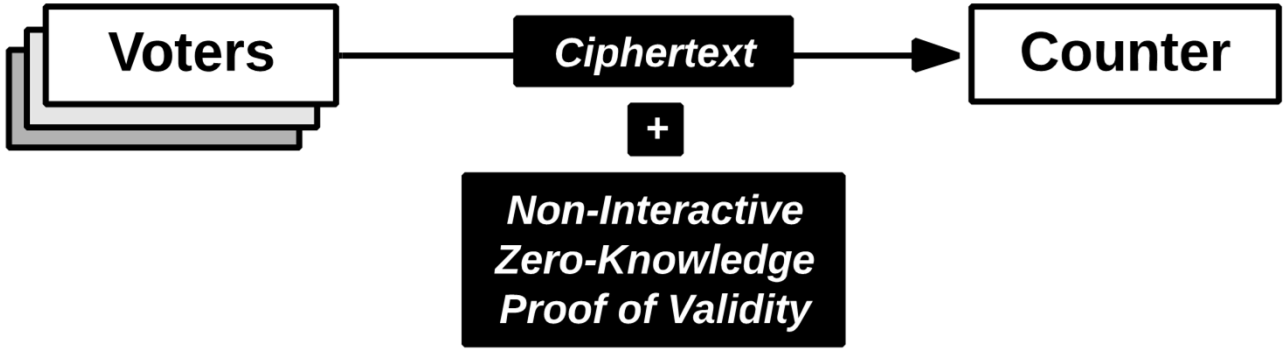
$$enc_{pk}\,(vote_1) \otimes \ldots \otimes\ enc_{pk}\,(vote_n) = enc_{pk}\,(vote_1 \oplus \ldots \oplus vote_n)$$

In turn, HE here leads to Ballot Privacy

Proving Ciphertext Validity

- But how can encryption malleability be tackled?
  - *I.e.,* what happens if an encryption of 100 instead of 1 happens?
  - There is the need for a path to assure the validity of the ciphertext, while assuring Ballot Privacy
- ***Zero Knowledge Proofs!***
  - Each voter proves that she cast a valid vote, without revealing her/his vote
  - The authorities prove that they know their secret keys (that match the election public keys), without revealing their secret keys
  - At the end of the election, the authorities prove that they have tallied correctly (decrypted the result correctly), again without revealing their secret keys

Achieving Privacy with HE and ZKPs



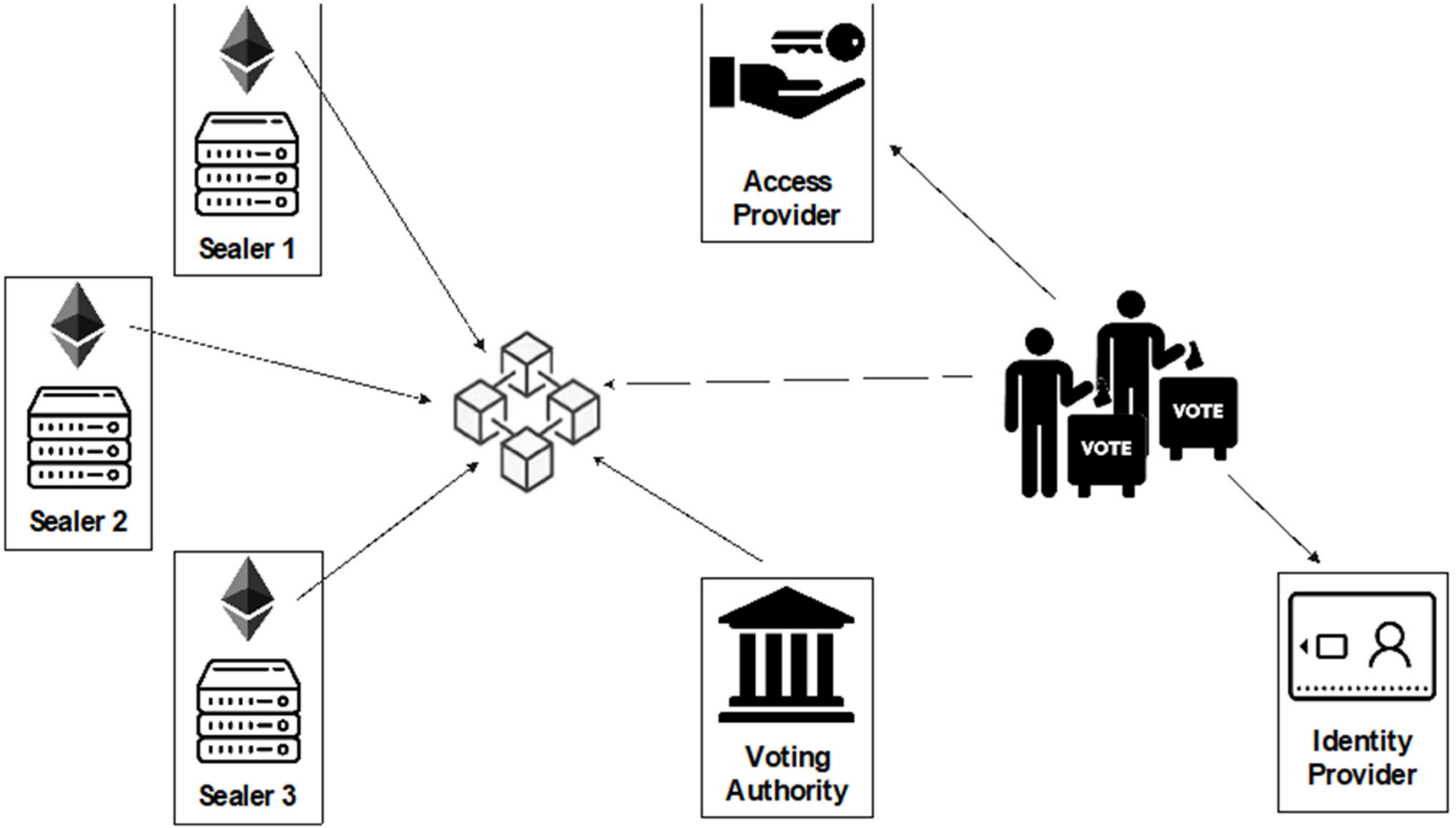
- Ballot Privacy achieved by encrypting with the public election key, and proving validity with NIZKPs
  - Thus, no individual ballot needs to be decrypted and authorities keep their secret election keys private, while homomorphically tallying the final result

- *How can voters verify the election data?*

BC-based REV

Provotum 2.0 Voting Protocol
and Evaluation

BC-based REV System: *Provotum*



Provotum 2.0 Identity Provisioning

REGISTRATION

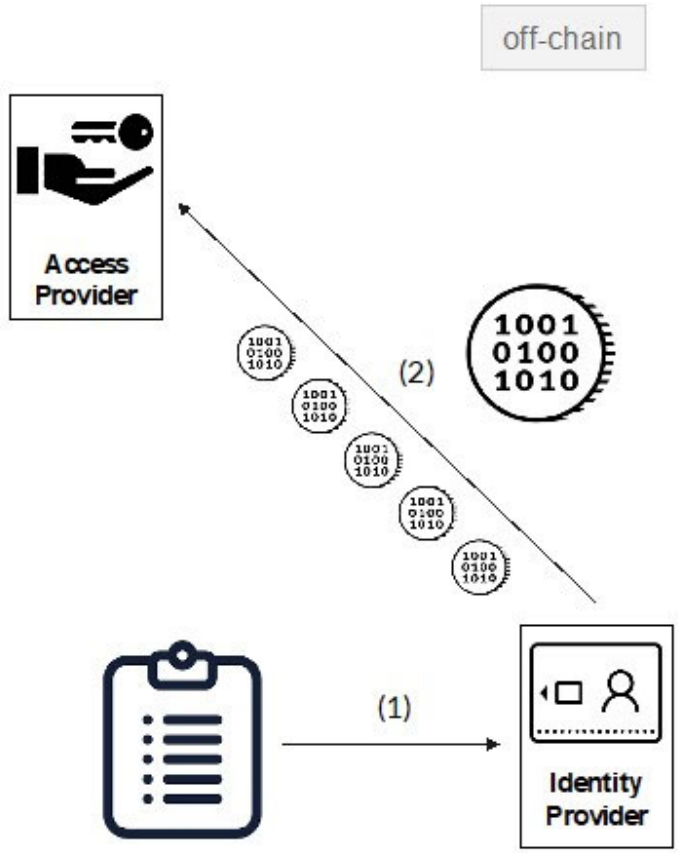
PAIRING

KEY GENERATION

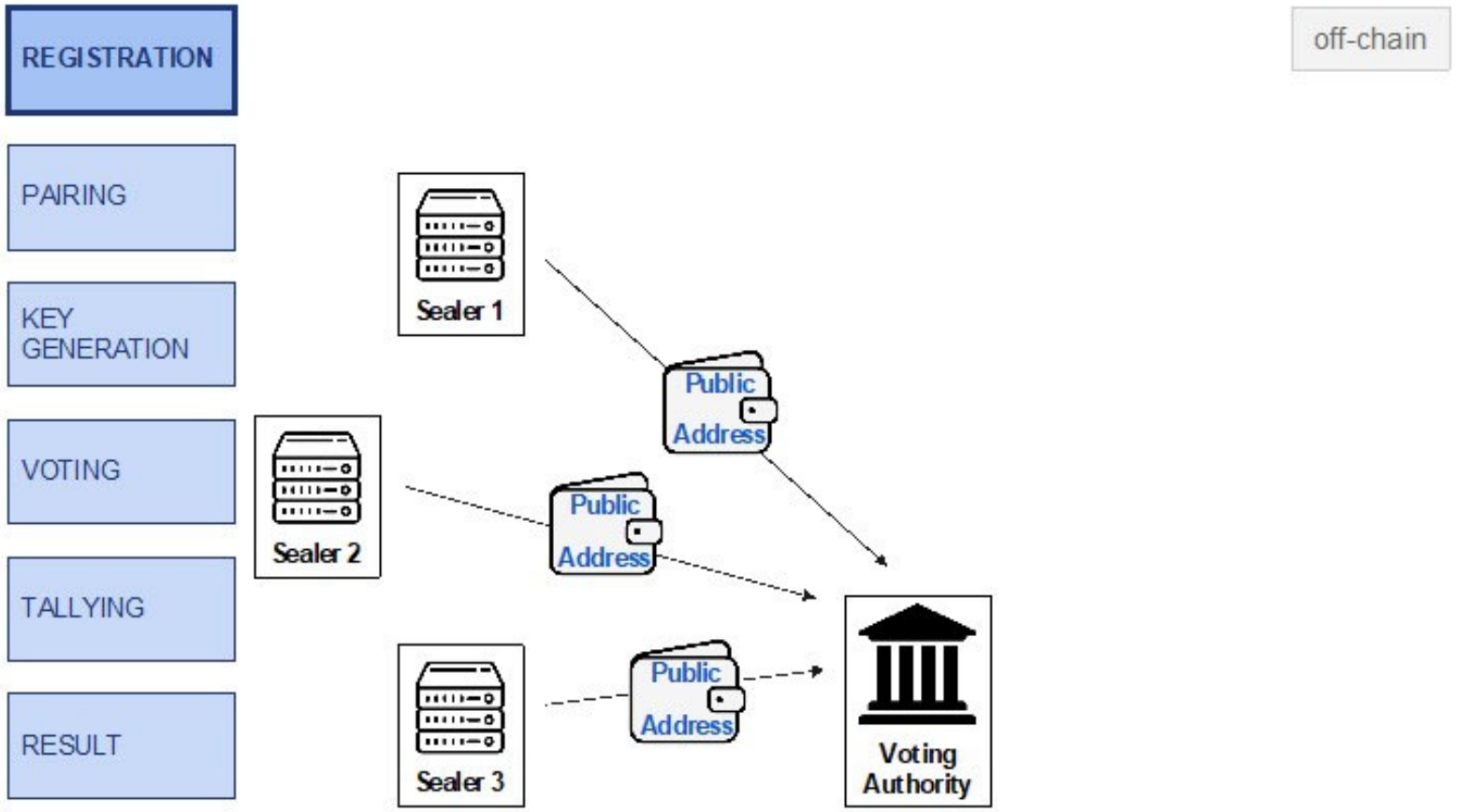
VOTING

TALLYING

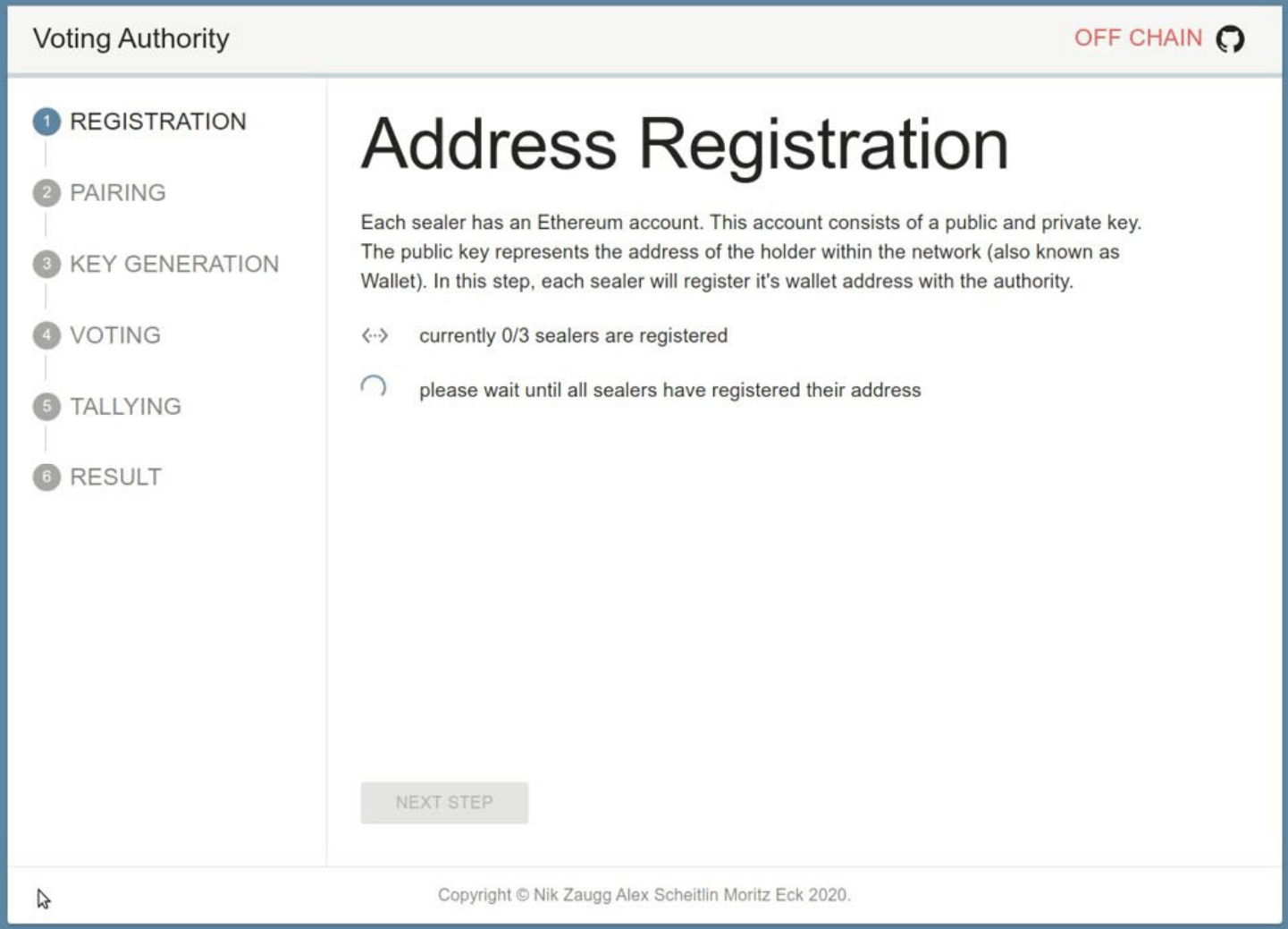
RESULT



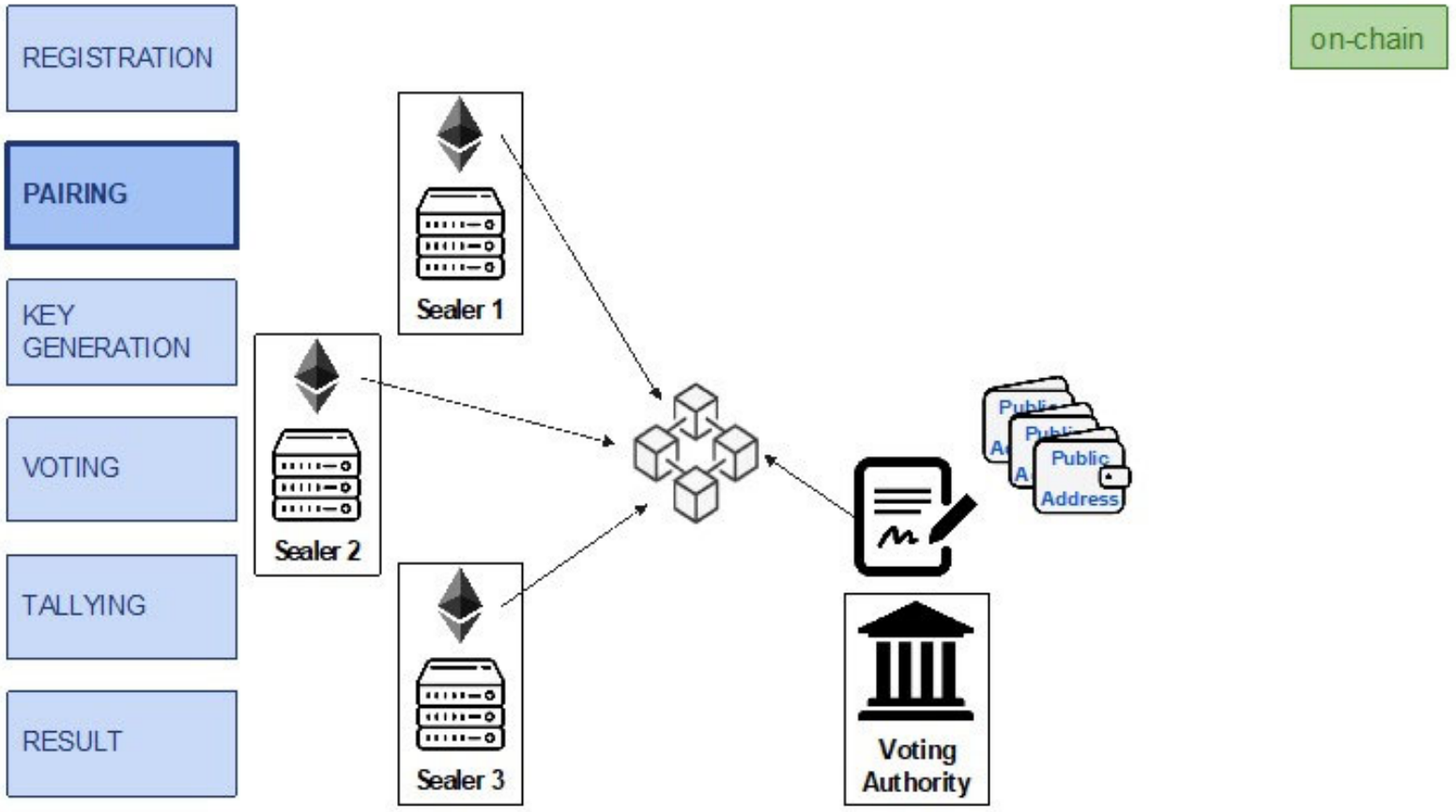
Voting Protocol: Registration (1)



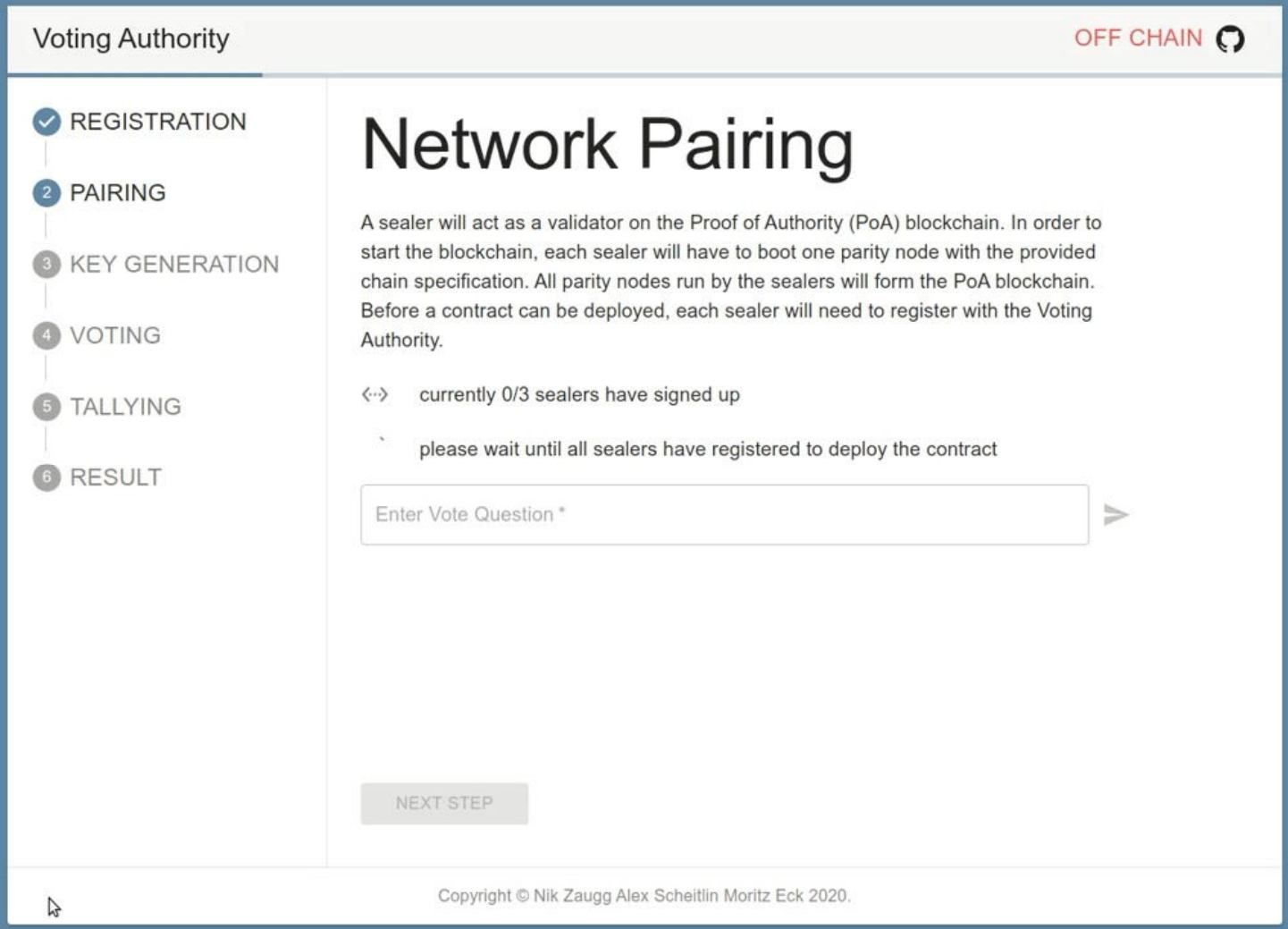
Voting Protocol: Registration (1)



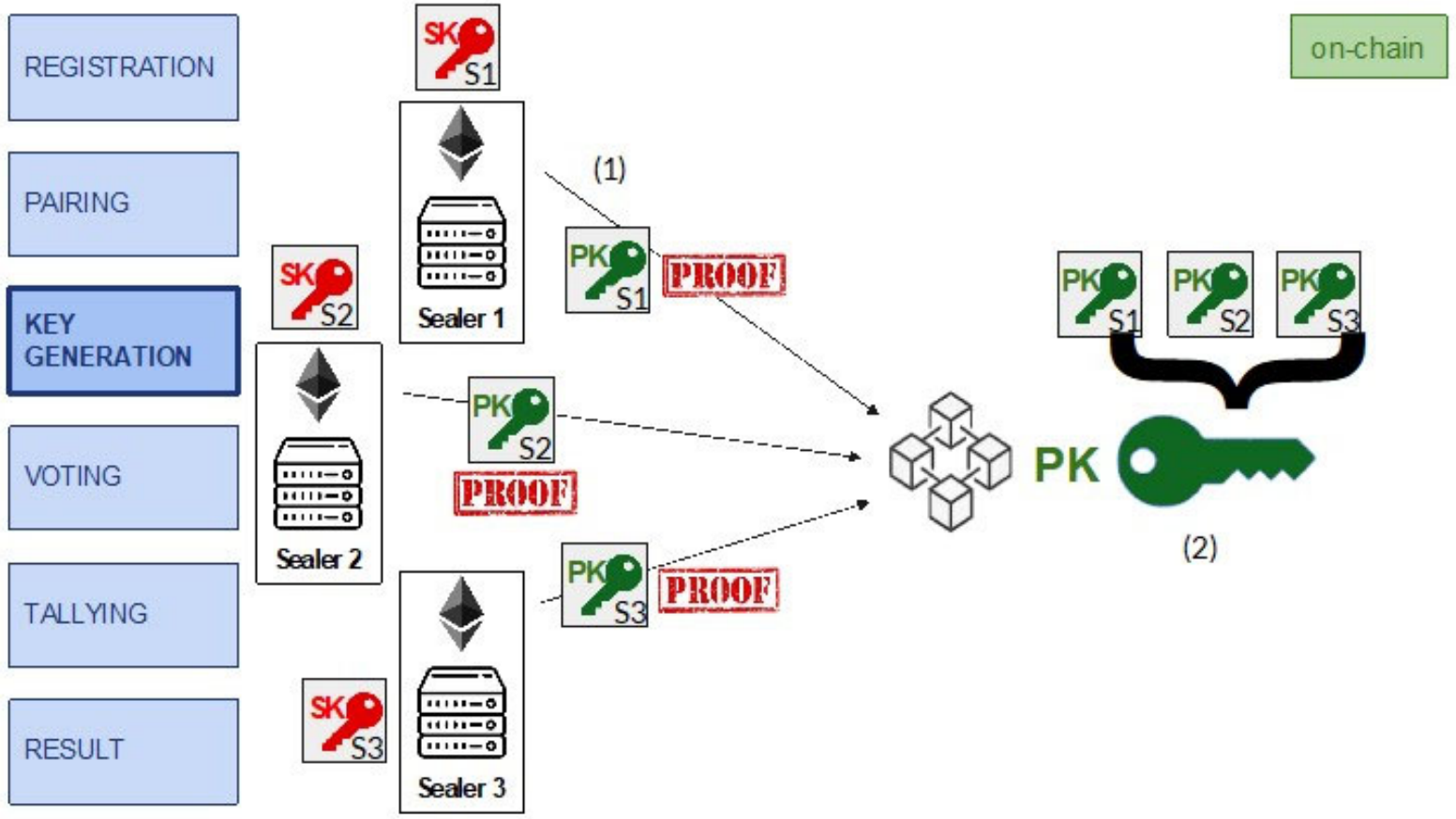
Voting Protocol: Pairing (2)



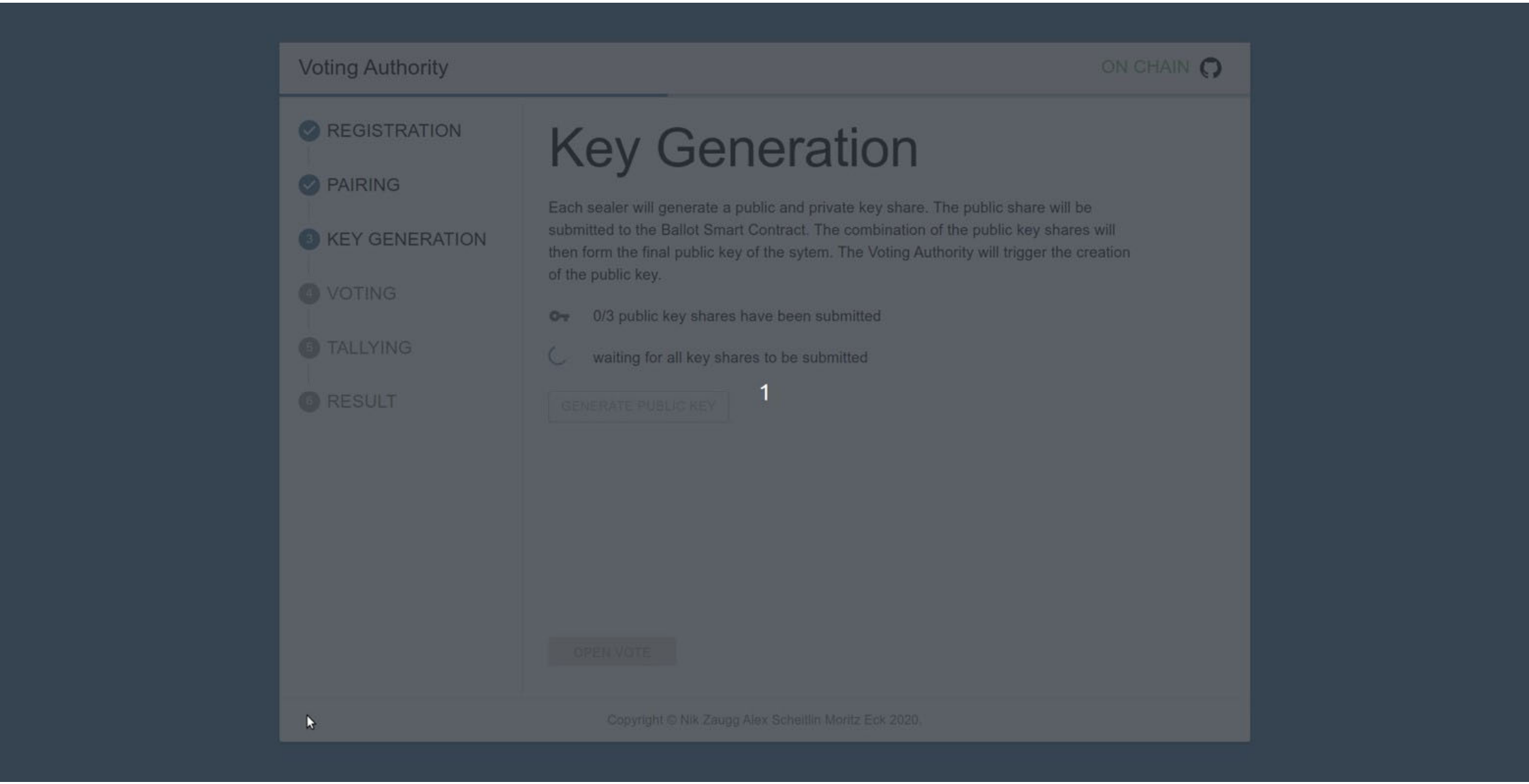
Voting Protocol: Pairing (2)



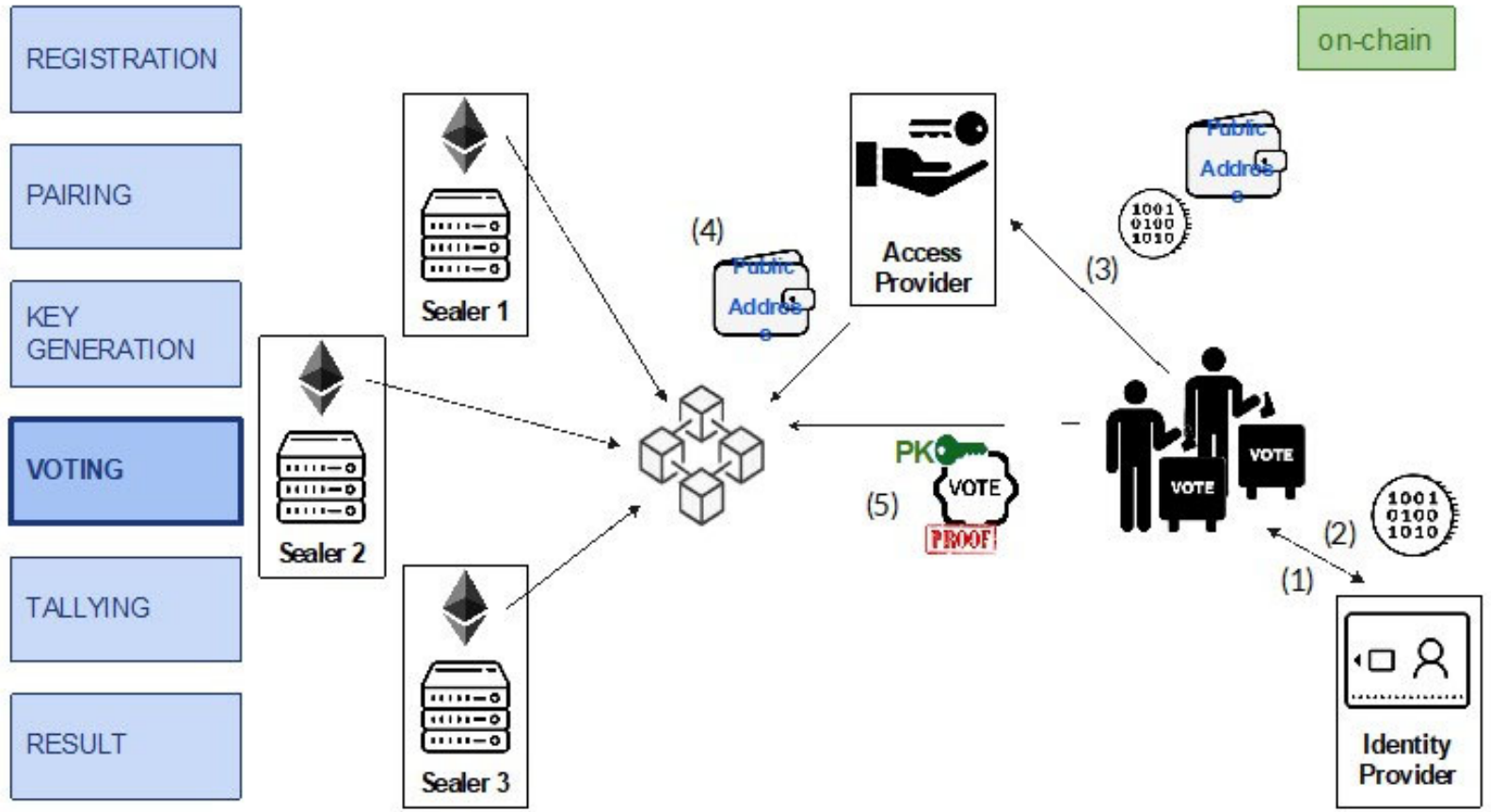
Voting Protocol: Key Generation (3)



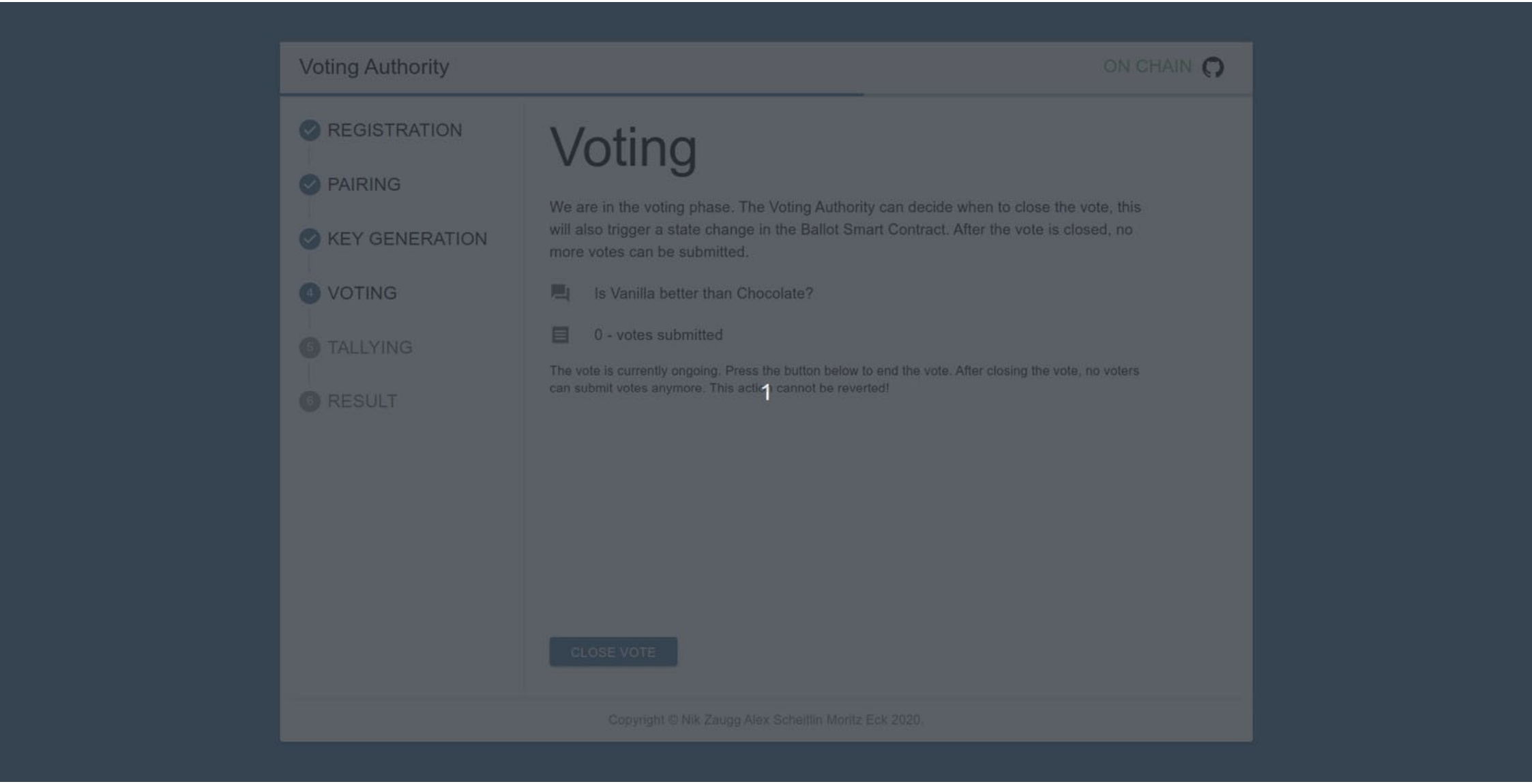
Voting Protocol: Key Generation (3)



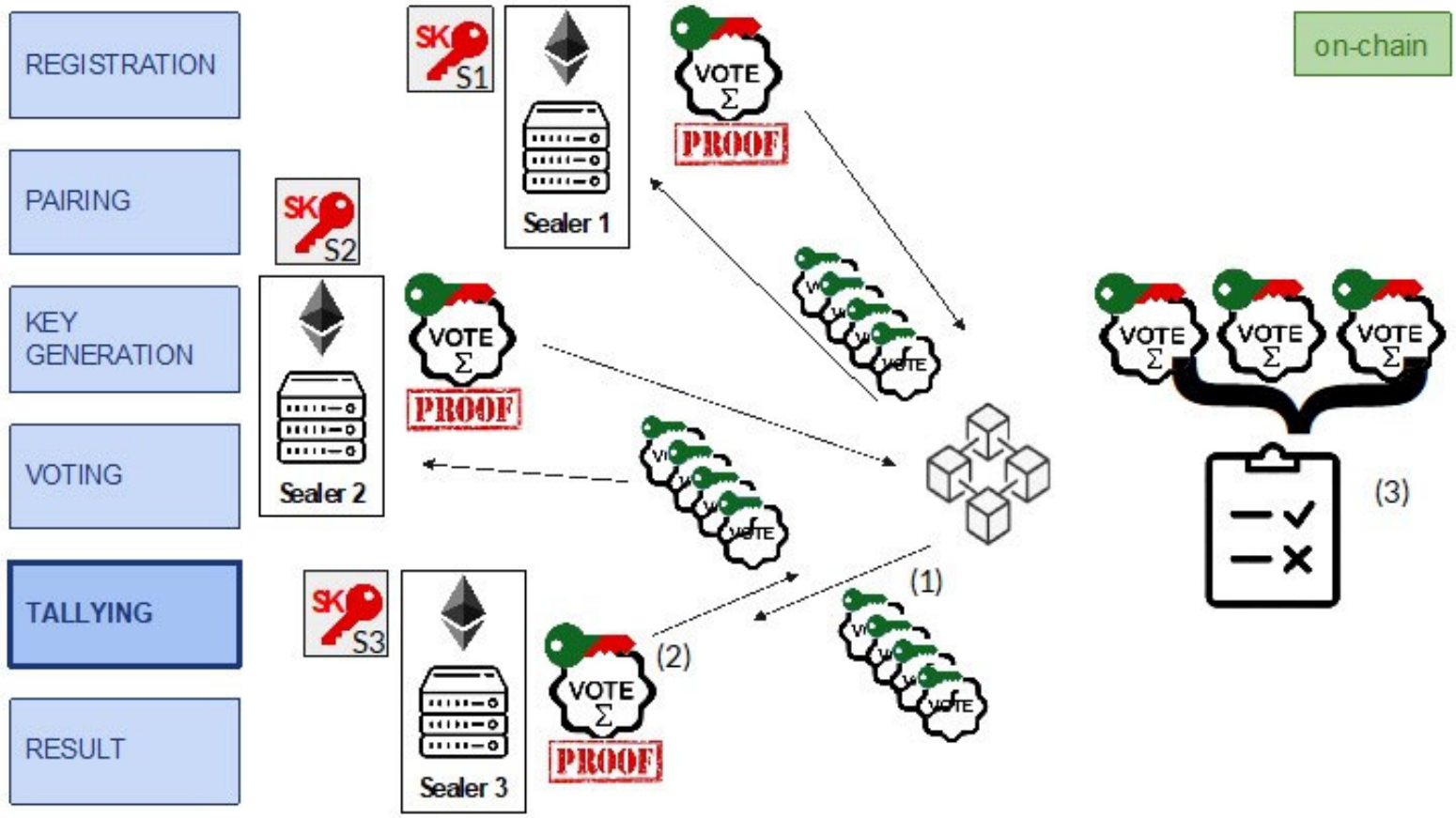
Voting Protocol: Voting (4)



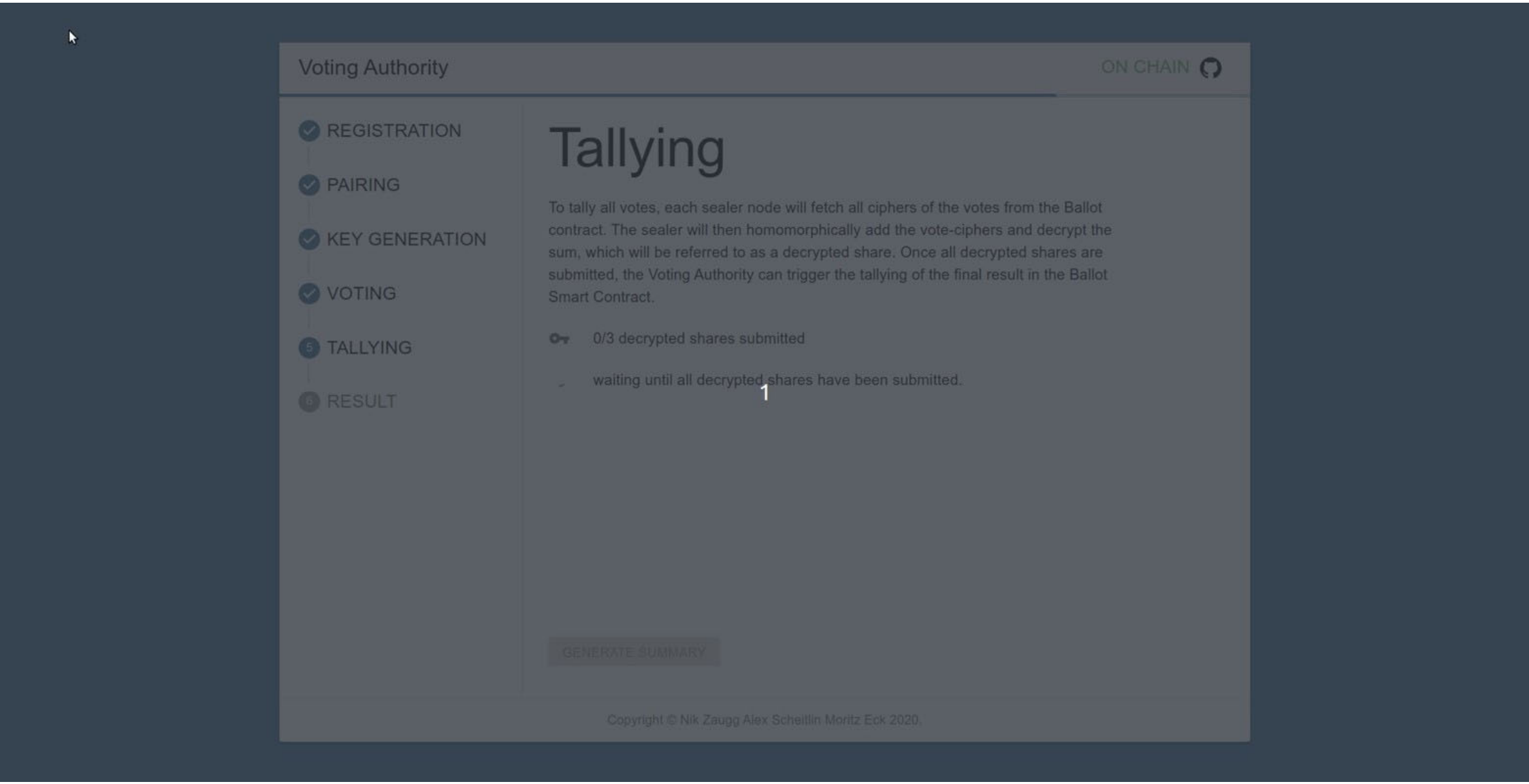
Voting Protocol: Voting (4)



Voting Protocol: Tallying (5)



Voting Protocol: Tallying (5)



Evaluations Provotum 2.0

- E2E-Verifiability and Ballot Secrecy achieved ✓
- Client-Side Cryptography ✓
- On-Chain Proof Verification ✓
- Distributed Key Generation ✓
- Cooperative Decryption ✓

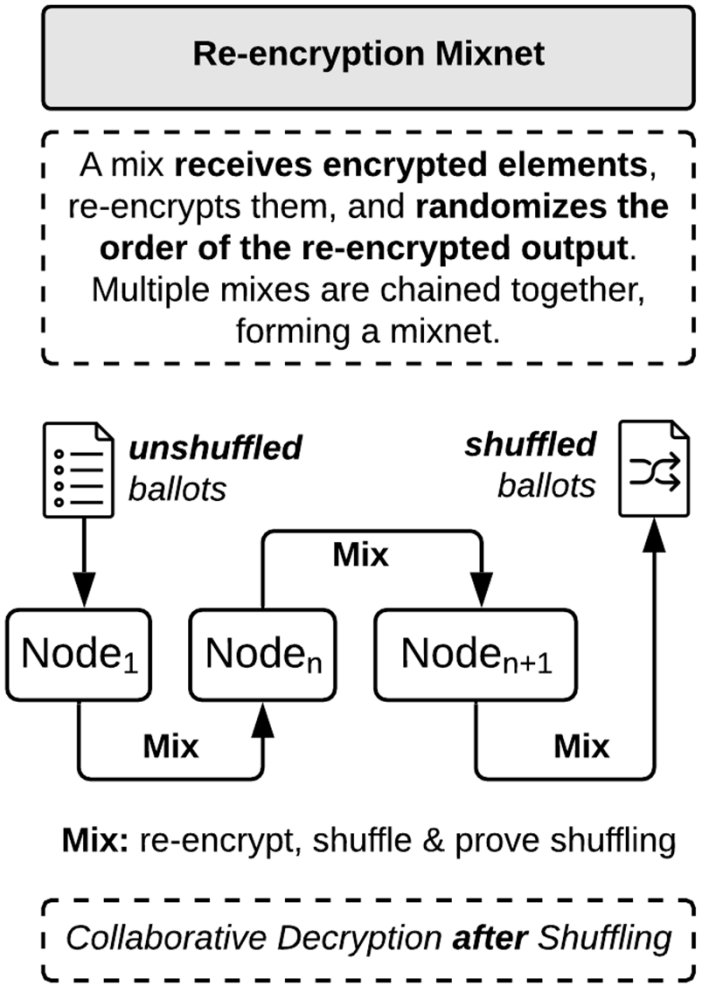
- Solidity EVM restricted to 256-bit integers ✗
- Mitigate Identity provisioning scheme threats ✗

- Receipt-Freeness and Coercion-Resistance ✗

From Homomorphic Encryption to Mixnets

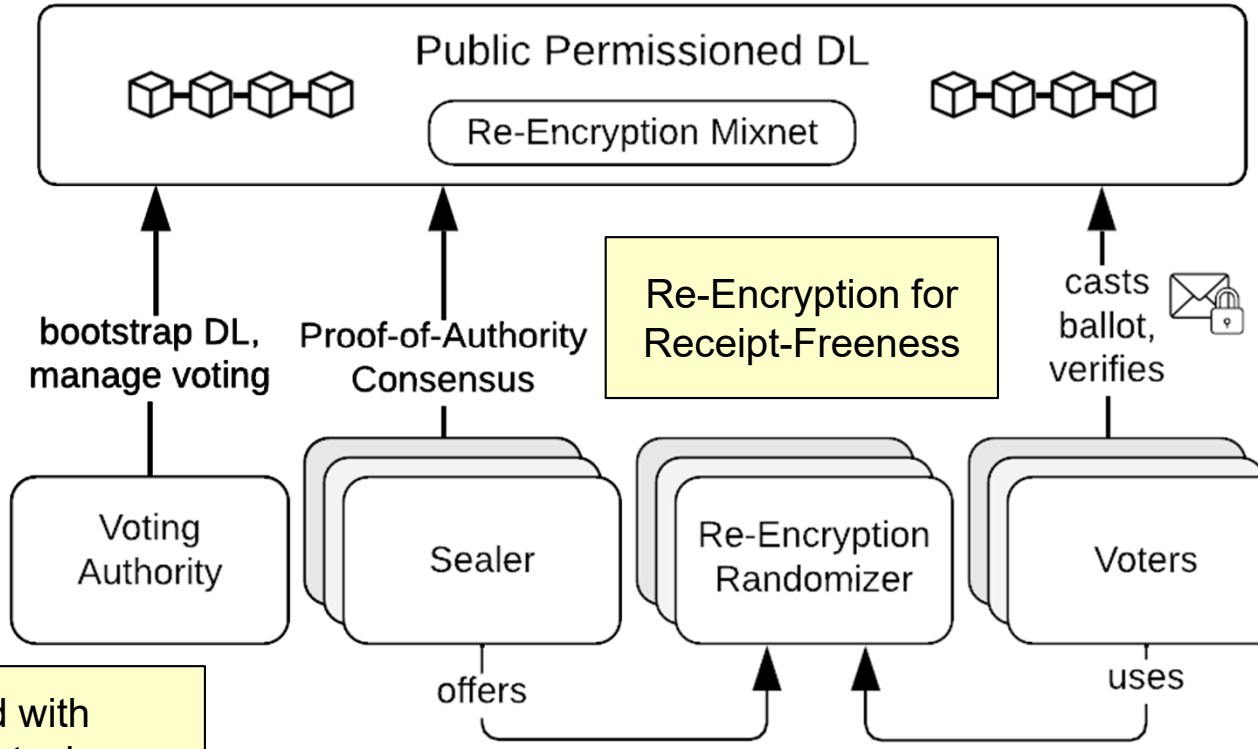
- **Homomorphic Encryption** has limitations in terms of Ballot Complexity and Encoding
  - Additionally, voters need to prove **validity** of the encrypted ballot

- With Mixnets, the encoding is flexible, because all ballots are decrypted in the end



ProvotuMN 3.0 Architecture

DL formed by the Sealers



VA entrusted with administrative tasks, such as boostrapping and managing voting phases

Trusted Sealers participating in Proof-of-Authority Consensus, Re-Encryption Mixnet, and Randomizer

Voters use Off-Chain Re-Encryption of ballot

Evolution toward ProvotuMN 3.0

- Substrate-based (Rust)
  - No Solidity or EVM limitations
  - Formulate Identity Provisioning as Assumption
- Homomorphic Encryption to Mixnet-based Scheme
  - Flexible ballot construction, less load on voter's ends
- Receipt-Freeness added
  - Re-Encryption Randomizer offered by the Sealers

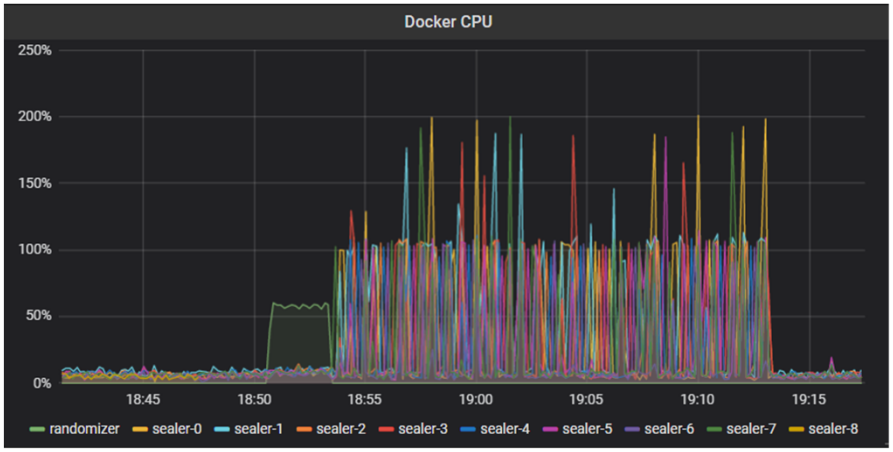


TABLE IX: Mixing Times [s] in the Distributed Deployment

Number of Votes	2 Sealers	5 Sealers	10 Sealers
100	99	91	111
1'000	1'280	1'580	1'750
10'000	10'002	11'535	13'405

Summary

- Remote Electronic Voting
  - If trust is distributed in REV, then REV can offer better privacy and verifiability than traditional voting channels, under certain trust and computational hardness assumptions
    - As soon as a system is public, future adversaries may be able to decrypt cipher-text

- Achieving such trust distribution can be implemented by using a Public Permissioned Blockchain, where authorities collaborate with a Proof-of-Authority consensus

Other Blockchains



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Other Blockchains (Selected Examples)
  - Overview
  - Era 1.0 – One Cryptocurrency
    - Bitcoin, Stellar, Monero, Libra
  - Era 2.0
    - NEO, Multichain, IOTA, HyperLedger, Ethereum, EOS, Cardano, Corda
    - "Era 1.0-4.0" Bazo
  - Era 3.0
    - Theta
  - Era 4.0
    - Polkadot, Aelf
- Summary

Community-based vs. corporation-based

BCs vs. DLs

Blockchains, Cryptocurrencies, Tokens

- App. 19,300+ different ones are listed at https://coinmarketcap.com/
- A classification wrt application domain had been presented already earlier
- A mapping on a set of selected BCs to these eras here is used to dive into selected details:
  - Done and to come

Era 1.0	Era 2.0	Era 3.0	Era 4.0
BinanceCoin	Algorand	Tetha	Aelf
Bitcoin	Bazo		Polkadot
Bitcoin Cash	Chainlink		
Bitcoin Gold	Corda		
Dasch	Elements		
Dogecoin	EOS		
Libra / Diem	Ethereum		
Litecoin	Hyperledger		
Monero	IOTA		
Paxos	MultiChain		
Ripple	NEO		
Stellar	OpenChain		
Tether	Tezos		
ZCash			

© 2022 UZH, CSG@IfI

Era 1.0

Era 1.0

Stellar (1)

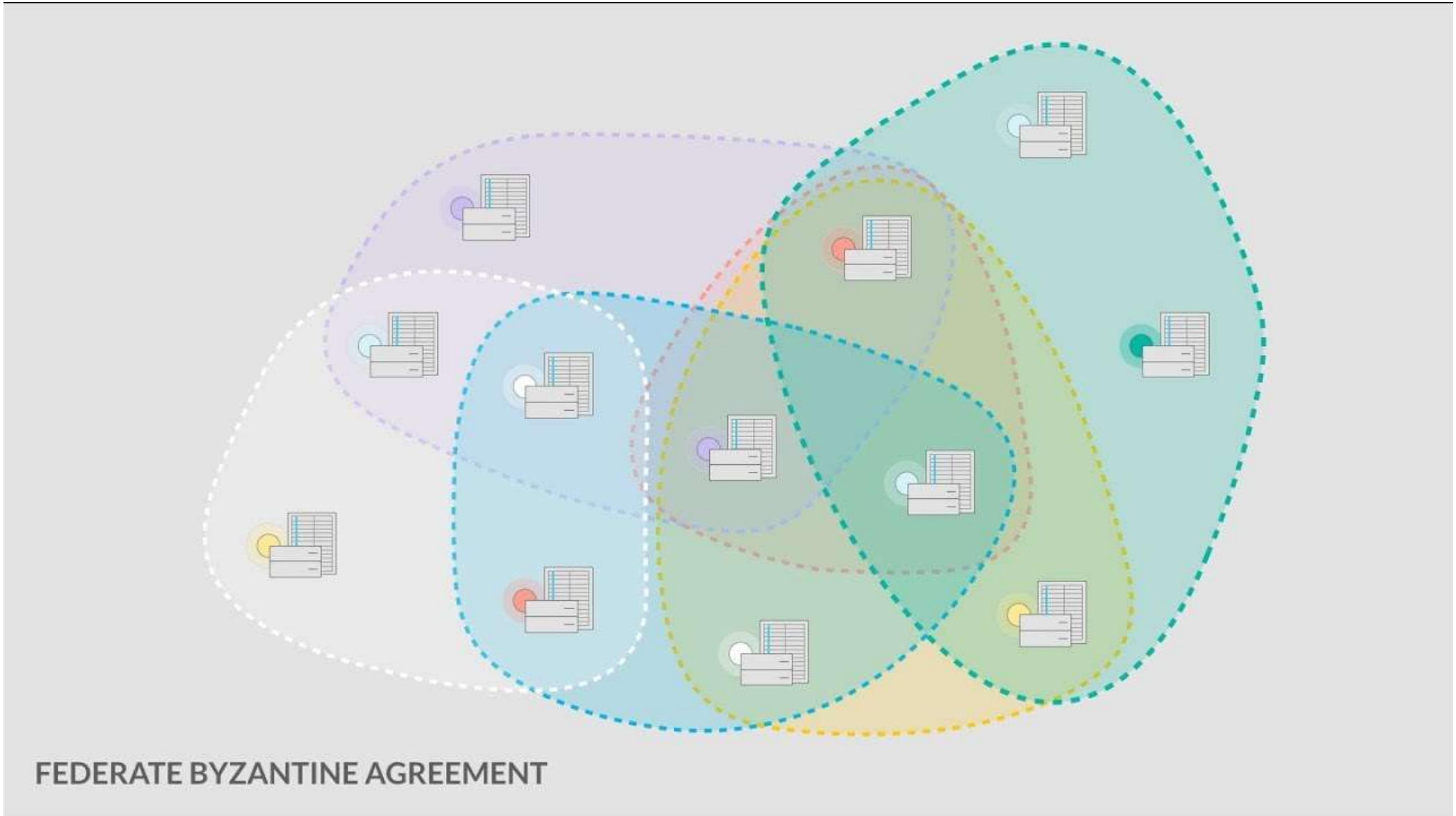
- Account-based
  - Public
- Market Cap. of 2.2 billion USD (March 2019)
- No Turing-complete SCs



- Stellar Consensus Protocol (SCP)
  - Each validator list trusted validators → *quorum slice*
  - Quorum slices overlap to form consensus
  - Block time of 5 s
  - Safety over liveness
    - Blockchain progress is halted until consensus is reached

https://pdfs.semanticscholar.org/3985/6a57fa0c6e7d646b7db88f48f17688693fe4.pdf

Stellar (2)



Era 1.0

Monero

- URL: https://getmonero.org
- BC type: Public



- Token type: Currency
- Account model: UTXO
- Consensus mechanism: Proof-of-Work (PoW)
  - Block size changes based on the transaction volume
- Additional mechanisms:
  - Pruning, ASIC-resistance PoW

© 2022 UZH, CSG@IfI

Monero Characteristics

- Major characteristics
  - Mining scheme used: Proof-of-Work (PoW)
  - Block size: Variable
    - Defined according the previous 100 blocks
    - Storage efficiency: 1,883,942 (block height)
      - ~ 72 GB (full node)
  - Transaction size: ~ 14 KByte
  - Transaction rate: ~ 7 tx/s (1,700 tx/s estimated)
  - Security features: ring signatures, ring confidential transactions (RingCT), and stealth addresses
    - All transactions on the network are private by mandate
    - Pseudo-anonymous

© 2022 UZH, CSG@IfI

Monero Economics

- Cryptocurrency: Yes, XMR
  - Exchange rates to fiat: Unstable
  - Transaction fees: Variable
    - Based on transaction priority, $0.021 USD (average)
- Wallet
  - Reference implementation in Javascript, https://mymonero.com
  - API: JSON RPC Interface
- Listed on crypto exchange(s)
  - https://localmonero.co, https://anycoindirect.eu
- Market capitalization (Aug 24, 2021, and May 9, 2022)
  - 5,804,308,881 USD and 3,890,429,848 USD

Monero Organization

- Developer community: Monero Team
  - Country of origin: N/A
  - SDKs: Daemon RPC, Wallet RPC
  - Blockchain explorer: https://moneroblocks.info
  - Full or light client: Both, https://getmonero.org/downloads
- Known vulnerabilities and attacks
  - Zero-amount miner TX, DDoS risk, Traceable transactions
  - Incident Response Team and community of developers work on reported bugs/vulnerabilities
- Major application domain(s)
  - Cryptocurrency with a focus on private and censorship-resistant transactions

© 2022 UZH, CSG@IfI

Monero Overall Evaluation

- Scalability
  - Large size of the blockchain, poorly scalable
- Centralization degree
  - 100%, fully decentralized
- Energy efficiency
  - 645.62 GWh of electricity (higher than a country like Haiti)
- Security level
  - 100% privacy-driven project

© 2022 UZH, CSG@IfI

Era 1.0

Originally Libra

- URL: https://libra.org
- BC type: Permissioned
  - Node types: Participant and Validators
- Token type: Currency
- Account model: Account-based
- Consensus mechanism: LibraBFT
  - Variation of the HotStuff protocol
    - Leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model
      - Works at the pace of actual (vs. maximum) network delay ⇨ Property “responsiveness”
  - Originally, about 100 founding members as Validators

© 2022 UZH, CSG@IfI

Original Libra Characteristics

- Major characteristics
  - Mining scheme used
    - Dedicated protocol based on trusted Validators
  - Block size: There is no concept of a block of transactions
  - Transaction size: 5 kByte
  - Transaction rate: 1,000 tx/s
  - Security features: Proof of authority, cryptographically authenticated database, memory-safe using Rust
  - Smart Contract
    - Language: “Move” Programming Language
    - Key details: Implemented using modules containing code values and resources containing data values.

Diem Economics

- Cryptocurrency: Yes, former “Libra Coins”, later “Diem”
  - Details: New coins are minted and excess is burnt
    - Value of the coin will be tightly bound to the global economy
  - Exchange rates to fiat: Stable (planned for)
  - Transaction fees: Low fees, variable according to gas price
- Wallet
  - Not available yet
- Listed on crypto exchange(s)
  - Not listed yet
- Market capitalization (w/o date)
  - The stablecoin “Diem” shall be released only in “(unknown)”!

© 2022 UZH, CSG@IfI

Libra/Diem Organization, now Silvergate

- Developer community: former Libra Association, then Diem Association, since January 31, 2022: Silvergate
  - Country of origin: Geneva, Switzerland
  - SDKs: https://developers.diem.com/main/docs/sdks
    - Java, Go, Python
  - Blockchain explorer: Yes (testnet)
  - Full or light client: Testnet only
    - https://developers.diem.com/main
- Known vulnerabilities and attacks
  - Double spending attack avoided by using the LibraBFT
- Major application domain(s)
  - Financial services, payments, e-commerce

Libra/Diem Overall Evaluation

- Scalability
  - Initial set-up at 1,000 tx/s for a set of 100 validator nodes
  - Plans to become permisionless as technology matures
- Centralization degree
  - 50%, partial decentralization
- Energy efficiency
  - Projected to use energy more like current data centers
- Security level
  - Designed to deal with most-common attacks (*e.g.*, double spending) and tolerate faults within the Validator network,
  - Not storing any personal data of users (plan)
- Future: unclear, due to (FinTech) regulator concerns

© 2022 UZH, CSG@IfI

Era 2.0

Era 2.0

NEO



- URL: https://neo.org
- BC type: Public
  - Node types: Participants, Validators, Speakers
- Token type: Currency, Utility
- Account model: UTXO
- Consensus mechanism: dBFT 2.0
  - Decision on consensus nodes for next round by voting
  - Users only need to wait for one confirmation (15 s)
- Additional mechanisms
  - Digital Identity based on KYC (Know Your Customer)
    - “Roll back the chain” with consensus

© 2022 UZH, CSG@IfI

NEO Characteristics

- Major characteristics
  - Mining scheme used: dBFT 2.0
  - Block size: ~ 5 KB (average)
    - Storage efficiency: 4,038,763 (block index)
  - Transaction size: ~ 200 Byte
  - Transaction rate: 1,000 tx/s (10,000 tx/s theoretically)
  - Security features: Anti-quantum cryptography mechanism (NeoQS), digital certificate
  - Smart Contract
    - Language: .Net, Java, C, C#, Go, Python, Kotilin, Javascript
    - Key details: Turing-complete, support concurrent operation, sharding, and unlimited scalability, when running in NeoVM
      - All operations have GAS costs, but, the first 10 GAS are for free

© 2022 UZH, CSG@IfI

NEO Economics

- Cryptocurrency: Yes
  - Details: 100 millions NEO created in the Genesis Block
  - Exchange rates to fiat: Unstable
  - Transaction fees: Variable based on operations (GAS)
- Wallet
  - Reference implementation in Javascript
    https://docs.neo.org/docs/en-us/node/gui/wallet.html
  - API: RPC, Smart Contract Framework
- Listed on crypto exchange(s)
  - *E.g.*, https://coinall.com, https://etoro.com
- Market capitalization (Aug 24, 2021, and May 9, 2022)
  - 4,067,320,276 USD and 1,088,966,776 USD

© 2022 UZH, CSG@IfI

NEO Organization

- Developer community: NEO Contributors
  - Country of origin: China
  - SDKs: NEO SDK (current version: 2.9.0)
  - Blockchain explorer: https://neotracker.io
  - Full or light client: both, https://neo.org/client
- Known vulnerabilities and attacks
  - NEO Vulnerability Bounty Program paid up to $10,000 USD per vulnerability reported
- Major application domains
  - Smart economy, digital assets, digital identity

NEO Overall Evaluation

- Scalability
  - Theoretically, NEO achieves infinite scalability with NEOVM
- Centralization degree
  - 50%, centralization of control
- Energy efficiency
  - Lower electricity costs and attendant carbon footprints
- Security level
  - Uses digital identity to link accounts to real life identity, traceable, assets protection
- Other dimensions of importance
  - Peer-to-peer networking, cross-chain interoperability

© 2022 UZH, CSG@IfI

Era 2.0

Multichain



- Open-source platform
- UTXO-based
- Private or public deployment possible
- Building Bitcoin-like blockchains
- PoW or PoA consensus
  - Customizable
    - Block time, *e.g.*, 15 s
    - Incentives, *i.e.*, block reward
    - Permissions to mine new blocks
    - Block size
    - Other parameters

© 2022 UZH, CSG@IfI

Era 2.0

IOTA

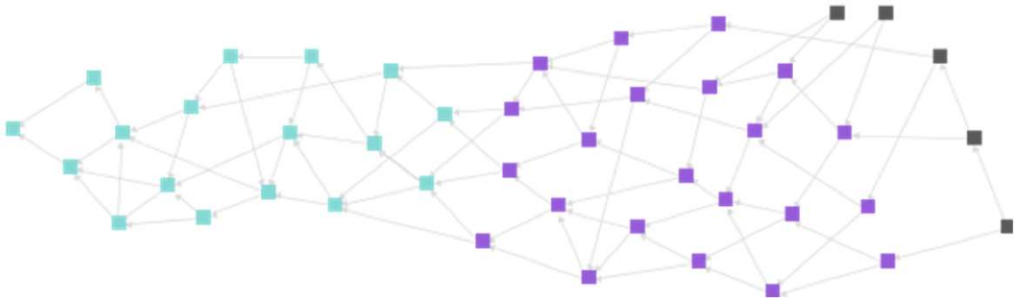
- Not a “real” blockchain
  - A public Distributed Ledger



- IOTA Tangle → Directed Acyclic Graph (DAG)
  - Each new transaction verifies two previous transactions
  - Single coordinator (IOTA Foundation) → Centralization
  - Block time of 60 s



Blockchain



Tangle (DAG/ Directed Acyclic Graph)

© 2022 UZH, CSG@IfI

Directed Acyclic Graph (DAG)

- All transactions linked to two previous transaction
- Regular snapshots (a certain centralization in place)
- More activity = more validation
  - But spam attacked happened in the past

- The more confidence in a transaction, the more secure
  - Run the tip selection algorithm 100 times
  - Count how many of those 100 tips approve our transaction, and call it A
  - The confirmation confidence of our transaction is “A percent”
  - Tip selection
    - Weighted random walk from the genesis towards the tips

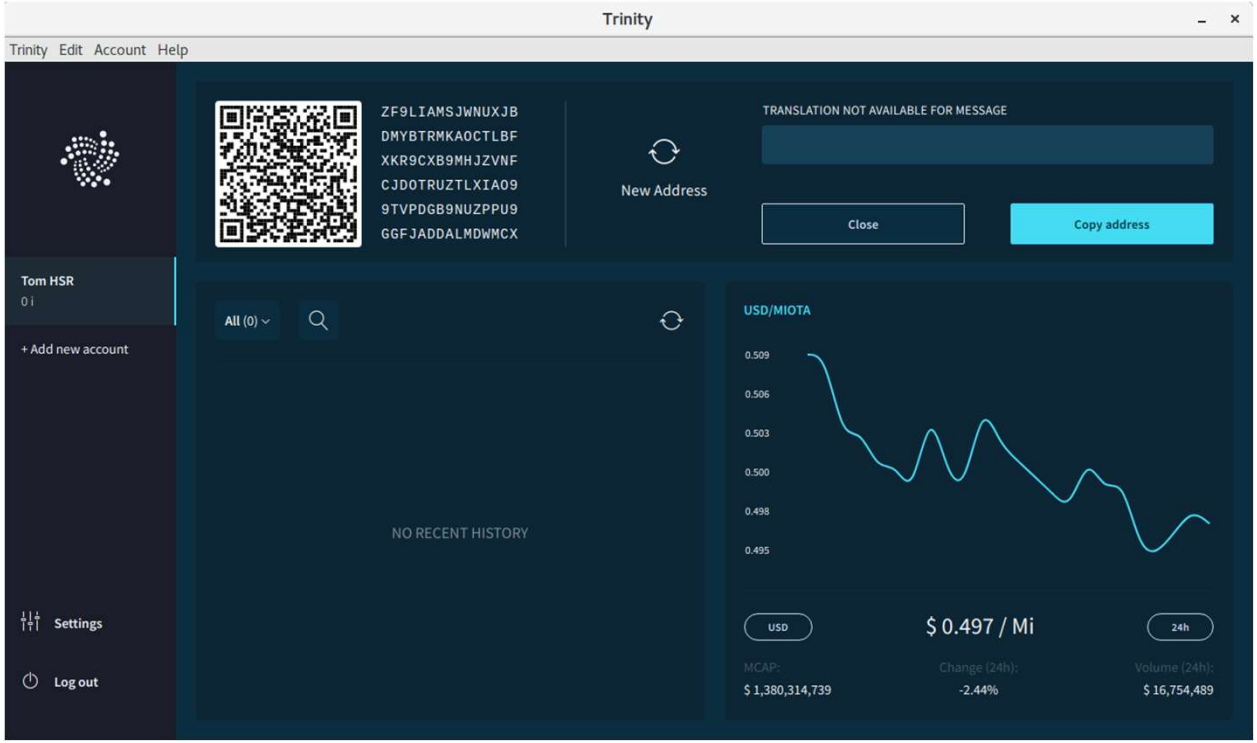
IOTA Characteristics

- Major characteristics
  - No support for SCs
  - Zero transaction fees for users
  - Feeless micro-transactions
  - Market Cap. of 1.26+ billion US$ (May 2022)
  - Data integrity for machines
    - A node can only issue a transaction after solving a cryptographic puzzle
  - Quantum-proof algorithms

  - “It is safe to receive any number of transactions to a given address until an outgoing transfer (a “send” transfer) is made. After that, this address should no longer be re-used!”

IOTA Wallet

- Wallet
  - Reference implementation in Java
    - https://github.com/iotaledger/iri
  - API in many languages: *E.g.*, C#, Javascript



https://trinity.iota.org/

Era 2.0

HyperLedger

- No currency
- (Private) Distributed ledger



HYPERLEDGER

- Umbrella Project
  - Fabric: Platform for building blockchain-based products
  - Composer: tools to build, test, and operate a blockchain
  - Cello: allows Blockchain-as-a-Service (BaaS)
  - Explorer: dashboard for blockchain monitoring
  - Burrow: Ethereum Smart Contracts execution
  - Sawtooth: modular enterprise-level blockchain
  - Caliper: blockchain benchmarking tool

Active Hyperledger Projects

- Open source effort created to advance cross-industry Distributed Ledger Technologies (DLT) hosted by “The Linux Foundation”
  - https://www.hyperledger.org
- Active HL projects include
  - HL Fabric a general-purpose, permissioned DL
  - HL Indy a special-purpose, permissioned DL
    - Focusing on decentralized identity management
  - HL Iroha is a permissioned DL
    - Tailored for digital asset management
  - HL Sawtooth a general-purpose, permissioned or permissionless DL

© 2022 UZH, CSG@IfI

Hyperledger Fabric

- Hyperledger Fabric (HLF) initialized by IBM
- HLF implemented in Golang
  - Smart contracts in Go, JavaScript, or Java
  - SDKs exist in Node.js, Java, Go, REST and Python
- Network roles and components of HLF
  - Channels for complete data isolation b/w a set of participants
  - Private data collections enable transactions between participants on the same ledger, but keep data private to a subset of transactors
    - Private data is shared peer-to-peer, with only hashes stored on the shared ledger as
  - Identity Mixer for anonymity of transaction submitters

Hyperledger Indy

- Hyperledger Indy focused on identity management
  - DIDs (Decentralized Identifiers) without requiring central resolution authorities.



    - Verifiable claims are an interoperable format for the exchange of digital identity attributes and relationships
      - Currently in the W3C standardization pipeline
    - Zero Knowledge Proofs enable that some or all of the data in a set of claims is true without leaking any additional information, including the identity of the prover
- HL Indy is independent, but commonly associated with "The Sovrin Foundation", which offers a public utility for identity, built on top of Indy's codebase

Hyperledger Iroha

- Hyperledger Iroha is written in C++, initially developed by a group of Japanese developers, who built their own BC for mobile use cases



- BFT Consensus algorithm called YetAnotherConsensus (YAC)
  - Iroha offers ready-to-use set of commands and queries, as well as multi-signature transactions
- Digital asset management capabilities
  - *E.g.,* create digital assets, register accounts, and transfer assets between accounts.
  - Robust permission system, allowing permissions to be set for all commands, queries, and joining of the network

Hyperledger Sawtooth

- Hyperledger Sawtooth initialized by Intel
- Permissioned and permissionless deployment possible



- Proof-of-Elapsed-Time (PoET) consensus algorithm
  - Dependent on Intel's Software Guard Extensions (SGX) to elect leaders to cut blocks based on random wait times.
  - Peers have access to all transaction data, unlike in Fabric, where "private" channels can be established
- Smart contract capabilities
  - SDK for Python, Javascript, Go, and Rust
  - Ethereum contract compatibility is provided with Seth, enabling the deployment of Sawtooth contracts to Ethereum

Era 2.0

EOS



- URL: https://eos.io/
- BC type: dApp Ecosystem
  - Node types: block producers, backup block producers, api nodes, seed nodes, validators
- Token type: Utility
- Account model: Account (name in a human readable format)
- Consensus mechanism: BFT-dPoS
  - 21 allowed block producers → centralization
  - EOS holders vote for block producers, EOS tokens for votes
- Additional mechanisms:
  - Free of charge for users
  - Contracts mutable, freezing accounts possible

EOS Coin

- Major characteristics
  - Mining scheme used: BFT-dPoS
  - Block size: limited by network capacity
    - Storage efficiency: 4 TB after 8 months of operation
  - Transaction size: N/A, max 30 ms for execution
  - Transaction rate: max 3,996 tx/s
  - Security features: SHA 256 is used to generate the digest and ECDSA to sign the block
  - Smart Contract
    - Supported binaries Web Assembly (WASM/ABI files), Turing-complete
    - Supported Languages: C++
    - Other languages that produce WASM code: RUST, Python, Solidity

EOS Economics

- Cryptocurrency: No
  - Utility token to acquire EOS CPU/RAM to run dApps
  - Exchange rates to fiat: floating, “rather” stable
  - Transaction fees: free of charge for users, accounts with dApps need to stake EOS tokens against EOS CPU and trade EOS tokens against EOS RAM
- Wallet
  - Reference implementation C++, Cleos, Nodeos, Keosd https://github.com/EOSIO/eos and API in C++
- Listed on crypto exchange(s)
    - Bitfinex, Binance, Kraken, HitBTC, Liqui, Gate.io, EtherDelta, Kucoin, Mercatox, Exrates, Livecoin, IDEX, COSS, TIDEX
- Market capitalization (cf. summary slide at end of slide deck)

EOS Organization

- Developer community: 188 developers (GitHub)
  - Country of origin: Cayman Islands
    - Deployment needs: depending on the role, *i.e.*, block producers, backup block producers, api nodes, seed nodes
  - SDK: Jeos (Java), Eospy (Python)
  - Blockchain explorer: Yes, https://bloks.io
  - Full or light client: Full https://github.com/EOSIO/eos
- Known vulnerabilities and attacks:
  - Against dApps: Numerical Overflows, Authorization Checks, Apply Checks, Transfer Error Prompt, Random Number Practice, Rollback Attack
- Major application domain(s)
  - SC-based distributed Applications (dApps)

EOS Overall Evaluation

- Scalability
  - High: both horizontal and vertical scaling
- Centralization degree
  - Centralized among block producers
- Energy efficiency
  - High
- Security level
  - Pseudo anonymous, smart contract mutability, account freezing, account code change

Era 2.0

Cardano

- URL: https://cardano.org/
- BC type: public blockchain
  - Node types: core, edge, relay nodes
- Token type: Currency (ADA)
  - Maximum supply at 45,000,000,000 ADA
- Account model: Extended UTXO (EUTXO, multi-assets)
  - Assets represented as balances within users' accounts
  - Balances stored as a global state of accounts, kept by each node, and updated with every transaction
- Consensus mechanism: Ouroboros PoS (BFT)
  - Family: runs permissionless and permissioned blockchains

© 2022 UZH, CSG@IfI

Cardano Characteristics (1)

- Additional mechanisms
  - Cardano's SC language allows to run end-to-end tests w/o leaving integrated development environment or deploying the code
- Major characteristics
  - Funded through ICO in 2015
  - Claim: "Bitcoin is too slow and inflexible, Ethereum is not safe or scalable … Cardano *solves* these concerns."
  - Mining scheme used: different PoS (BFT) schemes
  - Block size:
    - Normal blocks vs. Epoch Boundary Blocks (EBB) at 2 MB
    - Epochs are divided into 21,600 "slot" units
    - During every slot, at most one block might be appended to the BC

Cardano Characteristics (2)

- Transaction size: N/A, 20 s PoS Block Time (slot time)
  - Tx fees are utilized
- Transaction rate:
  - Until 2020 app. on ave. 2,500 tx/s, since 2021 on ave. 35,000 tx/s
- Security features:

https://medium.com/@undersearcher/how-secure-is-cardano-5f1e076be968
https://medium.com/unraveling-the-ouroboros/introduction-to-ouroboros-1c2324912193

  - Randomness seed for the next epoch generated via a cryptographic scheme Publicly Verifiable Secret Sharing (PVSS)
    - Once a block is created, nodes play a coin flipping game to generate a random number and use PVSS to encrypt outcomes onto the BC
  - Replaced by cryptographic function Verifiable Random Function (VRF)
- Smart Contract
  - "Where programming Ethereum requires coding in two languages, Solidity for the on-chain code and Javascript for the off-chain parts, and other systems suffer a similar split, Plutus is the only system that provides an integrated language for both, based on Haskell."

Era 2.0

CORDA

- Open-source enterprise blockchain platform
  - Enables legal contracts and other shared data to be managed and synchronized between mutually untrusting organizations
  - Private distributed ledger
- Consensus services
  - Notary pools
  - Optimized for different purposes on the same network
- Corda network governing body
  - Establishes and maintains a list of notary pools
  - Operated to “sufficient” standards of integrity
  - Centralization

© 2022 UZH, CSG@IfI

Corda Similarities with Bitcoin BC

- Immutable states
  - Created by transactions
- Transactions have multiple inputs and outputs
  - Bitcoin: Ledger defined as the unspent transaction output set (UTXO set) as a result
  - Corda: Generalization of this concept to support arbitrarily complex data models
- A contract is pure function
  - Contracts do not offer storage or the ability of interaction
  - Given the same transaction, a contract's “verify” function always yields exactly the same result

Era 1.0-4.0

Bazo

- Bazo is a permissionless blockchain for research
  - Its not an ICO, not looking for funding, but for ideas
  - Testing of new ideas, new mechanisms, new algorithms
- Blockchain from scratch at CSG and now HSR
  - Started with simple PoW, a solid, well-tested mechanism
  - Gradually extended with new ideas and concepts
  - Now with unique feature proposals/ideas
    - Self-contained proofs
    - Sharding
    - Pruning
    - Smart Contracts with only big-ints
  - https://github.com/bazo-blockchain



© 2022 UZH, CSG@IfI

Bazo Functionality (as of March 2022)

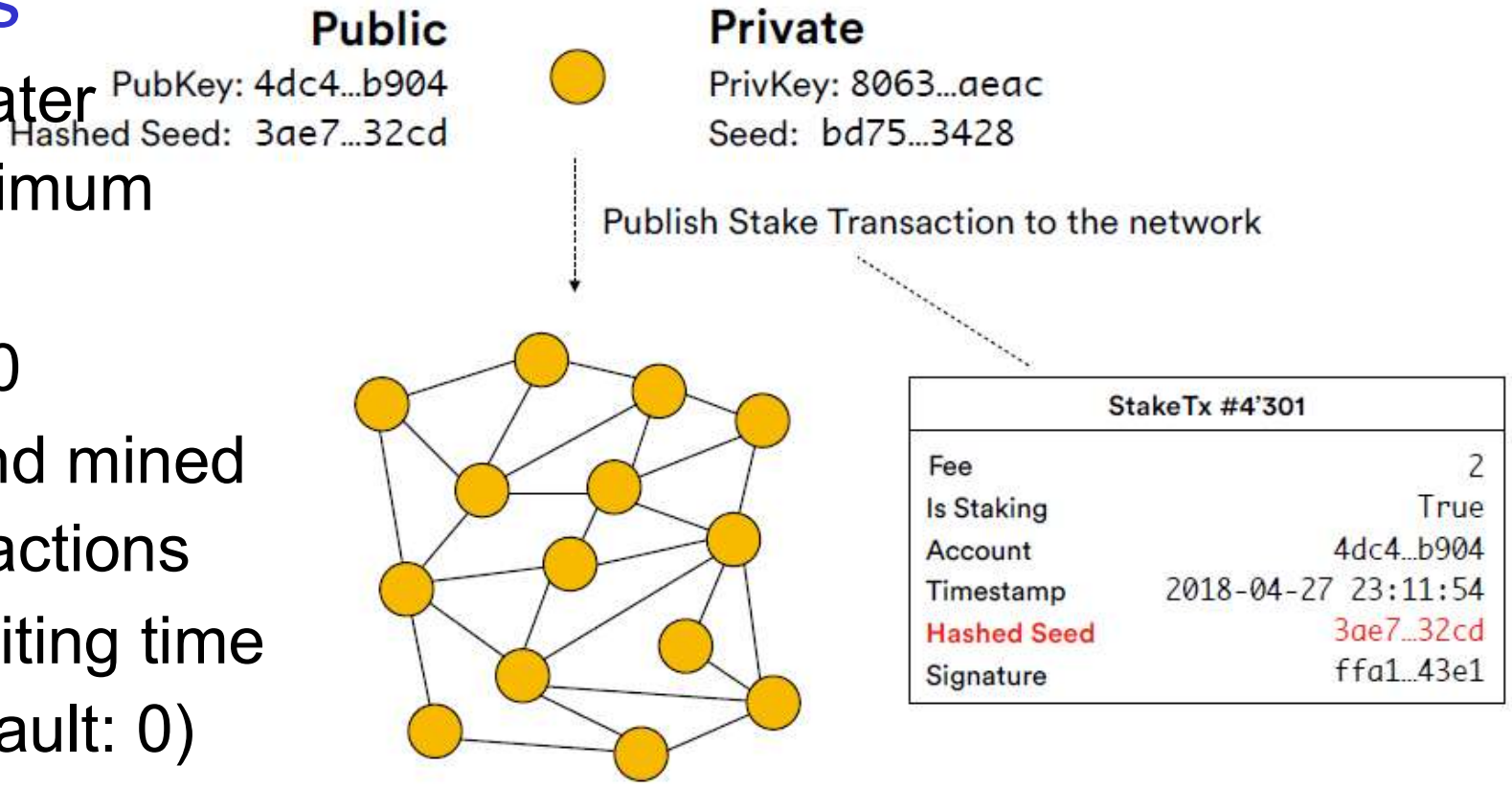
- PoW consensus mechanism initially
- PoS consensus mechanism later + cryptographic proof
- Progressive Web App (PWA)-based Mobile Wallet for Bazo
- Blockchain Explorer for Bazo
- Mobile Light Client for Bazo
- Pruning in Bazo and self-contained proofs
- Bazo Smart Contracts
- Sharding (in progress)

- Application
  - Oysy (Aduno) BC-based loyalty system

Credits go to T. Bocek (now HSR), S. Rafati, B. Stiller, and MA/BA students with CSG: L. Sgier, J. van der Assen, L. Boillat, S. Bachmann, M.-A. Chételat, A. Kürsat and HSR students R. Blum, S. Fontana, E. Meier, M. Steiner

Bazo's PoS

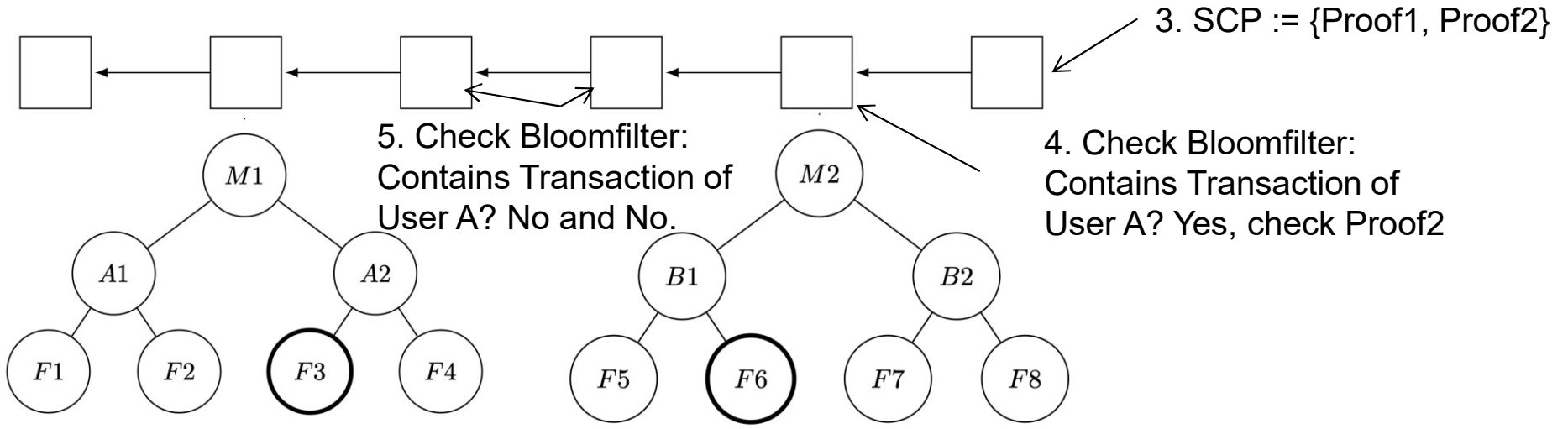
- Miner ("**validator**") **of a block chosen depending on his** wealth, also defined as "stake"
  - Validators restricted to 1 hash per second (H/s)
- Requirements
  - Balance greater or equal minimum staking fee
  - Default: 1000
  - Published and mined Stake Transactions
  - Minimum waiting time elapsed (default: 0)



© 2022 UZH, CSG@IfI

Bazo's Self-contained Proofs (SCP)

- Traditionally, miners require the full blockchain to validate a transaction of a user
- With SCPs a user has to provide all required proofs in the transaction in order for validators to verify a transaction independent of the blockchain



Bazo's Sharding

- Inspired by the applicability in database research

- Current approach
  - Partition the Blockchain state into shards (communities)
    - Enables parallel validation of transactions
  - Build up several shard-chains concurrently
    - Each shard has own set of transactions
- Next steps (possibly)
  - Entity management in shards according to the SCP concept
  - Save state in the newest block of a shard-chain
    - Merkle Patricia Trees

Era 3.0

Era 3.0

THETA

- URL: https://www.thetatoken.org/
- BC type: Public
  - Node types: Validators, Guardian, Normal
- Token type: Currency and Utility
- Account model: Account/Balance
- Consensus mechanism: PoS/BFT
  - 10-20 validators
  - Guardian nodes finalize blocks
- Additional mechanisms
  - Blockchain pruning, off-chain payments

© 2022 UZH, CSG@IfI

THETA Characteristics

- Major characteristics
  - Mining scheme used: Multi-Level BFT
  - Block size: up to 8,192 tx per block
    - Storage efficiency: N/A
  - Transaction size: N/A
  - Transaction rate: > 1,000 tx/s
  - Security features: Pseudo-anonymous addresses
  - Smart Contract
    - Language: Solidity
    - Key details: Ethereum-based Smart Contracts, Turing-complete, Ethereum Virtual Machine

THETA Economics

- Cryptocurrency: Yes
  - Details: THETA and TFuel (related to gas in Ethereum)
  - Exchange rates to fiat: unstable
  - Transaction fees: variable on transaction size (gas)
- Wallet https://github.com/thetatoken/theta-protocol-ledger
  - Reference implementation in Go
  - API: Web, Command-line only
- Listed on crypto exchange(s)
  - *E.g.*, Binance, https://www.binance.com
- Market capitalization (Aug 24, 2021, and May 9, 2022)
  - 7,402,131,793 USD (THETA) and 1,812,976,609 USD (TFuel)
  - 83,203,333,469 USD (THETA) and 572,222,249 USD (TFuel)

THETA Organization

- Developer community: 10 (Theta Developer Team)
  - Country of origin: United States of America (U.S.A.)
  - Deployment needs: Theta Ledger Node
  - SDKs: Android SDK, REMIX
  - Blockchain explorer: https://explorer.thetatoken.org/
  - Full or light client: Full https://github.com/thetatoken/theta-protocol-ledger
- Known vulnerabilities and attacks
  - 51% Attacks (Expensive, but doable)
- Major application domain(s)
  - Video Streaming, Live TV

THETA Overall Evaluation

- Scalability
  - High
- Centralization degree
  - 10-20 Validators (small number)
  - Limited number of Guardian nodes
- Energy efficiency
  - Energy-efficient (BFT-based)
- Security level
  - Standard BC-security
- Other dimensions of importance
  - Plans to reduce dependency from Content Delivery Networks

© 2022 UZH, CSG@IfI

Era 4.0

Era 4.0

Polkadot

- URL: https://polkadot.network
- BC type: interoperability platform
  - Network roles: Validators, Collators and Nominators
  - Network structure: Parachains, Relay chains and Bridge chains
- Token type: Utility (DOT) tokens
- Consensus mechanism: Hybrid GRANDPA/BABE
  - **G**HOST-based **R**ecursive **A**ncestor **D**eriving **P**refix **A**greement/**B**lind **A**ssignment for **B**lockchain **E**xtension
  - Hybrid and variant of Proof-of-Stake consensus

Polkadot Characteristics

- Major characteristics
  - Heterogeneous multichain with scalable security and an interoperability protocol
  - Central chain of Polkadot termed relay-chain
    - On which all validators are staked on with DOT tokens.
  - Polkadot network based on parachains
    - Parallelizable blockchains comprising the Polkadot network
    - Each parachain can have a unique architecture
    - All parachains are connected and secured by the relay chain
  - Hybrid consensus mechanism
    - Splits up the finality gadget (GRANDPA) from the block production (BABE) mechanism of the relay-chain
    - BABE runs b/w validator nodes and determines authors of new blocks

Polkadot Economics

- Cryptocurrency: Yes
  - Details: Native DOT tokens are used for 4 purposes
    - Used for governance of the network
    - Staked for operation of the network
    - Bonded to connect a chain to Polkadot as a parachain
    - Enabling interoperability between blockchains
- Multiple wallets
  - *E.g.*, Parity Signer, Fearless Wallet, Klever, Polkawallet
- Crypto exchanges
  - Binance, Huobi Global, HBTC, OKEx, Binance.KR
- Market valuation (Aug 24, 2021, and May 9, 2022)
  - 26,051,693,277 USD and 12,710,920,329 USD

© 2022 UZH, CSG@IfI

Polkadot Organization

- Polkadot is a project of the Web3 Foundation
  - Sponsoring community development and software services
    - Web3 Foundation is based in Switzerland, Zug
  - SDKs: Substrate framework
  - Blockchain explorer available: https://telemetry.polkadot.io/

- Major application domain
  - Easy deployment of public or private parachains
  - Enabling interoperability between those parachains
  - Using “Bridges” to interact with other Blockchains
    - Such as Ethereum or Bitcoin

Era 4.0

Aelf

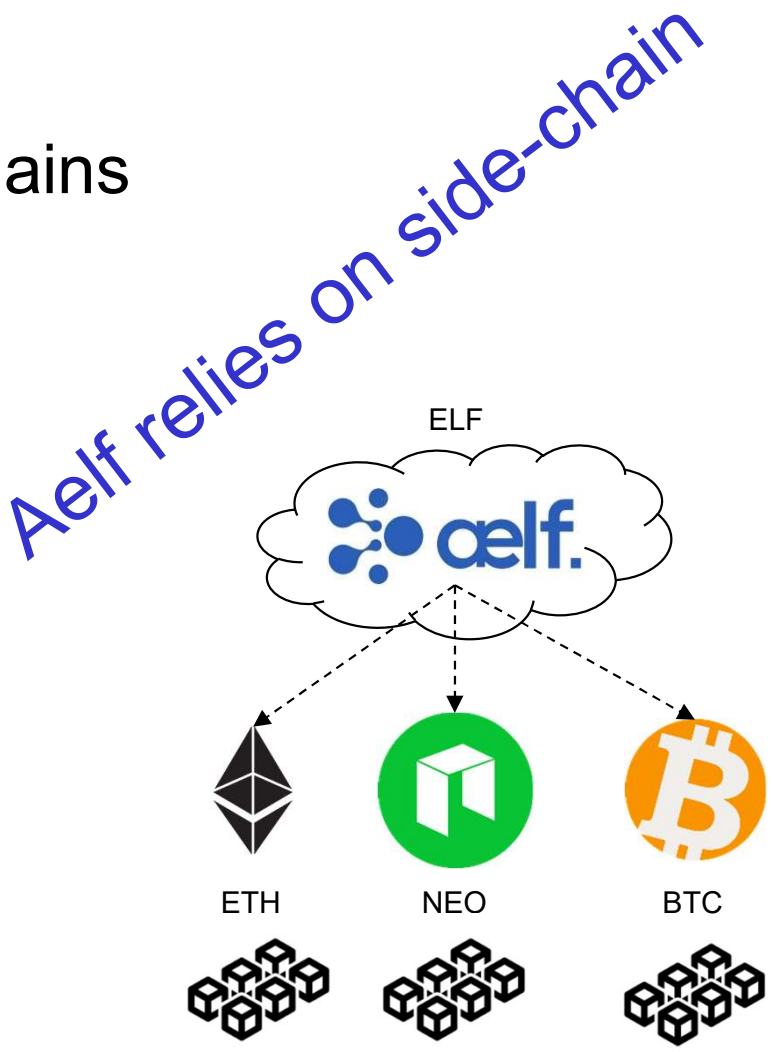
- URL: https://aelf.io/
- BC type: multi-purpose chain
  - Node types: Participant and Miner (stake)



- Token type: Currency
- Account model: Account-based
- Consensus mechanism:
  - Delegated Proof-of-Stake (dPoS)
  - Token holders cast votes to elect nodes that produce blocks
- Additional mechanisms:
  - Cross-chain communication

Aelf Characteristics

- Major characteristics
  - Mining scheme used: DPoS
  - Block size: making use of side-chains
    - Storage efficiency: N/A
  - Transaction size: variable
  - Transaction rate: 14,968 tx/s*
  - Security features: N/A
  - Smart Contract
    - Language: N/A



*Data from Aelf test scenario

Aelf Economics

- Cryptocurrency: Yes (ELF)
  - Details: multi-chain (and multi-purpose) blockchain
  - Exchange rates to fiat: unstable
  - Transaction fees: variable based on the market perception
- Wallet
  - Reference implementation: Coinomi
  - Also available on Android, iOS, Windows, Linux and MacOS
- Listed on crypto exchange(s)
  - Binance, BitMex, Poloniex
- Market capitalization
  - Cf. summary slide at end of slide deck

Aelf Organization

- Developer community: 34 contributors on GitHub
  - Country of origin: China
  - Deployment needs: Wallet
  - SDKs: Docker, Java, C#, JavaScript, Lua
  - Blockchain explorer: https://explorer.bitcoin.com/bch
  - Full or light client: full
  - Known vulnerabilities and attacks:
    - Heavy centralization due to DPoS
    - 51% Attack, i.e., single organization holds 51% of the mining capacity and may censor transactions
- Major application domain(s)
  - General purpose applications

Aelf Overall Evaluation

- Scalability
  - High
- Centralization degree
  - Partially Decentralized
- Energy efficiency
  - High
- Security level
  - Immutable, transparent
- Other dimensions of importance
  - Enable cross-chain communications

Summary

Besides BNB, USD Coin, Solana, Terra,
TerraUSD, Binance USD, Dogecoin, Avalanche,
Shiba Inu, Wrapped Bitcoin, Tron, DAI,
NEAR Protocol, Polygon:
Remaining top 20, not mentioned so far

As of May 9, 2022)

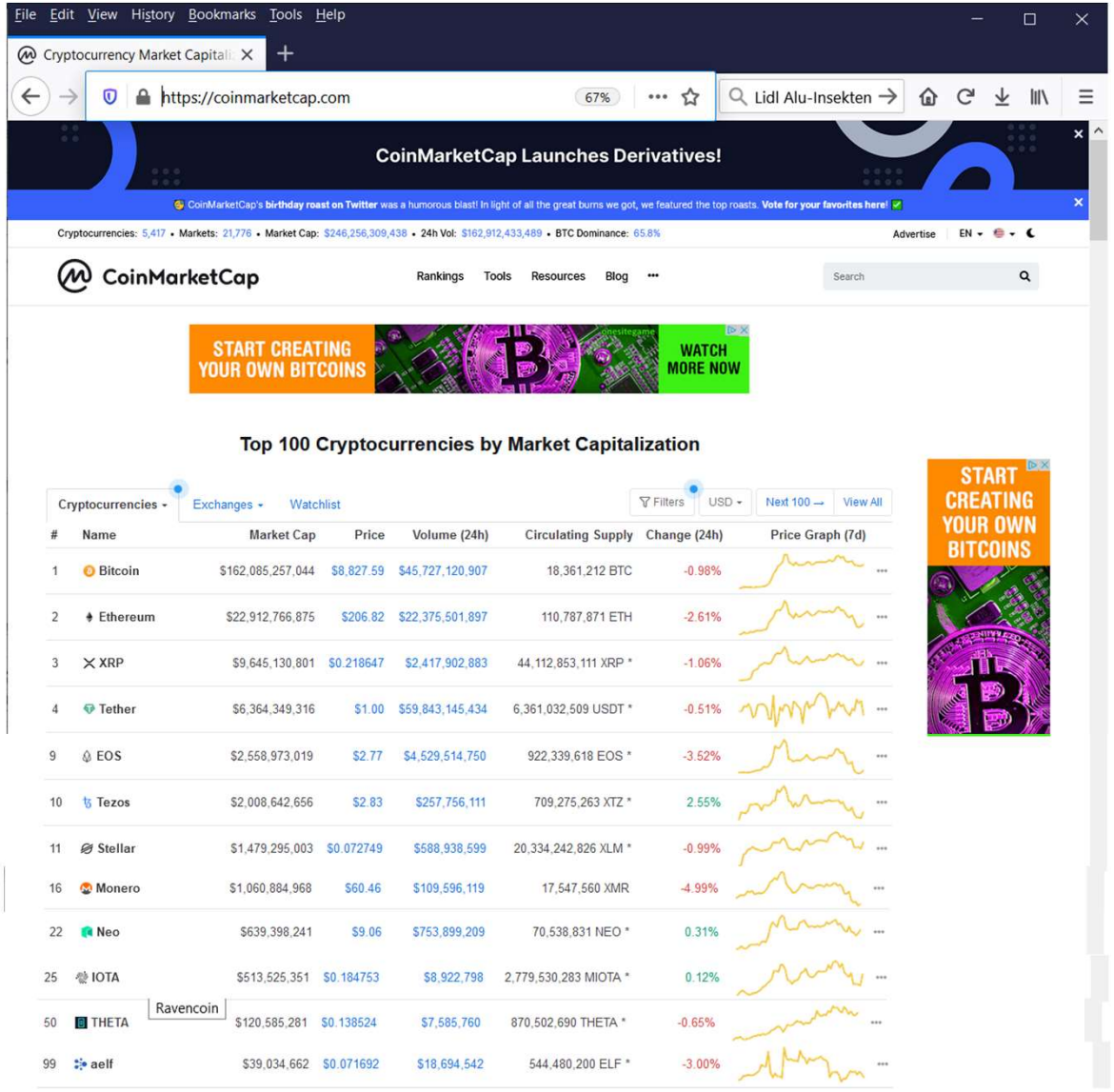
© 2022 UZH, CSG@IfI

Eras and Market Capitalization (May 5, 2020)

- https://coinmarketcap.com/

- Summary
  - Era 1.0
  - Era 2.0
  - Era 3.0
  - Era 4.0

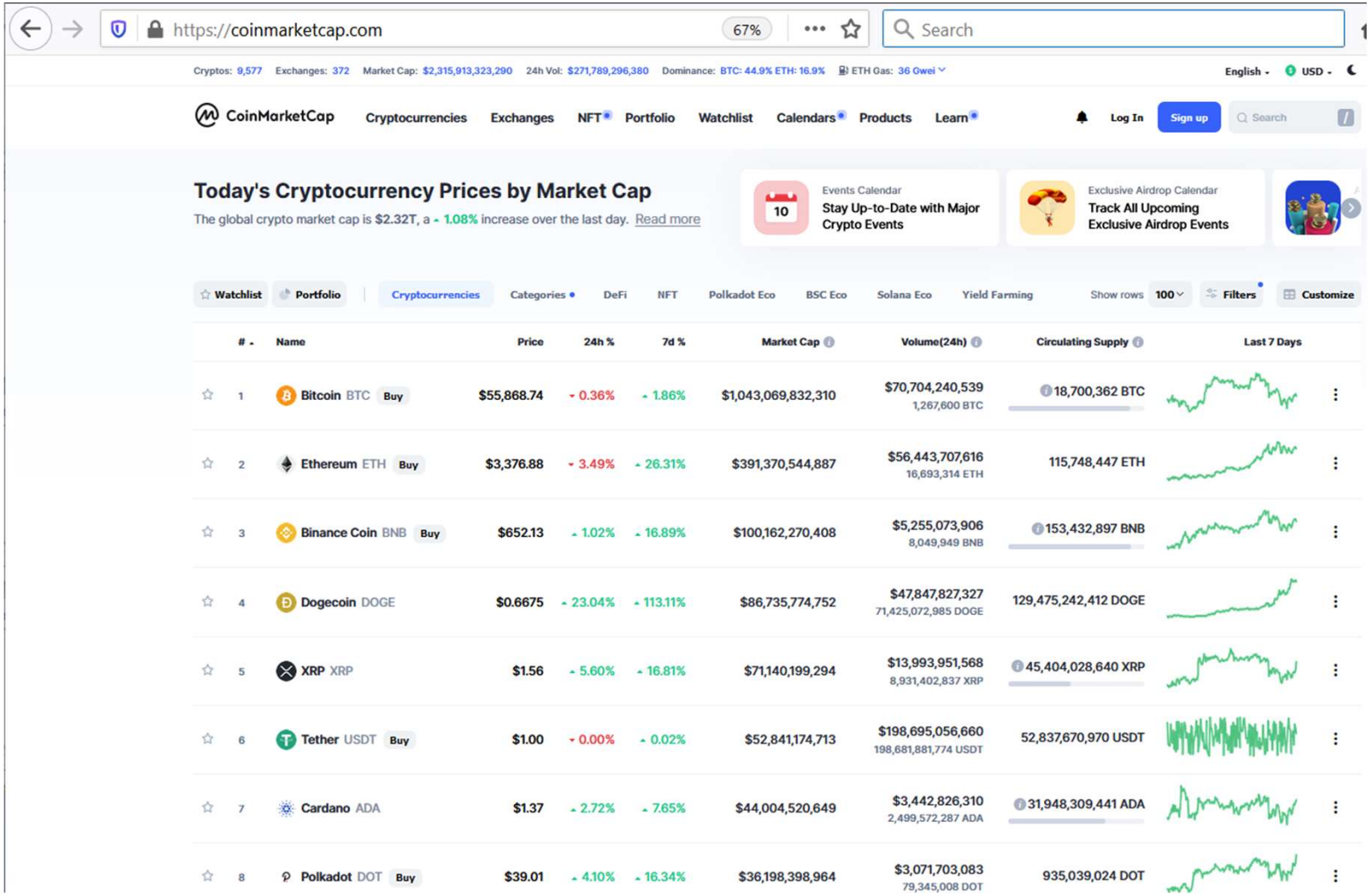
- Dimensions
  - Community-based vs. Corporation-based
  - *BCs vs. DLs*



Cryptocurrencies shown in slide deck, but not above, are not (yet) listed.

Market Capitalization (May 5, 2021)

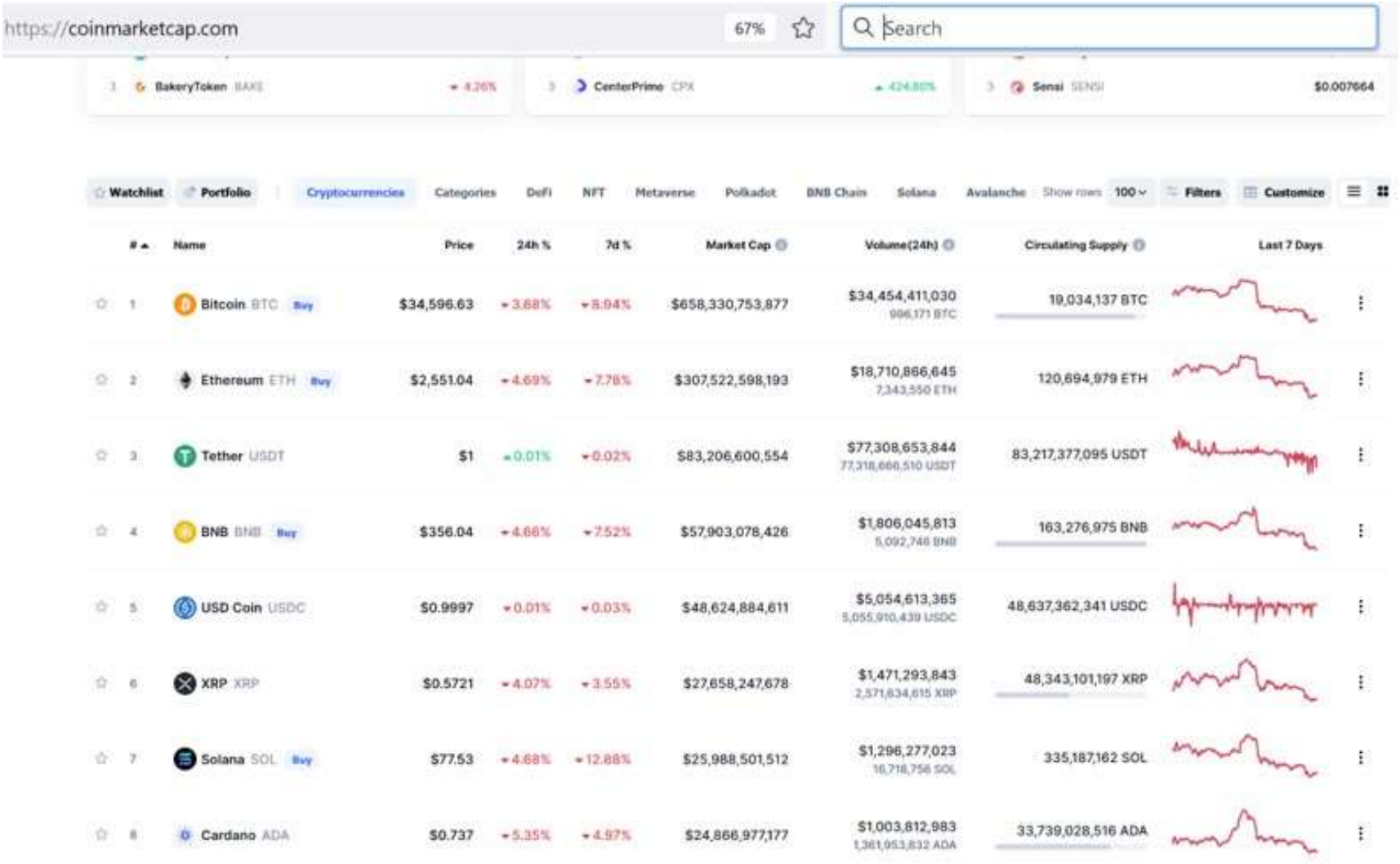
- https://coinmarketcap.com/



© 2022 UZH, CSG@IfI

Market Capitalization (May 8, 2022)

- https://coinmarketcap.com/



© 2022 UZH, CSG@IfI

Blockchains in Practice



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

Volatility of Blockchains (1)

- One very recent example of cryptocurrencies "uncontrollable" valuation https://coinmarketcap.com/
  - May 12, 2021: 6.04: 57.929 US$
  - May 12, 2021: 23.34: 54,564 US$
  - May 13, 2021, 2.04: 49,980 US$
  - May 13, 2021, 10.34: 50.622 US$
  - In-between (May 13, 2021, 12.06)



Musks Tweet liess den Bitcoin-Preis schlagartig um Tausende Dollar abstürzen. Zuletzt lag der Kurs nur noch bei 48'000 Dollar – über 15 Prozent niedriger als vor einem Tag. Allerdings standen Kryptoanlagen im allgemeinen Abwärtstrend an den Börsen zur Wochenmitte schon vor Musks Statement deutlich unter Druck. Nicht nur Bitcoin, auch die zweitgrösste Digitalwährung Ether, das Krypto-Meme Dogecoin und andere Cyberdevisen wie Binance Coin und Ripple gaben stark im Kurs nach.

https://www.tagesanzeiger.ch/tesla-stoppt-zahlungen-mit-bitcoins-wieder-370310879909

May 18, 2021, 11.35: 44.918 US$

© 2022 UZH, CSG@IfI

Volatility of Blockchains (2)



As of May 11, 2022

ToC

- Welcoming two guests with industrial experience
  - The two look back on CSG research in the past
  - The two received their PhD degrees from UZH at the CSG
    - They employed in their teams many IfI students so far ☺
- DAOs: Concepts and Practice
  - Dr. Guilherme Machado, AxLabs
- DeFi
  - Dr. Thomas Bocek, Axelra AG

DAOs:
Concepts and Practice

Dr. Guilherme Sperb Machado

AxLabs



guil@axlabs.com

https://axlabs.com



AxLabs

- Software Engineers, based in Zürich – specialized in Blockchain Engineering
- Using bleeding-edge tech to build rock-solid products
- Focus on decentralized solutions and open protocols

- Some of the projects:
  - **neow3j**: Java/Kotlin/Android toolkit to develop on the Neo blockchain
  - **GrantShares:** A decentralized and transparent grants program for the Neo ecosystem
  - **Flatfeestack**: “Software Engineers should be able to make a living with open-source”
  - **Yaky**: Experience and learn web3
  - ICOnator: A secure, configurable, and scalable open source ICO engine

- Projects in the industry: modum, procivis, swisscom Blockchain, VETRI, ARC

- Neo Blockchain: AxLabs is a Council Member neo

AXLABS

AxLabs Team



Dr. Guilherme Sperb Machado

guil@axlabs.com



Claude Müller

claude@axlabs.com



Michael Bucher

michael@axlabs.com

Agenda

- What are DAOs?
- Why do we need DAOs?
- Types of DAOs
- How do DAOs work?
- Examples
- GrantShares DAO
  - Governance Flow
  - Intents & Proposal Types
  - Neo Blockchain & Tooling
  - Demo
- Q&A Session

What are DAOs?

- **D**ecentralized **A**utonomous **O**rganization

- "[…] is an organization constructed by rules encoded as a computer program that is transparent, controlled by the organization's members and not influenced by a central government" [1]

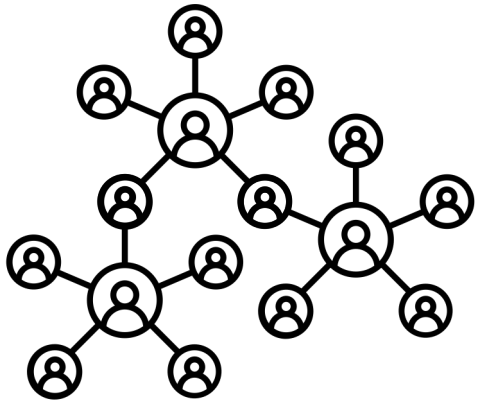
- Member-owned communities without centralized leadership

- The DAO (Ethereum, 2016)
  - Investor directed VC fund
  - The DAO Hack: 3.6million ETH
  - Who hacked the DAO?

[1] https://en.wikipedia.org/wiki/Decentralized_autonomous_organization

Why do we need DAOs?

- Trust
- Collaboration
- Transparency
- Verifiability of actions within an organization



Model Articles of Association of a DAA (Decentralized Autonomous Association): https://bit.ly/3P6GO1r

High-Level Comparison

Tradition Organization	DAO
Usually **hierarchical**	Usually **flat**, and **fully democratized**
Depending on the structure, changes can be demanded from **a sole party**, or **voting** may be offered	**Voting required** by members for **any changes to be implemented**
If voting allowed, **votes are tallied internally**, and outcome of voting must be **handled manually**	Votes tallied, and **outcome implemented** automatically **without trusted intermediaries**
Requires human handling, or **centrally controlled automation**, prone to **manipulation**	**Services** offered **are handled automatically in a decentralized way** (e.g., distribution of philanthropic funds)
Activity is **typically private**, and limited to the public	All activity is **transparent and fully public**

Source: https://ethereum.org/en/dao

Example

- Let's assume a simple corporation running a chain of stores

- Three classes of members:
  - Investors (and co-founders)
  - Employees
  - Customers

- Membership rule for *Investors:*
  → Buy shares to get in; Sell shares

- Membership rule for Employees:
  → Get hired by Investors; Fired

- Membership rule for Customers:
  → Anyone can freely interact with the store (open-membership)

DAO Membership Types

- **Token-based**
  - Usually fully permissionless
  - Tokens are traded permissionlessly on a DEX (Decentralized Exchange)
  - Holding tokens provide access to voting
    - E.g., 1 token = 1 vote
  - Example: MakerDAO
- **Share-based**
  - Usually, more permissioned but still open to membership
  - Prospective members can submit a proposal to join the DAO in exchange of work or staked value
  - “Shares” represent direct voting power and ownership
  - Example: MolochDAO

Challenges with Coin Voting

Mitigation: Delegation

1) Misalignments even in the absence of attackers

- Whales versus Small Holders
  - Small groups of wealthy participants → More agile & unified decisions
  - Large group of small-holders → Not agile & fragmented decisions
- Coin holder interests at the expense of other parts of the community
- Conflict of interest
  - E.g., investment funds or holders that also hold tokens of other DAO-based DeFi platforms

2) Attacks through forms of vote buying

- Bundle of two rights that are combined into a single asset:

Economic interest in protocol's revenue

+

The right to participate in the governance

Source: https://vitalik.ca/general/2021/08/16/voting3.html

Other Types of DAO Membership

- **Proof of personhood systems**
  - Verify that accounts belong to specific humans
  - 1 human = 1 vote
  - E.g., ProofOfHumanity, BrightID

- **Proof of participation**
  - Ensure that accounts corresponds to a person that has participated in something or performed work in the ecosystem
  - E.g., POAP, SourceCred, Discord-based communities with airdrops
  - But there are challenges with automated evaluation of participation: is it always “fair”?

→ Demo: communities using Discord

Source: https://vitalik.ca/general/2021/08/16/voting3.html

How do DAOs work?

- **Backbone: Smart Contracts**
- Usually: Governance + Treasury
- Set of rules
- Set of parameters
- Specific or Generic execution model

- Example of parameters:
  - Quorum
  - Acceptance ratio of proposals
  - Proposal expiration
  - Voting period for a proposal
  - Maximum amount to be released from the treasury at once
- Example of rules:
  - If *X* address is a member for *t* period, allow voting; not otherwise

Useful Tools and Docs

- OpenZeppelin Wizard
  - https://wizard.openzeppelin.com
- Docs on how to set-up on-chain Governance
  - https://docs.openzeppelin.com/contracts/4.x/governance

Examples of DAOs

- https://www.ukrainedao.love
  - Donations to Ukrainians
- https://gov.gitcoin.co
  - Govern the future of Gitcoin, a grants platform
- https://constitutiondao.com
  - Single-purpose DAO: raise capital to buy the original copy of the US constitution (Sotheby's auction)
- https://1of1.works
  - Manage NFTs, Proof of Art Work
- https://radicle.xyz
  - DAO to govern Radicle: "GitHub, but fully decentralized with p2p tech"
- https://andao.aragon.org
  - Aragon is a network providing various tools and infrastructure for DAOs
- https://metacartel.org
  - Focused on investments in new dApp projects (in return of tokens)
- https://boredapeyachtclub.com
  - Social club
- https://grantshares.io
  - Grants platform to foster the **Neo blockchain** ecosystem, governed by a DAO



- https://grantshares.io
- Governed by a DAO
- Initially, Neo Foundation provides funds to the treasury
- Later, treasury could accept funding from third-parties
- Permissioned DAO first, hybrid and permissionless later

- Initial members

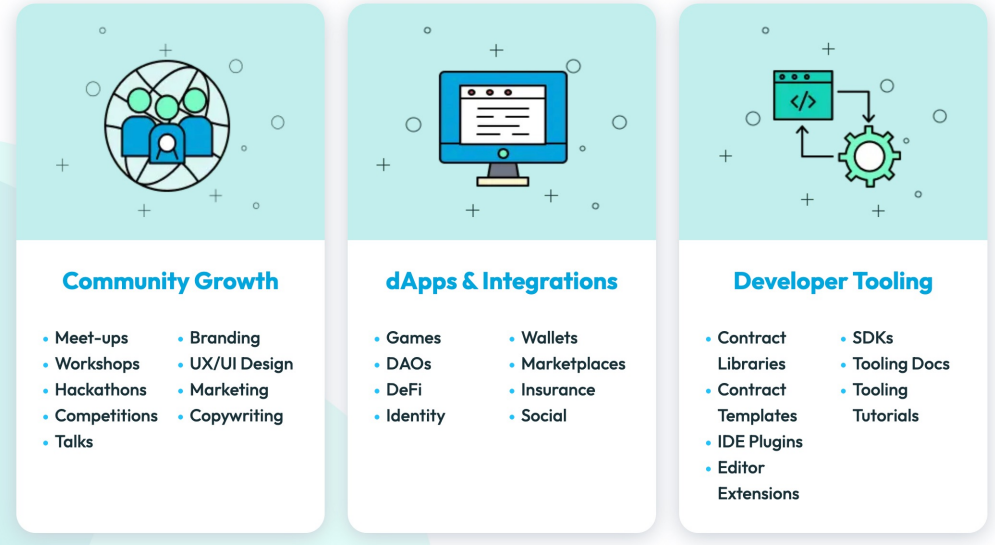
coz · AXLABS · RED4SEC · ngd · NEXT · SPCC NEO · neo | Foundation · neo | global development · NEONEWSTODAY

Source: https://neonewstoday.com/governance/grantshares-dao-launching-to-support-ecosystem-growth-and-grassroots-initiatives/ AXLABS

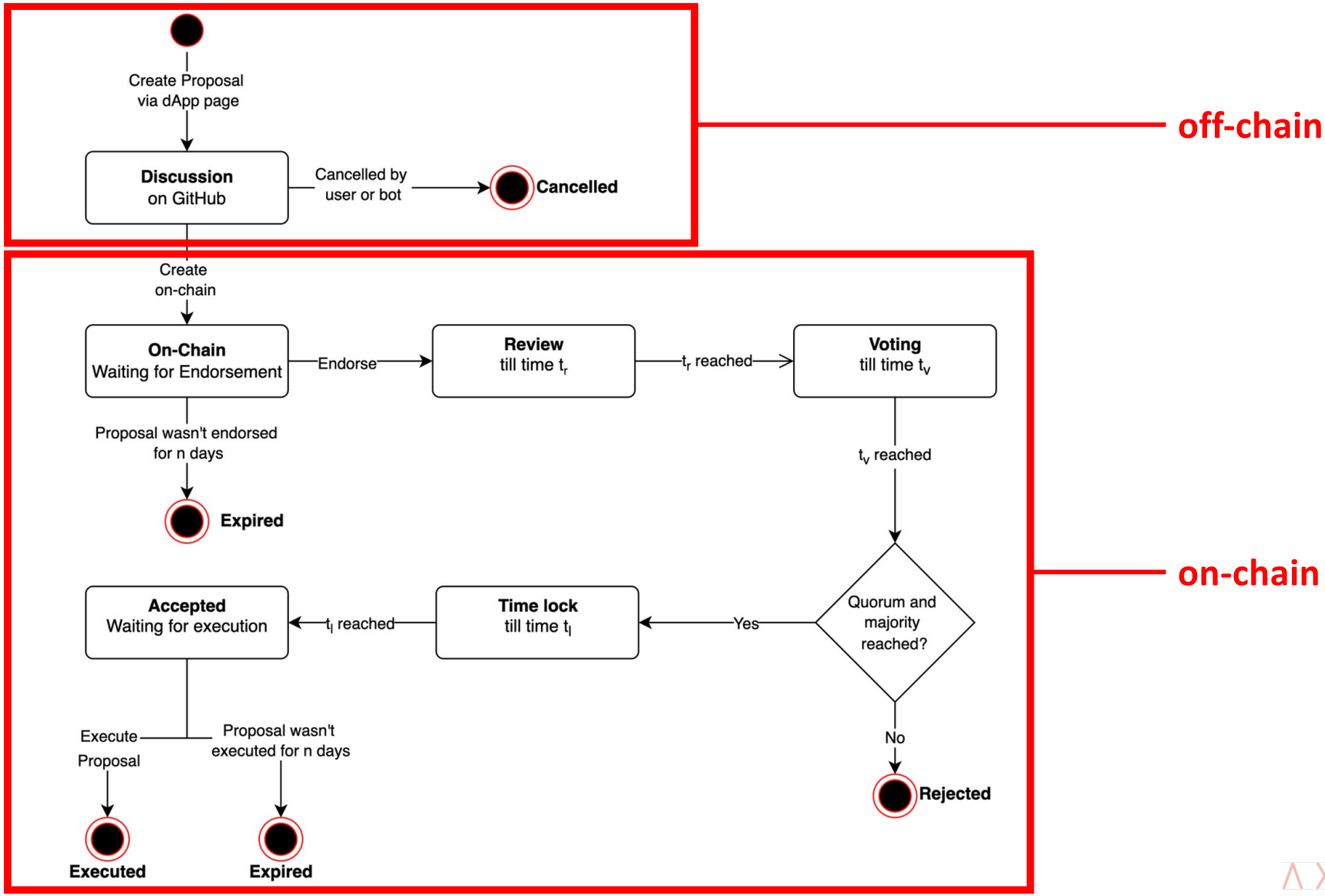
GrantShares Concepts

- Anyone can submit a proposal!
  - No need to send transactions or pay GAS fees (or anything related to blockchain)
  - Really low barrier entry
- Community openly discusses the proposal and give suggestions
- Strong integration with GitHub (off-chain process)

- Initial funding limit
  - ~$50,000 per proposal



AXLABS


Create Proposal via dApp page
Discussion on GitHub
Cancelled by user or bot
Cancelled
off-chain
Create on-chain
On-Chain Waiting for Endorsement
Endorse
Review till time t_r
t_r reached
Voting till time t_v
Proposal wasn't endorsed for n days
Expired
t_v reached
Quorum and majority reached?
Yes
Time lock till time t_l
t_l reached
Accepted Waiting for execution
No
Rejected
Execute Proposal
Proposal wasn't executed for n days
Executed
Expired
on-chain

AXLABS

How does a proposal technically work? [1]

- Amongst other parameters, a proposal is composed of:

Intents

- What should be invoked? (e.g., smart contract address)
- Which kind of information should be given as a parameter to such a SC invocation?
- What are the permissions for the SC invocation?

```
/**
 * Creates a proposal.
 *
 * @param proposer       The account set as the proposer.
 * @param intents        The intents to be executed when the proposal is accepted.
 * @param offchainUri    The URI of the part of the proposal that exists off-chain.
 *                       E.g., a unique discussion URL.
 * @param linkedProposal A proposal that preceded this one.
 * @param acceptanceRate The desired acceptance rate.
 * @param quorum         The desired quorum.
 * @return The id of the proposal.
 */
1 usage  Claude Muller *
public static int createProposal(
        Hash160 proposer,
        Intent[] intents,
        String offchainUri,
        int linkedProposal,
        int acceptanceRate,
        int quorum
) {...}
```

Source: GrantShares Smart Contract source code, https://bit.ly/3M83EUg

- Proposal types are just groups of “intents” that achieve a goal

- Example:
  - Proposal type: ‘request-for-funding’
  - Proposal type: ‘add-member’
  - …

```
{
  // The target contract for request-for-funding requests
  // is always the Treasury smart contract
  "target_contract": "0x6276c1e3a68280bc6c9c00df755fb691be1162ef",
  // The target method for funding requests is alwasy "releaseTokens"
  "method": "releaseTokens",
  "params": [
    {
      // Token Contract
      "type": "Hash160",
      "value": "0xef4073a0f2b305a38ec4050e4d3d28bc40ea63f5"
    },
    {
      // Receiver
      "type": "Hash160",
      "value": "0x3ff68d232a60f23a5805b8c40f7e61747f6f61ce"
    },
    {
      // Amount, no decimals allowed.
      "type": "Integer",
      "value": "100"
    }
  ],
  "call_flags": 15
}
```

Validation checks

Execution of each intent

```
public static Object[] execute(int id) {
    abortIfPaused();
    ByteString proposalBytes = proposals.get(id);
    if (proposalBytes == null)
        fireErrorAndAbort( msg: "Proposal doesn't exist",  method: "execute");
    Proposal proposal = (Proposal) deserialize(proposalBytes);
    if (proposal.endorser == null || getTime() < proposal.timeLockEnd)
        fireErrorAndAbort( msg: "Proposal not in execution phase",  method: "execute");
    if (proposal.executed)
        fireErrorAndAbort( msg: "Proposal already executed",  method: "execute");
    if (proposal.expiration <= getTime())
        fireErrorAndAbort( msg: "Proposal expired",  method: "execute");
    ProposalData data = (ProposalData) deserialize(proposalData.get(id));
    ProposalVotes votes = (ProposalVotes) deserialize(proposalVotes.get(id));
    int voteCount = votes.approve + votes.abstain + votes.reject;
    if (voteCount * 100 / Storage.getInt(getReadOnlyContext(), MEMBERS_COUNT_KEY) < data.quoru
        fireErrorAndAbort( msg: "Quorum not reached",  method: "execute");
    int yesNoCount = votes.approve + votes.reject;
    if (votes.approve * 100 / yesNoCount <= data.acceptanceRate)
        fireErrorAndAbort( msg: "Proposal rejected",  method: "execute");

    proposal.executed = true;
    Object[] returnVals = new Object[data.intents.length];
    proposals.put(id, serialize(proposal));
    for (int i = 0; i < data.intents.length; i++) {
        Intent t = data.intents[i];
        returnVals[i] = Contract.call(t.targetContract, t.method, t.callFlags, t.params);
    }
    executed.fire(id);
    return returnVals;
}
```

Source: GrantShares Smart Contract source code, https://bit.ly/3srAckk

GrantShares: Demo

- Create a proposal (off-chain)
- Put the proposal on-chain
- Endorse, vote, and execute it (on-chain)

Neo Blockchain

- Neo Project was founded in 2014
- Open-sourced on GitHub in June 2015
- Neo is an open-sourced and non-profit global community
- Developers across the globe join and contribute to Neo



https://github.com/neo-project/neo

- Smart contract support
- Often compared to Ethereum
  - "The closest, most practical, and sustainable chain to compete with ETH"
- Neo N3 is a new chain and an evolution of Neo Legacy
  - Assets and smart contracts migrated



NEO



GAS

More info about the duo-token model:
https://neo.org/neogas#tokens

Neo Smart Contracts

- Neo VM (Neo Virtual Machine)
  - Execution environment
  - Stack-based VM

- Turing Complete

- **Unlike Ethereum, Neo has support for high-level programming languages**

Neo Blockchain: Vision w.r.t. Smart Contract Dev

- “No need to learn a new programming language to write your decentralized application”

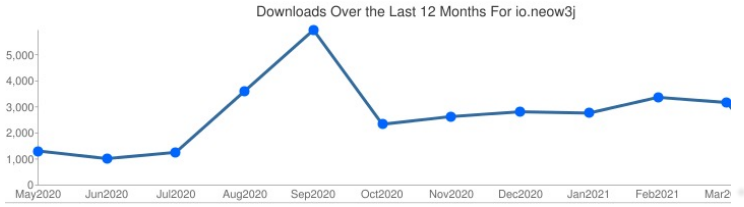
- Today:
  - Go, C#, **Java**, Python, Javascript, Typescript, C++

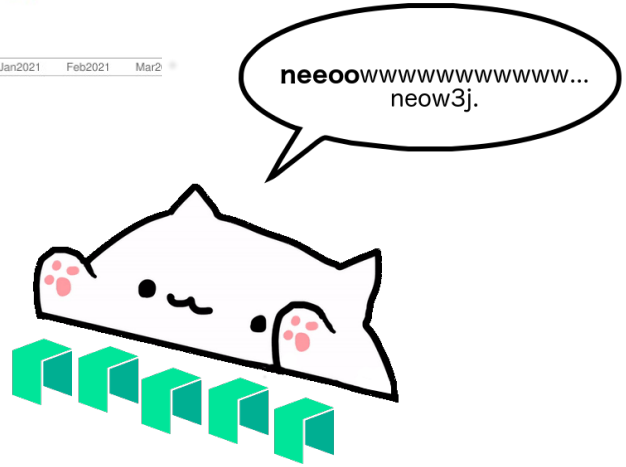
- But, in 2018:
  - No Java, no Kotlin, no JVM-based language
  - Demand for JVM-based languages (mainly companies due maintanability)



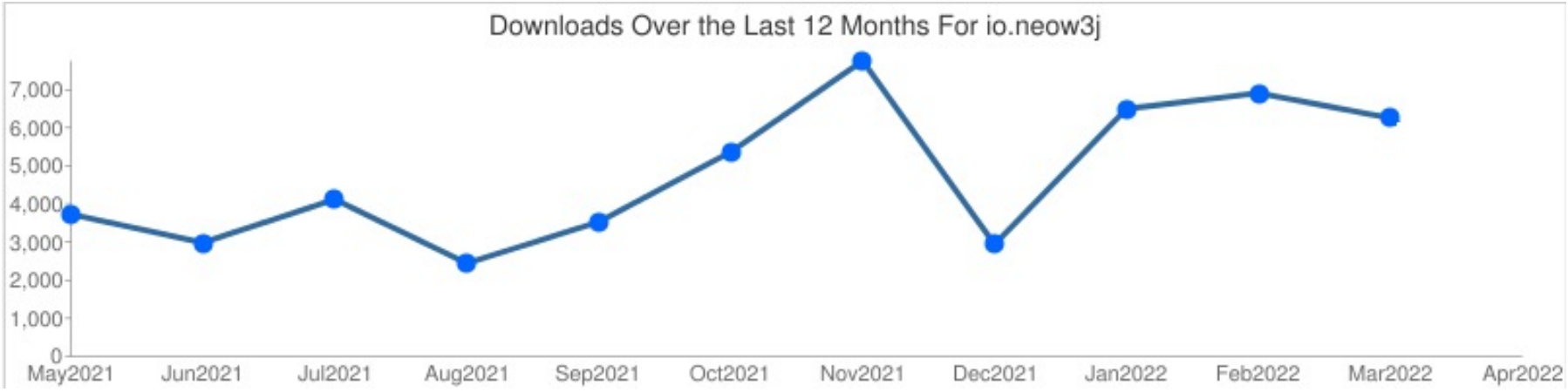
AXLABS

neow3j [1]





- neow3j SDK (Software Development Toolkit)
- neow3j compiler (write in Java/Kotlin, run on Neo)
- Java/Kotlin/JVM Community
- First release: end of 2018

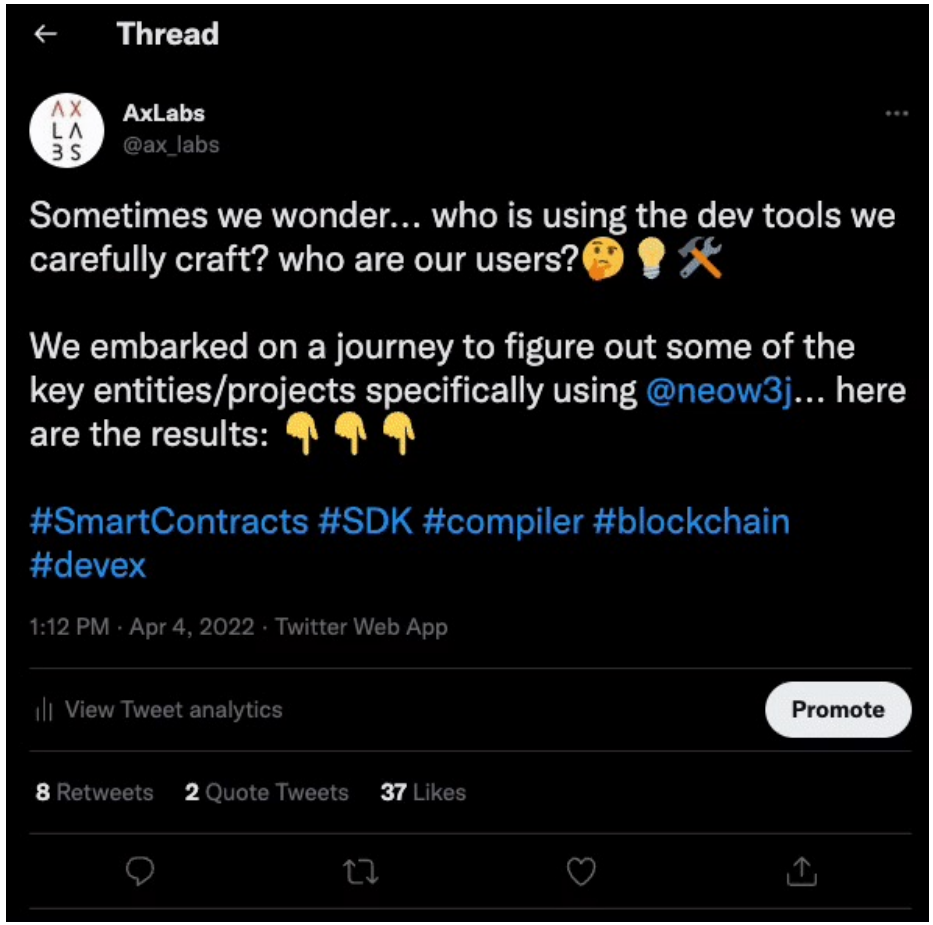


Source: https://oss.sonatype.org

AXLABS

neow3j [2]

- Who's using it?
- We started a journey to find out
  - Interviews, chatting with the community, and checking on GitHub
- Results
  - https://twitter.com/ax_labs/status/1510938397908557830



AXLABS

neow3j [3]

Hello World Smart Contract

```
1  package io.neow3j.examples;
2
3
4  public class HelloWorldContract {
5
6  }
7
```

- Several Smart Contract examples:
https://github.com/neow3j/neow3j-examples-java

Click here to watch the GIF in the browser

GrantShares DAO: Smart Contracts

- Open source!
- Written using neow3j
- https://github.com/AxLabs/grantshares-contracts
- Audited (soon to be released)
- AxLabs Team accomplishment 🚀🎉
  - Developed the entire stack:
    from a compiler, to SDK, and now a dApp

GS-01	**GS-01**	Denial of Service in Neo Candidate selection	High
GS-02	**GS-02**	Use of ASSERT instead of THROW	High
GS-04	**GS-03**	Vulnerable to Proposal Flooding	Medium
GS-05	**GS-04**	Lack of Inputs Validation in Treasury Contract	Medium
GS-06	**GS-05**	Lack of Inputs Validation in Governor Contract	Medium
GS-07	**GS-06**	Incomplete OnNEP17Payment logic	Medium
GS-08	**GS-07**	Insecure Multisig Threshold	Low
GS-09	**GS-08**	Wrong Pausable Logic	Low
GS-11	**GS-09**	Denial of Service in the Query Methods	Low
GS-12	**GS-10**	Improvable Intents	Low
GS-03	**GS-11**	Abuse of discussionUrl argument	Informative
GS-13	**GS-12**	Safe Storage Access	Informative
GS-14	**GS-13**	Missing NEF Information	Informative
GS-15	**GS-14**	Improper Call Rights	Informative
GS-16	**GS-15**	GAS Optimization	Informative
GS-17	**GS-16**	Unify storage prefixes	Informative
GS-19	**GS-17**	Open to-do	Informative

What would you build as a DAO?

How would you architect the smart contracts and governance model?

Try it out!
Build something!
Engage a community!
Learn and enhance it.

What about building something together?

- Passionate about open-source?
- Have an idea to extend or enhance existing blockchain tool?
- Have an idea to build a dApp and bring to the public?
- Or, maybe, would you like to **turn your Challenge Task work** into a real-world dApp with the *potential to be supported* by a public chain?

- **Contact us.**



https://axlabs.com



https://grantshares.io

Thanks! Questions?

AXLABS

https://axlabs.com



OST
Eastern Switzerland
University of Applied Sciences

MINF4224 - Blockchains and Overlay Networks

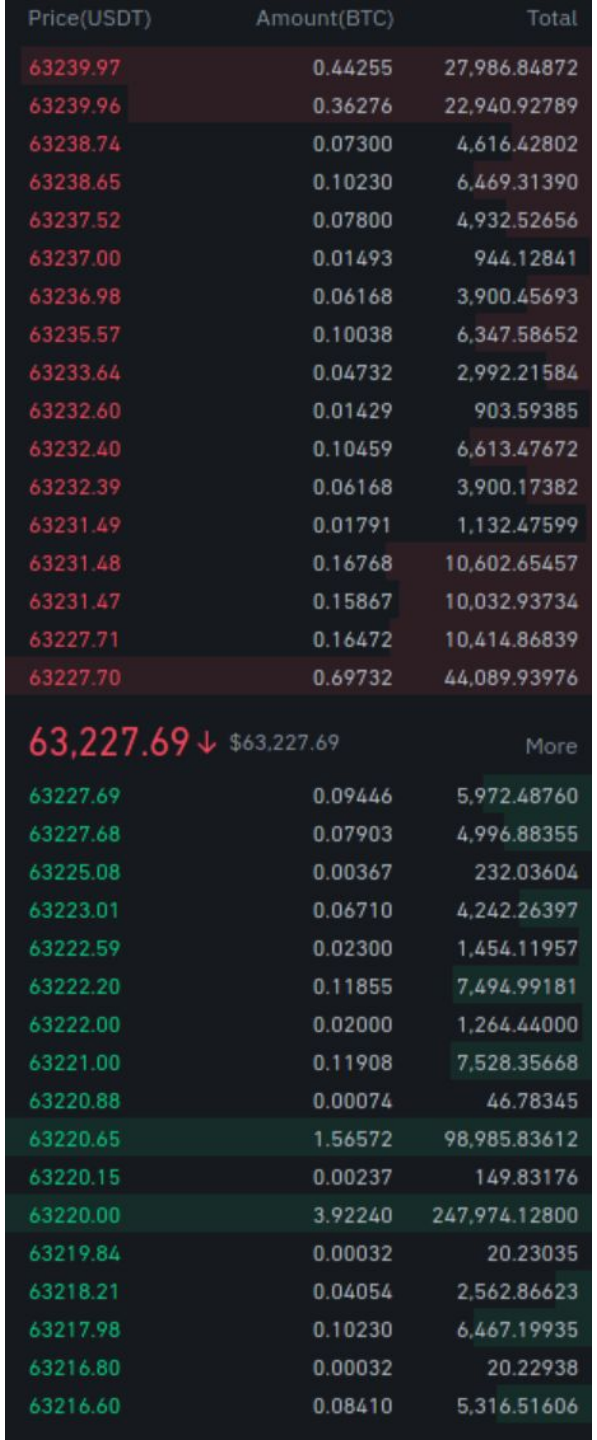
DeFi

Thomas Bocek

08.05.2022

Exchange Rate

- Centralized (CEX): ask/bid, sell/buy, the last trade, e.g., 200 DAI for 1 ETH → price (order book)
  - Workflow: create order, publish on exchange, wait to get filled. Browse orders, start fill order.
  - Prince changes if trade happens, ask was same or lower than bid. Ask/bid submitted by users – add/remove orders
  - Slippage: you see a price, submit, and until its executed, price can change.
    - Set limits, order may stay in the orderbook
- Decentralized (DEX): ratio of pairs (automatic market making)
  - Workflow: exchange pairs
  - Slippage: the “same” - sometimes (mis)used as price impact
  - Example amount in pool: DAI 200, ETH 1 → price 200DAI/1ETH
- Both: large swap can change price



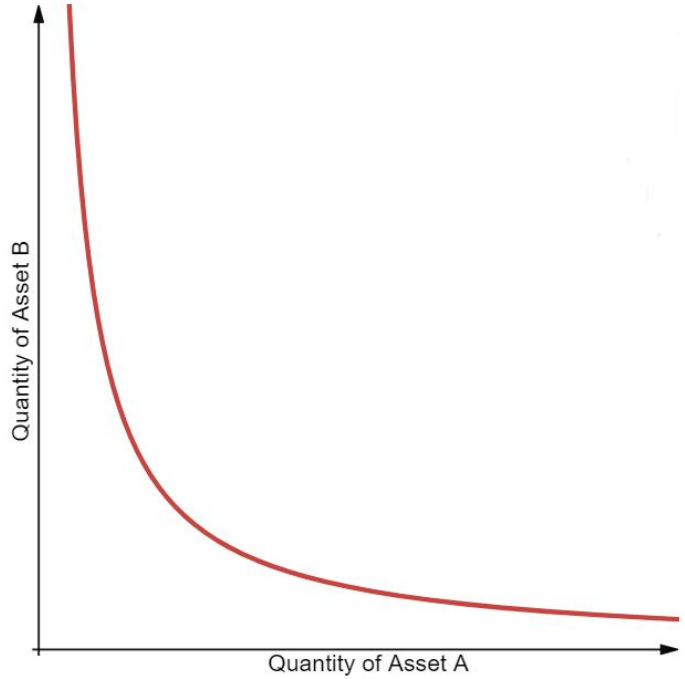
Price(USDT)	Amount(BTC)	Total
63239.97	0.44255	27,986.84872
63239.96	0.36276	22,940.92789
63238.74	0.07300	4,616.42802
63238.65	0.10230	6,469.31390
63237.52	0.07800	4,932.52656
63237.00	0.01493	944.12841
63236.98	0.06168	3,900.45693
63235.57	0.10038	6,347.58652
63233.64	0.04732	2,992.21584
63232.60	0.01429	903.59385
63232.40	0.10459	6,613.47672
63232.39	0.06168	3,900.17382
63231.49	0.01791	1,132.47599
63231.48	0.16768	10,602.65457
63231.47	0.15867	10,032.93734
63227.71	0.16472	10,414.86839
63227.70	0.69732	44,089.93976
63,227.69 ↓ $63,227.69		More
63227.69	0.09446	5,972.48760
63227.68	0.07903	4,996.88355
63225.08	0.00367	232.03604
63223.01	0.06710	4,242.26397
63222.59	0.02300	1,454.11957
63222.20	0.11855	7,494.99181
63222.00	0.02000	1,264.44000
63221.00	0.11908	7,528.35668
63220.88	0.00074	46.78345
63220.65	1.56572	98,985.83612
63220.15	0.00237	149.83176
63220.00	3.92240	247,974.12800
63219.84	0.00032	20.23035
63218.21	0.04054	2,562.86623
63217.98	0.10230	6,467.19935
63216.80	0.00032	20.22938
63216.60	0.08410	5,316.51606

OST

Exchange Rate / Decentralized Swaps

- DEX uses X * Y = k, where k is constant, X and Y are asset values (if you take out X you need to provide Y)
  - DAI = 200, ETH = 1, k = 200
- Constant function market makers (CFMM)
  - “We are still very early in the evolution of constant function market makers...” [ref]
- Exchange price calculation (X * Y = k) → X * Y = (X + x) * (Y - y)
  - Swap for 0.5 ETH, if you send 0.5 ETH to pool
    - 200 * 0.5 / (1+0.5) = 66
    - 66/0.5 → 133 DAI → ~**133 DAI for 1 ETH**
  - Deduct 66 DAI from pool, add 0.5 ETH → 133/1.5 → ~ **88 DAI for 1 ETH**
    - K = 133.333 * 1.5 = 200
    - Never draining the pool
    - Trade with better price than resulting pool (huge price impact)

$$y = \frac{Yx}{X + x}$$




```
x is input asset amount   (ETH)
X is input asset balance  (ETH)
y is output asset amount  (DAI)
Y is output asset balance (DAI)
```


OST

Exchange Rate / Decentralized Swaps

```
x is input asset amount   (ETH)
X is input asset balance  (ETH)
y is output asset amount  (DAI)
Y is output asset balance (DAI)
```

- Reverse of 133/1.5 → I want to buy ETH with 66 DAI
  - 1.5*66/(133+66) = 0.5 $x = \frac{Xy}{Y+y}$
  - Swap at price 133.333 → new price 200DAI/ETH → reversible
- If swap price == the final pool price
  - $\frac{Y-y}{X+x} = \frac{y}{x}$ $y = \frac{Yx}{X+2x}$
  - Example 200*0.5/(1+2*0.5) = 50
    - 150/1.5 = 50/0.5 (swap and final price the same)
  - However, not reversible with same numbers. To get the same price (200), need to swap 75 DAI for 0.375ETH → 225/1.125ETH
  - THORChain $y = \frac{xYX}{(x+X)^2}$

OST

Decentralized Swap

- Many AMM variations
  - THORChain – punish large swaps [how its calculated]
  - Example: 0.5 * 200 * 1 / (0.5 + 1)^2 = 44.4 (price 88DAI/1ETH)
  - Resulting pool: 155.555/1.5 → price 103.7DAI/1ETH
    - Large trades gives you a worse rate than the resulting pool price. Small values, e.g., 0.1 ETH → 16.5DAI / 165DAI/ETH, pool: 166.8DAI/ETH
- More AMMs, here
- Attacks: Exploit slippage tolerance: sandwich attack (front-running) [seen in practice]

```
x is input asset amount   (ETH)
X is input asset balance  (ETH)
y is output asset amount  (DAI)
Y is output asset balance (DAI)
```

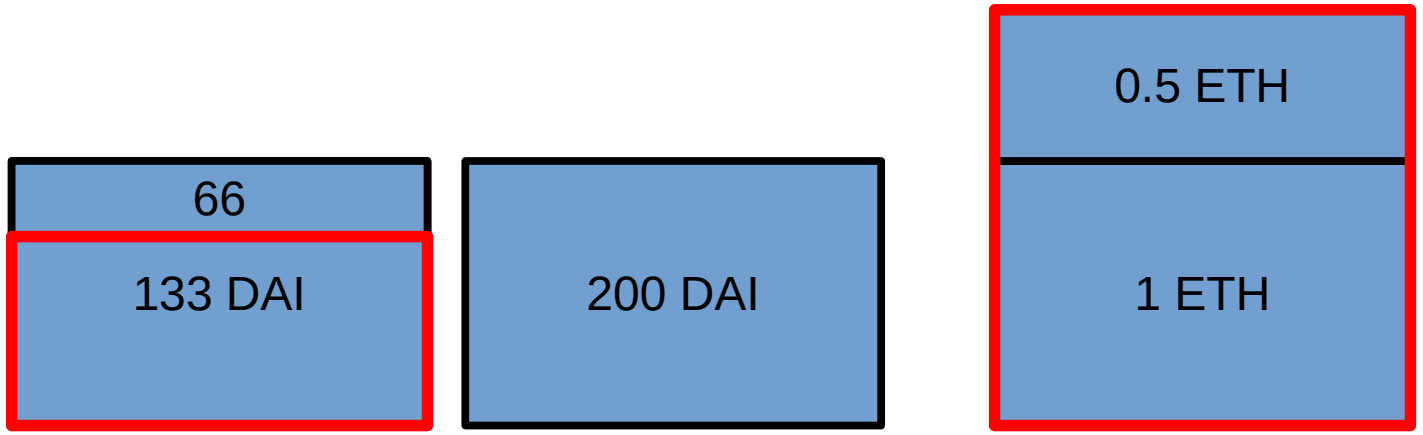
$$y = \frac{xYX}{(x+X)^2}$$

OST

Decentralized Swap

- Swap 0.5 ETH for DAI, how much DAI? (price 200DAI/ETH)
- (price 133DAI/ETH), but DAI funds not decreased yet
- Pool: 133 DAI, 1.5 ETH, price (88DAI/ETH)

$$y = \frac{Yx}{X + x}$$



OST

DeFi Fundamentals

- Swaps (just covered)
- Arbitrage bots
  - Swapping in multiple pools or CEX, if a boot sees e.g., a trading opportunity,
    - Example: Pool 1: 250 DAI for 1 ETH, pool 1: 200 DAI for 1 ETH
    - Arbitrage bot has 1 ETH (not considering price impact in this example)
      - Buy for 1 ETH 250 DAI in pool 1
      - Sell 250 DAI for 1.25 ETH in pool 2, profit = 0.25 ETH
  - Keep the same price across exchanges
- Flash loans
  - Same chain: arbitrage bots can use flash loans
  - Get loan and pay it back in the same transaction
    - No risk for lender, either he gets the tokens back or transaction is invalid
- Example with Aave: from 1000 to 1045, including payback of 1000.9
  - SushiSwap, Uniswap, DAI, wETH
- Without arbitrage bots, DEX does not work!
  - Essential part of the ecosystem

OST

DeFi Fundamentals

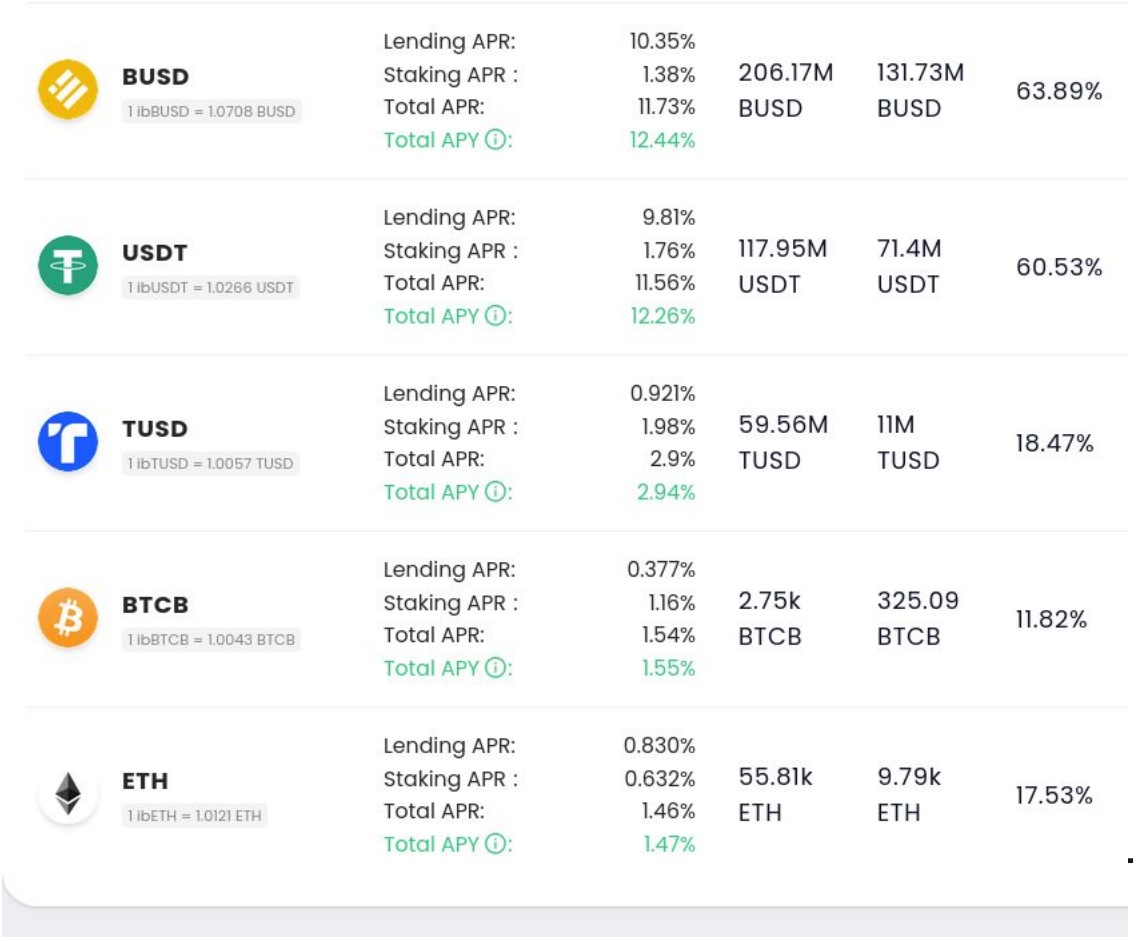
- Swaps (just covered)
- Arbitrage bots (just covered)
- Liquidity providers (LP)
  - Someone should provide liquidity - filling the pools
  - General rules for AMM-based DEX
    - Providing / removing liquidity - **no change of price**
    - LP provide 50/50 ratio of assets, example with a 200DAI/1ETH pool
    - LP can provide 100DAI/0.5ETH, or 400DAI/2ETH
    - LP_a gets liquidity units (LP token), with 100/0.5 (new pool: 300/1.5), LP_a owns 1/3 of the pool
- Why should a LP provide liquidity?

OST

Liquidity Providing

- LP token → % of the liquidity provided in the pool
  - For each swap, user has to pay fees
    - Fees are distributed proportionally to the amount of LP tokens
    - Eg., fees collected are 2ETH, LP gets 10%, 0.2ETH
  - Earn fees for providing liquidity
  - E.g., LP provides 20DAI/0.1ETH, pool is 200/1 → LP tokens says its 10% of the pool
    - Eg., more liquidity provided by others, the 10% will be decreased → less earnings

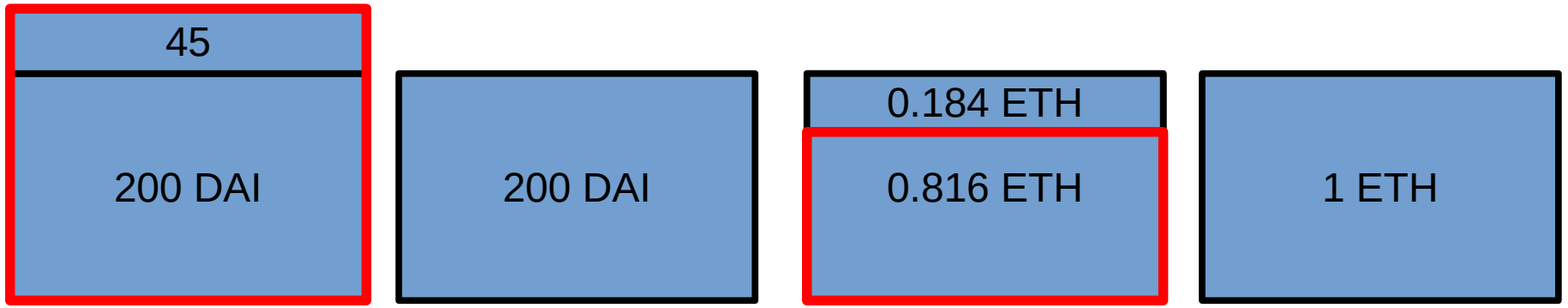
- Why is not everyone providing liquidity?
  - Free money?



Liquidity Providing

- Impermanent Loss (its mostly permanent, better name: price shift loss)
  - “Users who provide liquidity to AMMs can see their staked tokens lose value compared to simply holding the tokens on their own.”
  - Example: I own 10% of the pool: 200 DAI (price 1$), 1 ETH (price 200$)
    - 20 DAI, 0.1 ETH ($20) → $40
  - ETH price goes up 300$, hodler:
    - 20 DAI, 0.1 ETH ($30) → $50
  - Pool is at 200/1, other pools are at 300/1 → arbitration

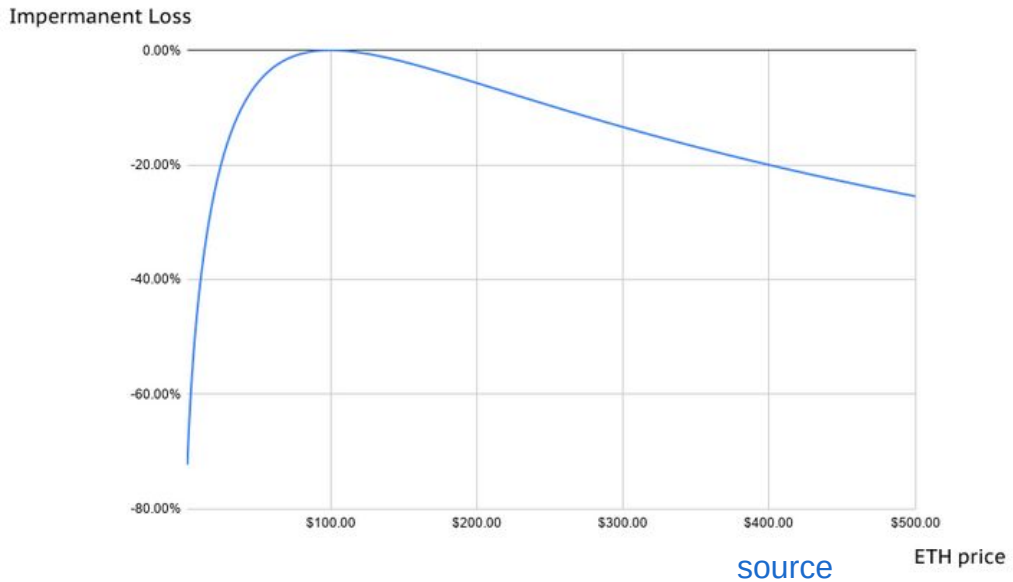
$$y = \frac{Yx}{X + x}$$



OST

Liquidity Providing

- Uniswap formula: ~245DAI/0.8166
  - 10% of 245 DAI → $24.5
  - 10% of 0.8166 ETH → 0.08166 ETH ($24.5)
- $49 vs $50 - 1$ loss instead hodling
  - The more volatile the market is the higher the impermanent loss



source

- LP Token: fees + impermanent loss
  - Other incentive staking: if you place your token in a staking contract
  - Staking != staking
    - Staking on the blockchain layer: proof of funds to mine blocks (ETH 2, Cardano)
    - Staking in contracts: remove liquidity (supply) from the market to influence the price, as a reward, get more tokens (any tokens, e.g., governance tokens – token economics)
- Liquidity mining = yield farming
  - A protocol chooses the best LP with highest APY/APR automatically (e.g., yearn)

OST

Blockchains in Research



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

ToC

- Research at CSG
  - General overview and BC focus
- Blockchain Signaling System (BloSS): Enabling a Cooperative and Multi-domain DDoS Defense
  - Bruno Rodrigues
- Blockchain Selection
  - Eder Scheid
- FlatFeeStack
  - Bruno Rodrigues

Research and Especially Blockchain Research at CSG

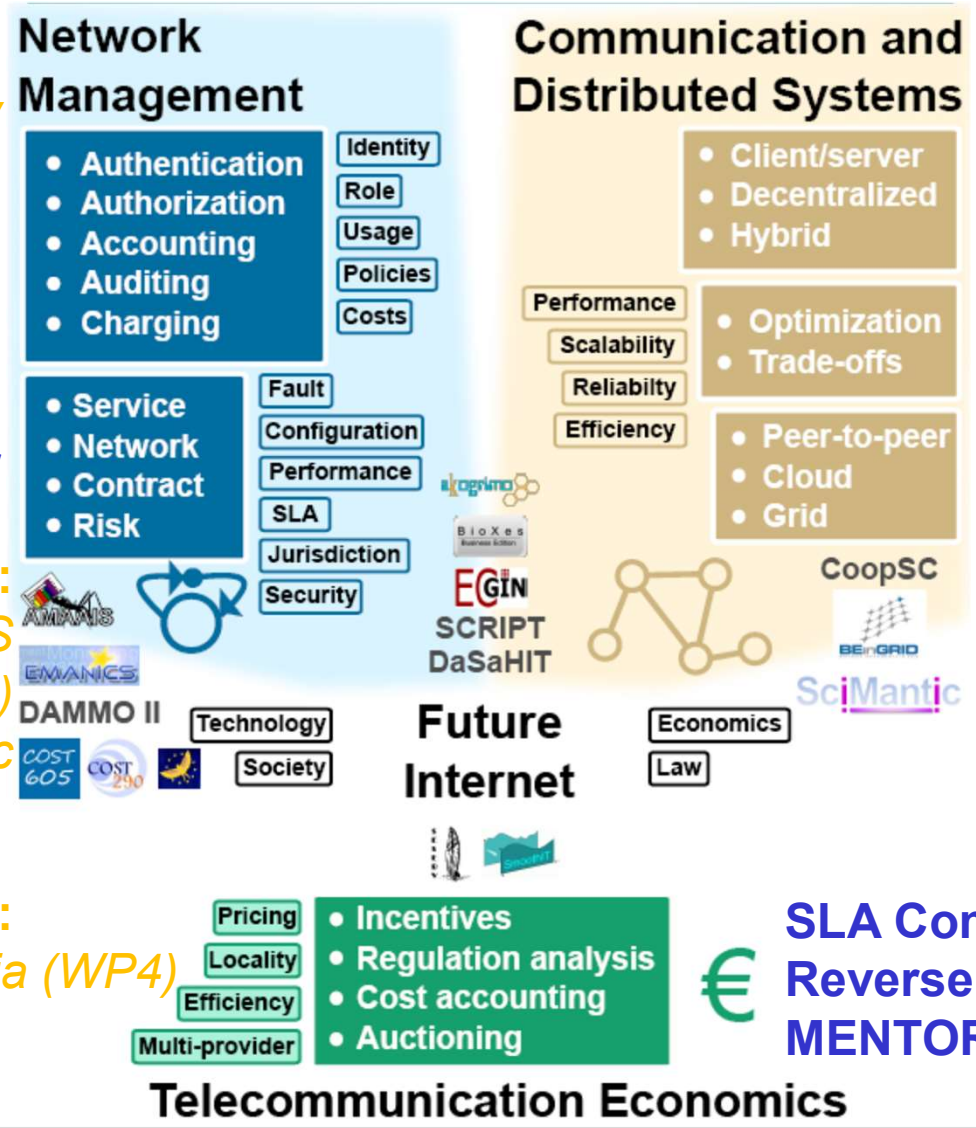


Universität Zürich[UZH]

CSG
Communication Systems Group

CSG's Areas of Work (Past to Present)

Dark yellow topics:
current work as of today



DDoS Mitigation
IoT Security
Cybersecurity Mgt.
Device Behavior

Projects:
BloSS
Concordia (WP1+3)
CyberSenseIT/CyberSpec

Blockchains
Cryptocurrencies
REV, eVoting
Passive Measurements

Projects:
Concordia (WP1)
eVoting
PasWITS
CheeseChains

Projects:
Concordia (WP4)

SLA Compensation
Reverse Auctions (NFV)
MENTOR, SerVIZ, SecBot

Blue topics:
work as from 2018/2019

CSG@UZH Blockchain Research (1)

- Coinblesk – 1st real-time Bitcoin payment Android app
  - Based on NFC communications and multi-signatures (2014-2016)
  - Experiments with UZH Mensa at Binzmühlestrasse (November 2014)
  - Exhibition at Hannover's CeBIT fare (March 2016)
- Blockchains for Coldchains (temperature, IoT) – modum.io
  - CSG's architecture and prototype developed and experimented (since 2015)
    - Zürich's Kickstart Accelerator: 1st Place (November 2016)
  - SME founded + ICO with 13.5 million US$, KYC'ed (September 2017)
- Various "Foodchains" – Tracing and tracking (since 2015)
  - Swiss Federal Office for Agriculture: high quality dairy products tracing
    - Blockchain-based Supply Chain Tracking (SCT) Platform
  - Innosuisse: CheeseChain (since 2021)
    - Blockchain-based Supply Chain Tracking and analytics for Tête-de-Moine cheese and Fromarte

© 2022 UZH, CSG@IfI

CSG@UZH Blockchain Research (2)

- Collaborative DDoS Mitigation Based on Blockchains
  - Decentralized, but collaborative approach (2016-2020)
- EduChain – Blockchains for UZH certificates and diploma
  - General architecture/prototype for “external” verification checks (2017-2019)
- Bazo Blockchain from scratch (since 2017)
  - Proof-of-Stake, mobile light client, blockchain-based loyalty program
- Blockchain-based E-Voting (since 2017)
  - Privacy, verifiability, auditability, secure cast-as-intended
  - DL-based Public Bulletin Board
  - Practically applicable solutions
- Smart Contract-based Applications
  - IoT pollution management (2017)
  - Smart Contract-based Know Your Customer (KYC) Platform (2018-2019)

© 2022 UZH, CSG@IfI

CSG@UZH Blockchain Research (3)

- Blockchains for IoT (since 2018)
  - Blockchain and IoT Integration (Performance, Functionality Experiments)
  - ITrade: A Blockchain-based, self-sovereign, and scalable marketplace for IoT data streams (since 2020)
- Studies on “Off-chain Data Storage Tools”, “Identity Management”
  - Steady collaborations with startups
    - modum.io, ScienceMatters, ICOnator, Axelra, AxLabs

Blockchain Signaling System (BloSS): Enabling a Cooperative and Multi-domain DDoS Defense

Bruno Rodrigues, Burkhard Stiller
and others



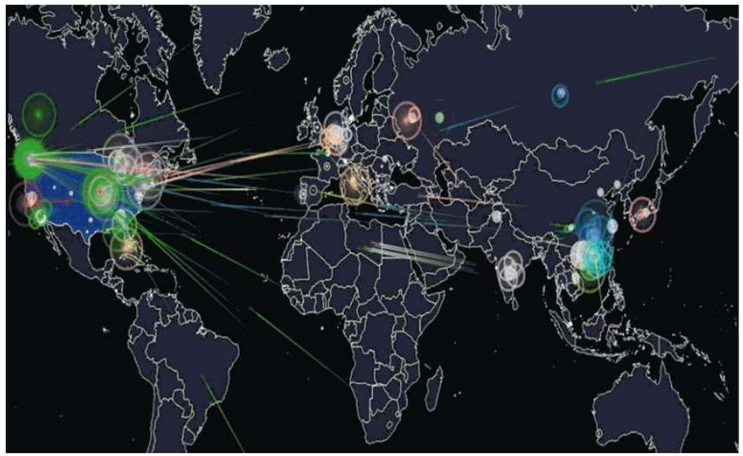
Universität Zürich[UZH]

CSG
Communication Systems Group

Distributed Denial-of-Service (DDoS)

- DDoS defense systems are easily overloaded
  - Simple attacks, but massive traffic
  - Attacks are becoming larger and more coordinated
  - No effective countermeasures
  - Increasing hardware/software defense at a single-domain is not a solution

DynDNS attack (October 2016)



- DDoS defense systems likewise require coordination and distribution
  - Reduce the burden of detection/mitigation in a single domain
  - Extend mitigation capabilities
  - Filter malicious traffic near source

Challenges for a Cross-organizational Defense

- Technical
  - Heterogeneous networking systems
    - Need for a common platform/standard protocols
  - Agility, complexity
  - Bulk data exchange (IP addresses, reports, traces)



- Economic
  - Incentives to share the infrastructure
  - Accountability



- Social
  - Trust/reputation
  - Global/regional coordination
  - Public image



- Compliance
  - Laws, network neutrality



© 2022 UZH, CSG@IfI

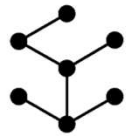
Blockchain Signaling System



- Private Blockchain of **trusted** collaborative domains



- Smart Contracts define **agreements** between domains



- BloSS is implemented as a **decentralized** App on an SDN-managed network.

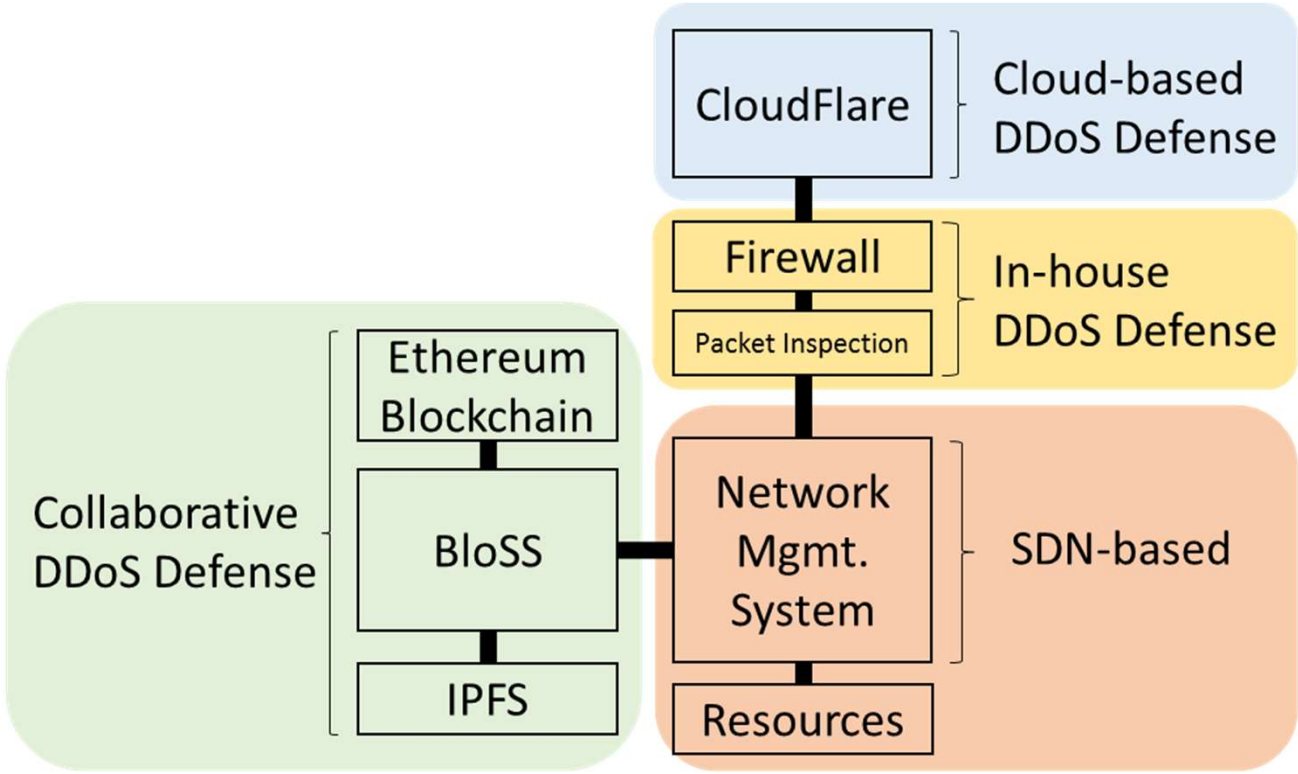


- BloSS makes use of **public** and **already available** infrastructure without the need to support specialized registries



- BloSS can be **combined** with different DDoS protection systems

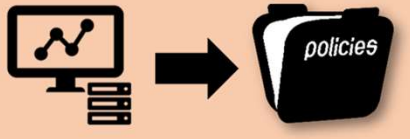
Architecture



© 2022 UZH, CSG@IfI

Stages

Configuration



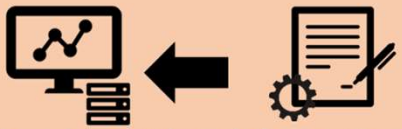
BloSS local configuration and domain-specific policies

- Polling intervals
- Thresholds
  - Minimal number of addresses to accept blocking request
  - Minimal number of addresses to request blocking action
- Available capabilities



Domains declare their managed networks to the Smart Contract

- **AS 100** responsible for: 192.168.10.X/24
- **AS 200** responsible for: 192.168.20.X/24
- **AS 300** responsible for: 192.168.30.X/24



Domains retrieve information about networks managed by other domains

- **AS 100:**
  - **AS 200:** 192.168.20.X/24
  - **AS 300:** 192.168.30.X/24

Operation



When an attack is detected, in-house and collaborative DDoS mitigation occurs in parallel

- **In-house** and **cloud-based** DDoS filtering can be **combined**
- List of addresses are stored at an **IPFS** directory



Update list of addresses to be blocked and output at the IPFS (InterPlanetary File System) directory

- Header declares the **requester IP address** and a **desired action**
- **Hash** pointing to the IPFS object is created



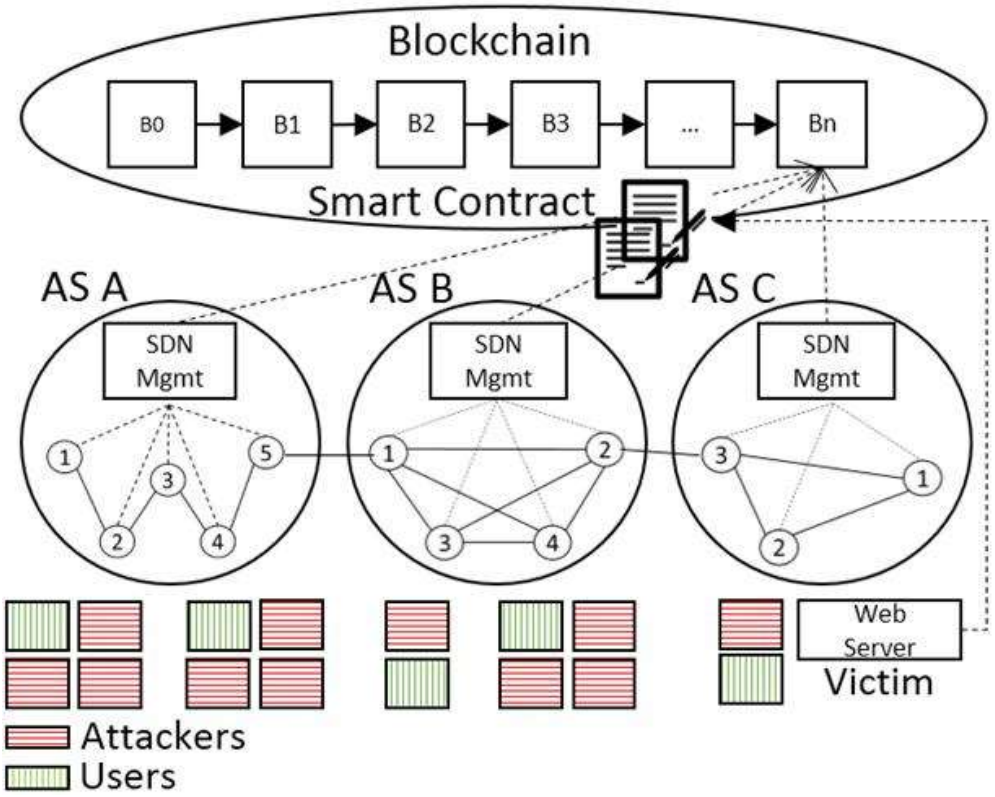
Submit a transaction to the SC containing the hash pointing to the addresses list

- Block is **broadcast** through the Blockchain
- Upon retrieval, other domains may **perform a mitigation or ignore the request**

© 2022 UZH, CSG@IfI

First Prototype (2017-1)

- SDN emulated on Mininet and Blockchain
- Ryu controller + Ethereum Blockchain
- Simple use case to evaluate the approach



Second Prototype (2017-2)

- Three independent Autonomous Systems (ASes 100, 200, 300)
- Two controllers for each domain and four hosts
- Compute: ASUS Tinker Board (Gb Ethernet)
- Network
  - Zodiac FX OpenFlow switches for inter-domain connection (Fast Ethernet)
  - MikroTik routers for intra-domain (Gigabit Ethernet)



© 2022 UZH, CSG@IfI

Prototype Improvements (2018-1)

- PoW → PoA Blockchain
  - No need for mining laptops
- Rockboards 64 for InfluxDB and Graphana
- Modular BloSS with a Rest API

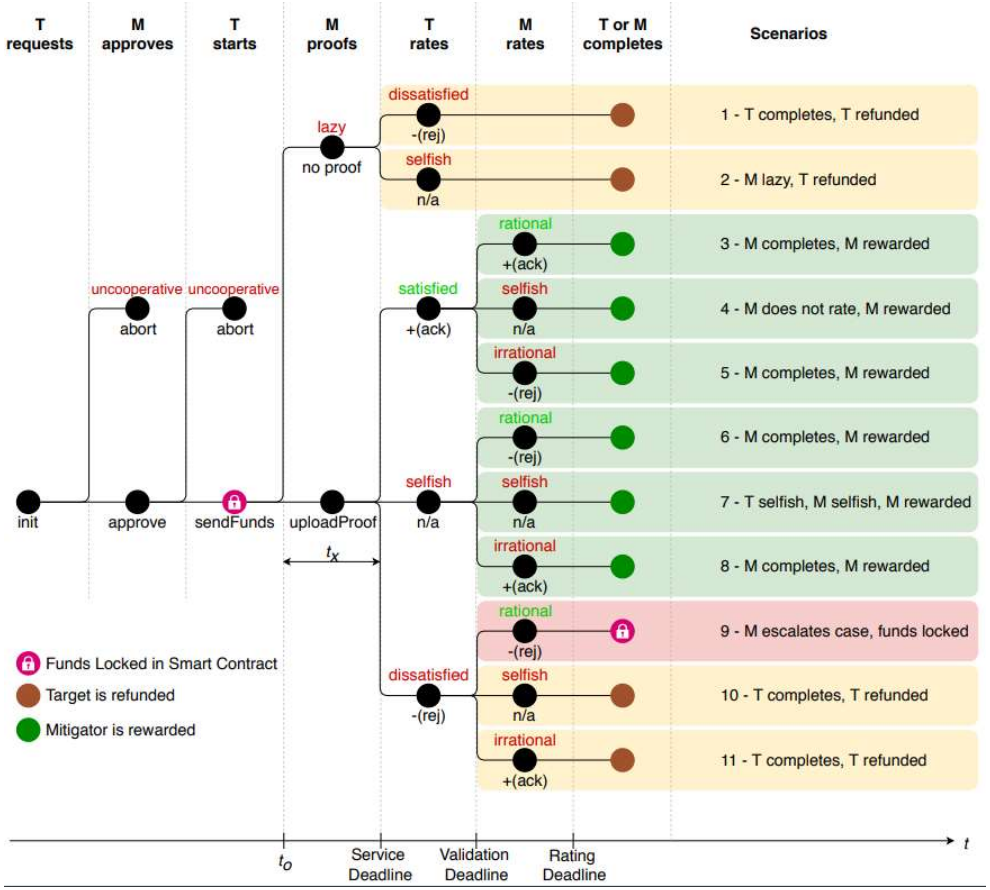


API: Application Programming Interface

Third Prototype (2019-2020)

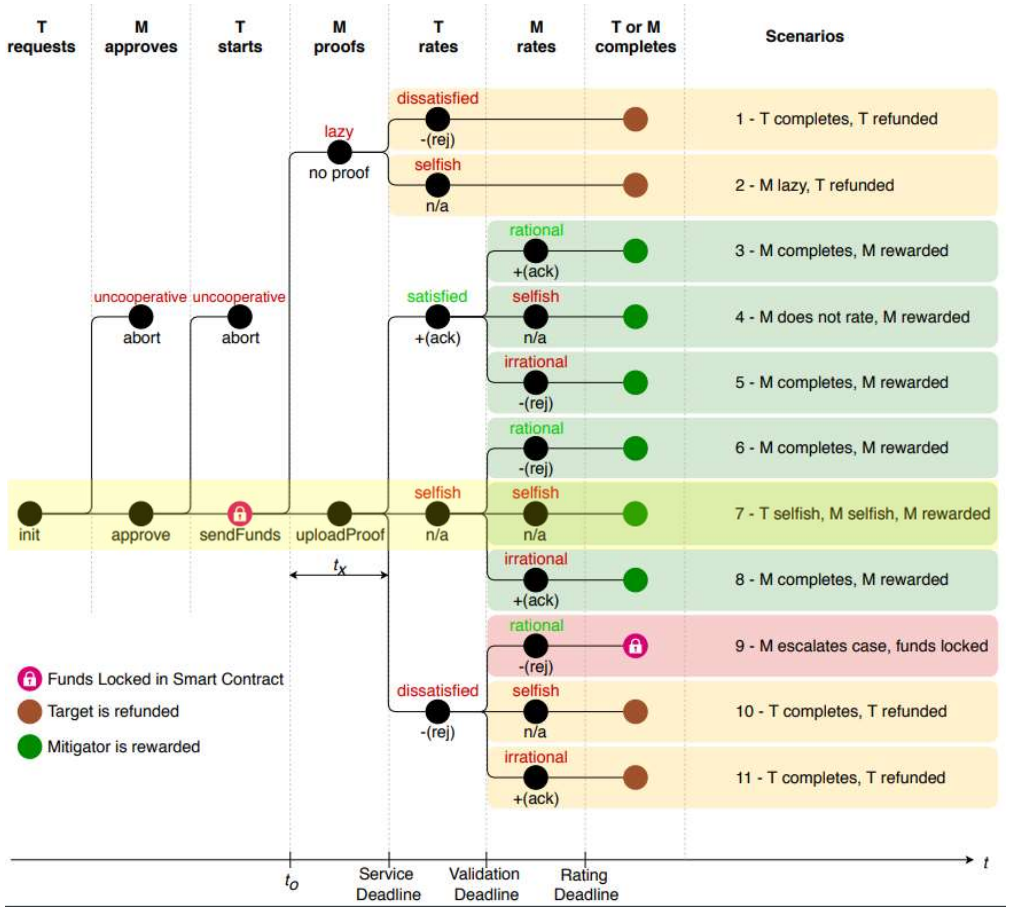
- Goal: to verify end-to-end latency to complete scenarios
- 11 Scenarios (different ending states of the protocol)
- 4 instances
  - Zürich (UZH)
  - São Paulo (AWS)
  - Ohio (AWS)
  - Tokyo (AWS)





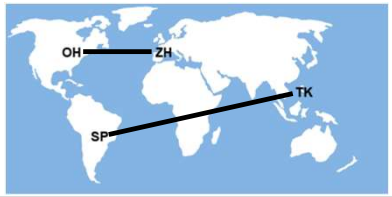
Evaluations (2019-2020)

Scenario	Average Processing Time [s]	Standard Deviation [s]
1	88.938	3.025
2	88.616	1.099
3	87.973	2.110
4	104.090	0.509
5	88.006	1.361
6	104.358	1.340
Worst-case 7	118.900	0.442
8	103.932	0.458
9	89.265	0.711
10	103.331	1.038
11	88.956	0.509
Average	96.950	1.146



Zürich-Ohio

"The average attack duration in Q4 2018 was 218 minutes, the longest one 16 days or 329 hours"



Number of runs per scenario: 20
Block time: 15 seconds

© 2022 UZH, CSG@IfI

Blockchain Selection

Eder Scheid, Burkhard Stiller



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2022 UZH, CSG@IfI

ifi

Myriad of BC Implementations

Bitcoin (2009)
- UTXO-based
- PoW - 10 min block
- 83 Byte
- 240 GB
- Payment

Ethereum (2013)
- Account-based
- PoW/PoS - 14 s block
- +4600 Byte
- 3000 GB
- Smart Contracts

Stellar (2014)
- Account-based
- SCP - 5 s block
- 28 Byte
- 675 GB
- Payment

Non-exhaustive list

Zcash (2016)
- UTXO-based
- PoW - 2.5 min block
- 512 Byte (Encrypted)
- 24 GB
- Privacy-focused

IOTA (2016)
- UTXO-based
- Tangle - 1 min block
- 1300 Byte
- 300 GB
- IoT-focused

EOS (2018)
- UTXO-based
- dPoS - 500 ms block
- 256 Byte
- 4000 GB
- Smart Contracts

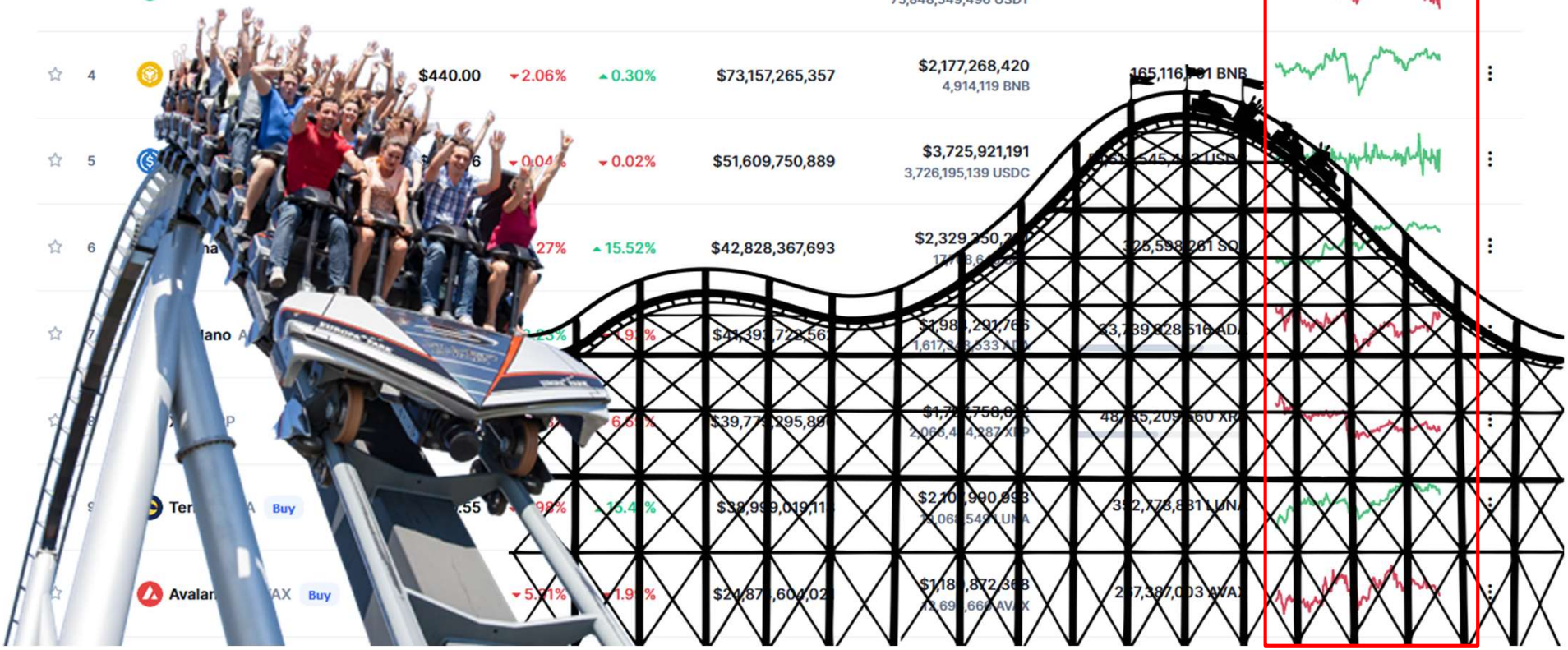
BC technical differences → No *one-size-fits-all* solution

© 2022 UZH, CSG@IfI

Cryptocurrencies Volatility

As of April 4, 2022

#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC Buy	$45,932.25	-1.36%	-3.51%	$872,828,942,109	$30,954,780,021 673,923 BTC	19,002,531 BTC	
2	Ethereum ETH Buy	$3,434.88	-2.21%	+1.56%	$414,876,964,813	$17,166,723,377 4,975,310 ETH	120,240,864 ETH	
3	Tether USDT	$1.00	-0.01%	-0.02%	$82,350,514,562	$75,865,670,080 75,848,549,496 USDT	82,331,930,545 USDT	
4		$440.00	-2.06%	+0.30%	$73,157,265,357	$2,177,268,420 4,914,119 BNB	165,116,[illegible] BNB	
5			-0.04%	-0.02%	$51,609,750,889	$3,725,921,191 3,726,195,139 USDC	[illegible]	
6			-27%	+15.52%	$42,828,367,693	$2,329,350,2[illegible]	[illegible]	
7			-1.23%	-1.9%	$41,39[illegible]	$1,98[illegible]	[illegible] ADA	
8			-6.6%		$39,77[illegible]	$1,7[illegible]	[illegible] XRP	
9	Ter[illegible] Buy		-98%	+15.4%	$38,9[illegible]	$2,10[illegible]	[illegible] LUNA	
10	Avalan[illegible] Buy		-5.21%	-1.9%	$24,87[illegible]	$1,18[illegible]	[illegible] AVAX	



Applicability Scenario

- Companies require different BC implementations to address their use-case
- How to deal with cryptocurrencies volatility?
- Cost reduction without loss of desired properties
  - *E.g.*, immutability, availability, and data integrity

- IoT-enabled Cold Chain → **M09 on Interoperability**

Internet-of-Things (IoT)

  - Devices monitor the temperature of drugs during transport
  - Constant monitoring to ensure compliance → Low prices and high transaction throughput
  - Sensitive data → Private BC or encryption

Identified Challenges



Different types of data and applications require different properties of BCs



Cryptocurrencies prices are hard to estimate and are highly volatile



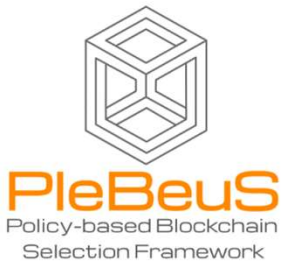
Complexity of data management in multiple BCs

Icons by *Becris* from https://www.flaticon.com

Research Question

How to select the best BC suitable,
in terms of number of fulfilled requirements,
based on user-driven input
considering dynamic environments
and a variety of BCs?

PleBeuS: a Policy-based Blockchain Selection Framework

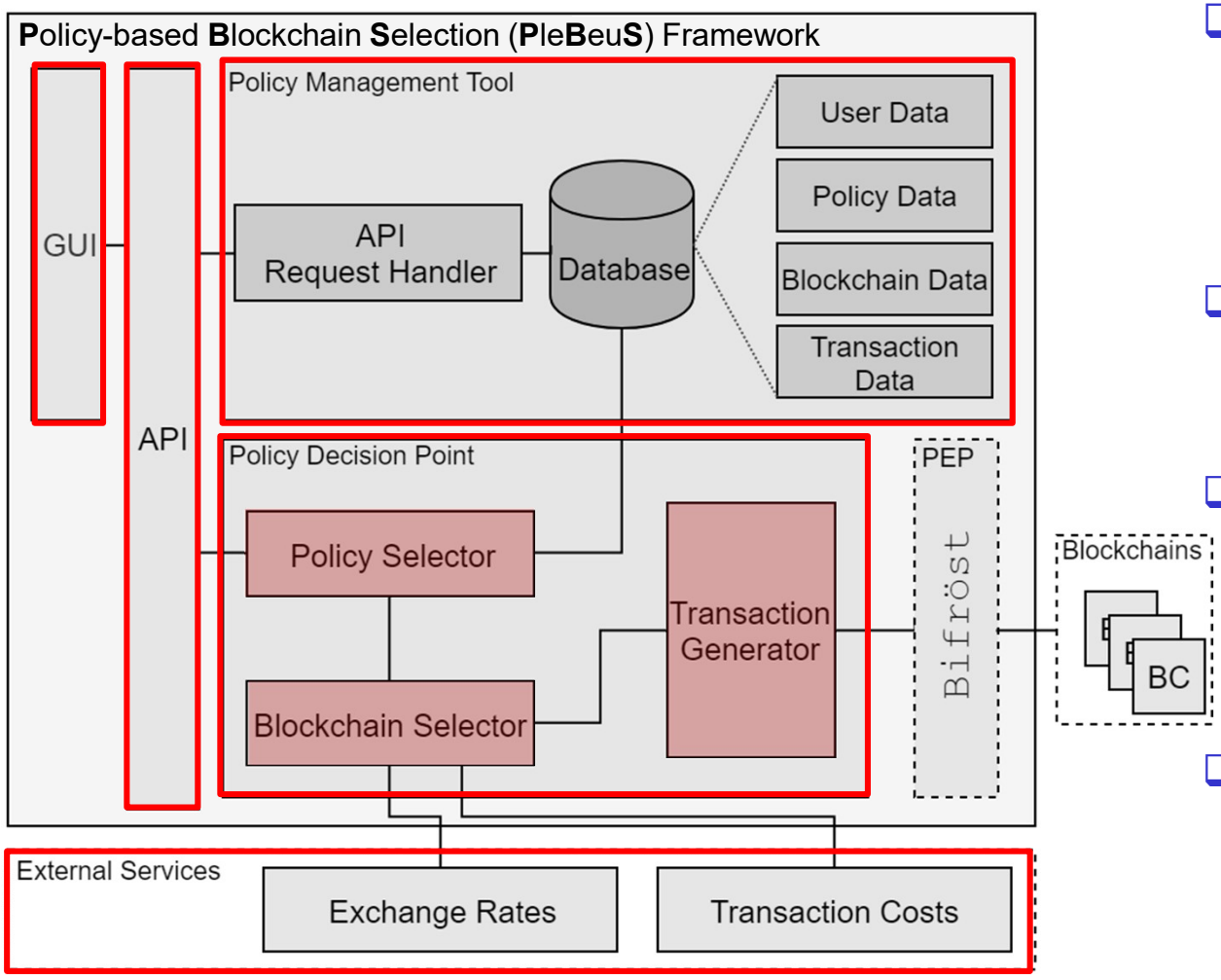


- Given a set of available BCs
  - Supported by the interoperability solution
- BC-specific filter applied
  - Static parameters, *e.g.,* type, data supported, and tps
  - Dynamic parameters, *e.g.,* pricing and fees

Transactions per Second (tps)

- Two selection algorithms
  - Prioritizing economy or performance
- Policy switching
  - Based on time-frame or costs
- Policy conflicts
  - Only one policy active and defined at a given time-frame

PleBeuS Architecture



- Databases
  - User and Policy
  - Transactions and BC
- Policy Selector
  - Trigger and Switching
- Blockchain Selector
  - BC Filters
  - Selection Algorithms
- External Services
  - Exchange rates
  - Transaction Costs

Policy Parameters

- BC-specific Parameters
  - **BC type**: Public or Private
  - **BC throughput**: transactions per second (tps)
  - **Data size**: minimum amount (bytes) of non-related data
  - **Turing-completeness**: BC support for Turing-complete SCs
- Externally-driven Parameters
  - **Costs threshold:** trigger a policy switch
  - **Cost interval**: daily, weekly, monthly, yearly, and default
  - **Cost profile**: performance or economic
  - **Transaction split**: BC round-robin scheme
  - **Time Frame**: period that policy is active (*e.g.,* 08:00 to 18:00)
    (used for detecting policy conflicts)

BC Selection Algorithms

- Filters are applied → yielding a BC set
- Two selection algorithms
  - Prioritizing the performance (tps)
  - Lowest current cryptocurrency cost (single Tx with data)

Transaction (Tx)

Algorithm 1 Fastest BC Selection

```
1:  procedure SELECTFASTESTBC(bcSet, costs)
2:      fastest ← bcSet[0]
3:      for all bc ∈ bcSet do
4:          if (bc.tps > fastest.tps) then
5:              fastest ← bc
6:          else
7:              if (bc.tps == fastest.tps) and (costs[bc] <
    costs[fastest]) then
8:                  fastest ← bc
9:              end if
10:         end if
11:     end for
12:     return fastest
13: end procedure
```

Algorithm 2 Most Economic BC Selection

```
1:  procedure SELECTECONOMICBC(bcSet, costs)
2:      economic ← bcSet[0]
3:      for all bc ∈ bcSet do
4:          if (costs[bc] < costs[economic]) then
5:              economic ← bc
6:          else
7:              if (costs[bc] == costs[economic]) and
    (bc.tps > economic.tps) then
8:                  economic ← bc
9:              end if
10:         end if
11:     end for
12:     return economic
13: end procedure
```

Discussion

- Policies defined

No.	BC Set	Interval	BC Type	Cost [CHF]	Profile
1	All public BC	daily	public	2	performance
2	BTC, ETH, EOS	daily	public	10	economic
3	EOS, IOTA, XLM	weekly	public	15	performance
4	BTC, EOS	weekly	public	30	economic
5	ETH, EOS, IOTA	monthly	public	40	performance
6	BTC, EOS, XLM	monthly	public	60	economic
7	ETH, EOS	yearly	public	80	performance
8	All public BC	yearly	public	100	economic
9	ETH	default	-	-	-

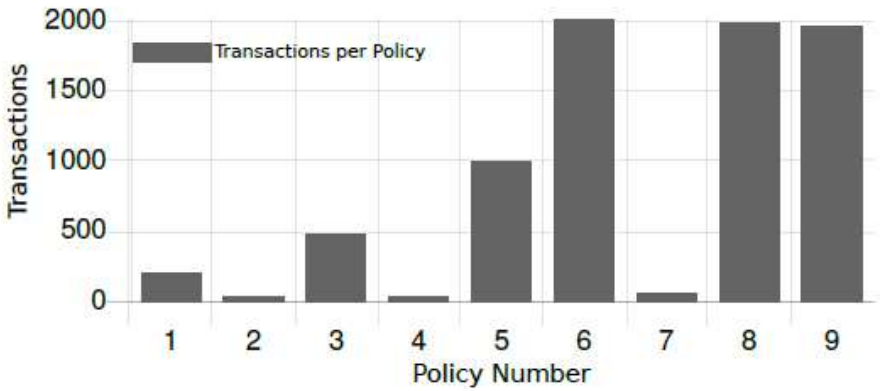
BTC never selected → Slower and costly

ETH selected for default policy only

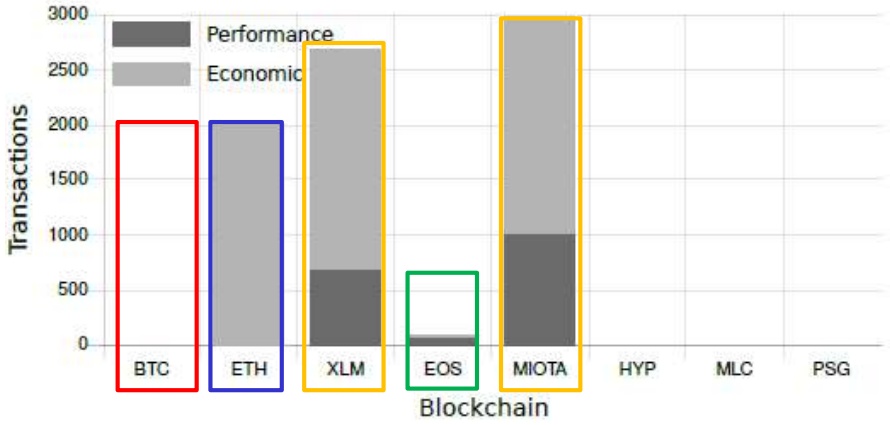
Stellar and IOTA comparable in speed and costs

EOS selected when compared to BTC and ETH

- Results



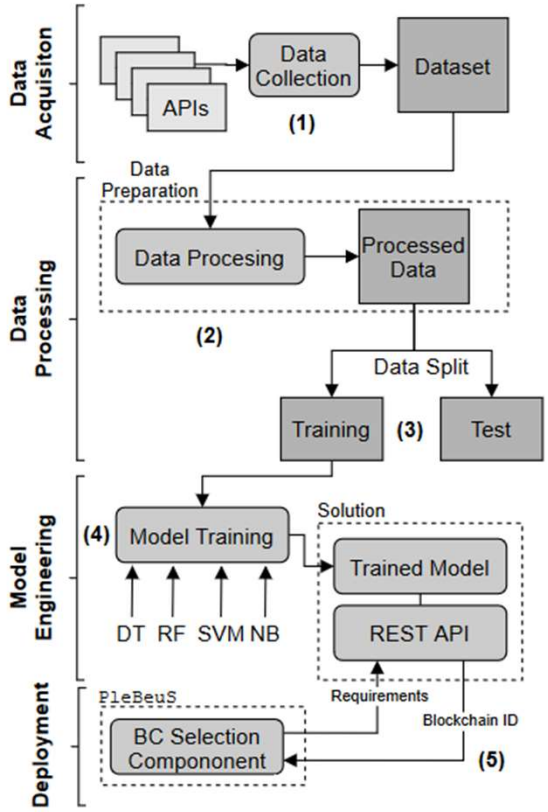
Total: 10,000 transactions

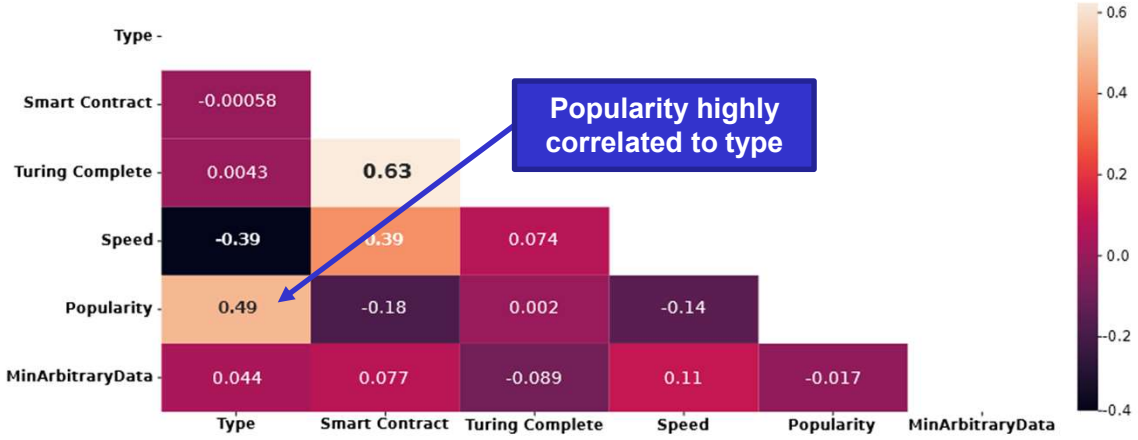


© 2022 UZH, CSG@IfI

Machine Learning-based Selection

- Employment of Machine Learning (ML) algorithms
  - Preliminary research conducted
  - Decision Trees. Support Vector Machines, and Naïve Bayes





ML Model	Accuracy Score	Performance [s]
Decision Tree	86%	
Random Forest	88%	
Naive Bayes	69%	
Support Vector Machine	88%	48.710479

High accuracy in three models

Ratanak Hy: Employing Machine Learning in the Policy-based Blockchain Selection Process; June 2021, URL: https://files.ifi.uzh.ch/CSG/staff/scheid/extern/theses/MA-R-Hy.pdf

Summary and Further Research

Policy-based Management (PBM)

- PBM concepts employed for BC selection
  - Based on user requirements (policies)
- Automatic BC selection
  - Management complexity reduced
- Two selection algorithms
  - Cost and performance-aware
- ML-based selection
  - Feasible and flexible
  - Research is still needed
- Further Research → **BA / MA Theses**
  - Definition of **subjective** BC parameters
    - Popularity, code maturity, BC stability, price history, and security

For example, **popularity** formula:

$$Score = N_{follow} \times W_1 + G_{search} \times W_2 + C_{papers} \times W_3$$

FlatFeeStack

Bruno Rodrigues, Eder Scheid,
Thomas Bocek, Guilherme Machado,
Burkhard Stiller



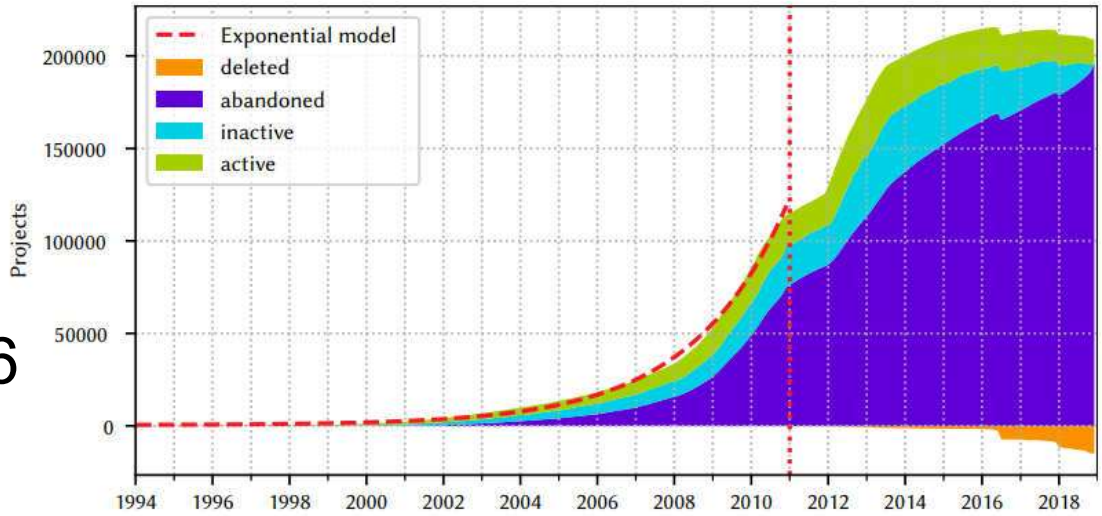
Universität Zürich[UZH]

CSG
Communication Systems Group

Landscape of Open Source Projects (1)

a) Number of projects

- Until July 2013, available Open Source projects grew exponentially, reaching a plateau in 2016
- The vast majority of Open Source projects are abandoned
  - Never received a contribution again until the end of measurement or their deletion



- There is a significant decrease of new projects in the year 2011

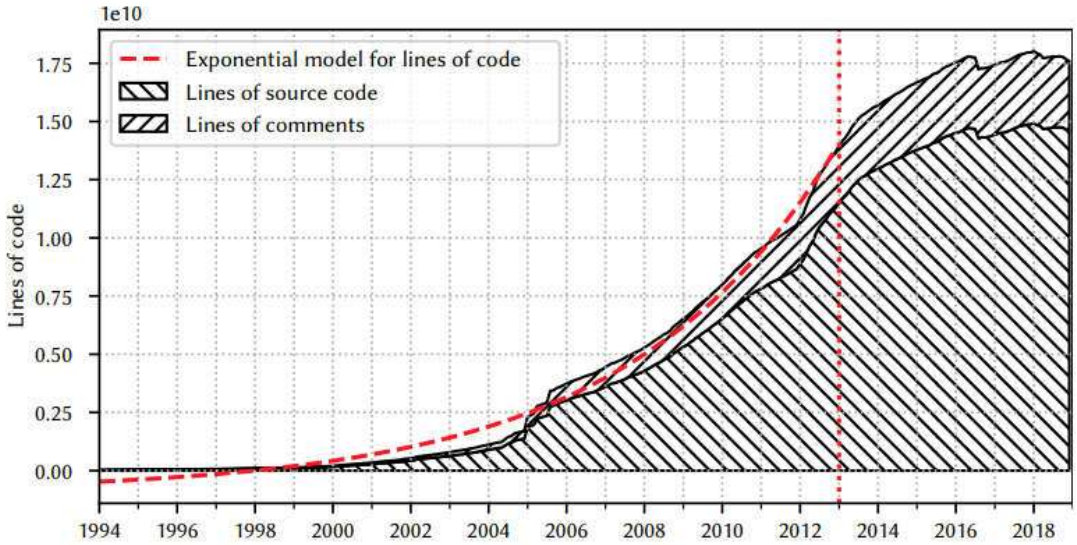
Dorner, Michael, Maximilian Capraro, and Ann Barcomb. "*Quo Vadis, Open Source? The Limits of Open Source Growth.*" *arXiv preprint arXiv:2008.07753,* 2020

Landscape of Open Source Projects (2)

b) Lines of code vs lines of comments

- Although initially growing exponentially through 2010, the growth in lines of code has continuously slowed down since 2011

The relative increase of comment lines may correlate with the popularity of programming languages and tools, which generate documentation from source code (e.g. JavaDoc for Java, Pydoc for Python, or Godoc for Go)

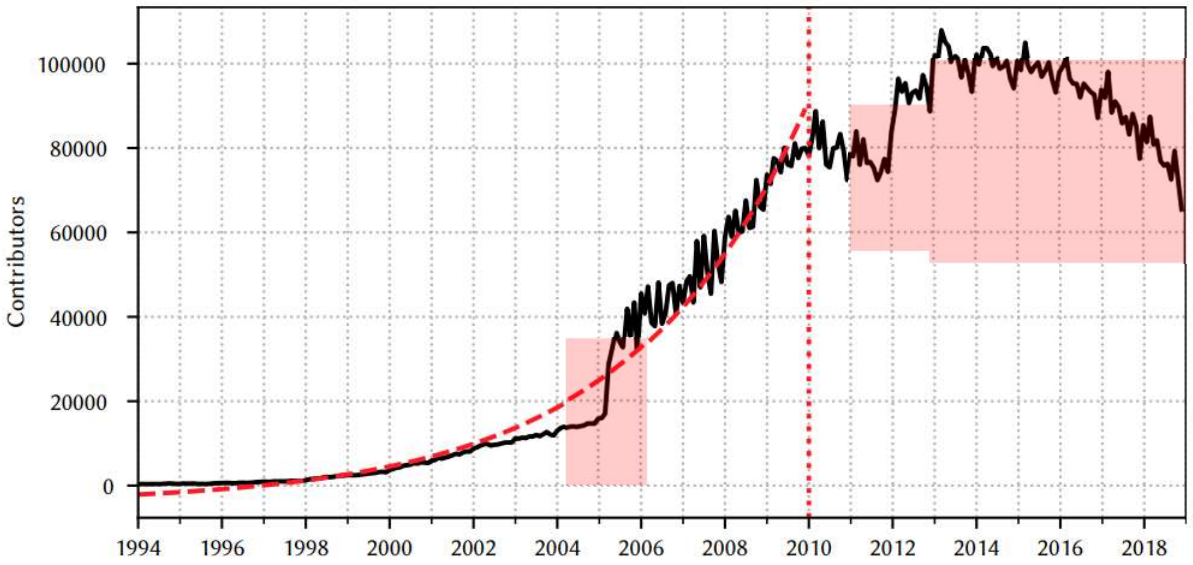


Dorner, Michael, Maximilian Capraro, and Ann Barcomb. "*Quo Vadis, Open Source? The Limits of Open Source Growth.*" *arXiv preprint arXiv:2008.07753,* 2020

Landscape of Open Source Projects (3)

c) Number of contributors

- **Erratic** rise of contributors in 2005, and again in 2012
  - Possibly error in the methodology of measurement due to Bot activities
- Since 2015 the general trend has been a decline



Dorner, Michael, Maximilian Capraro, and Ann Barcomb. "*Quo Vadis, Open Source? The Limits of Open Source Growth.*" *arXiv preprint arXiv:2008.07753,* 2020

Reasons for the Decline

- The major question
  - Why open-source projects not only stopped growing exponentially, but actually declined?
- Lack of contributor's time
  - *The shift from volunteer to paid contributions reducing the effective time for contributing for each participant, due to company resource management*
- Lack of contributor's interest
  - *A decrease in developers willing to volunteer, and no corresponding increase in paid development work*
- Project becomes obsolete
    - Based on outdated OR complex technologies
      - *Increasing code complexity requiring skills fewer developers possess, and discouraging newcomers*
      - *A decreased quality of contributions and, therefore, a lower acceptance rate of contributions and an overload for reviewers and committers*
- Lack of funding
  - Lack of motivation? Excessive number of projects?

Source: *Jailton Coelho and Marco Tulio Valente. 2017. Why Modern Open Source Projects Fail. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017). Association for Computing Machinery, New York, NY, USA, 186–196. DOI:https://doi.org/10.1145/3106237.3106246*

Open Source Incentives (1)



- Basic premise
  - Open collaboration to produce software and services without **express** expectation of return
- Two categories of incentives (which are not strictly economic)
  - Individual (*e.g.*, person that uses Open Source software and wants to contribute)
    - Highlight in the community, altruistic reasons to make the world a better place, small items (t-shirt, stickers), and **monetary** incentives
  - Corporate (*e.g.*, company that base its products/services on Open Source software)
    - Talent acquisition (or retention), profit by providing services supporting open-source software (*e.g.*, Red Hat)

Open Source Incentives (2)

- Although monetary incentives are not to be expected, they are highly **relevant**
  - Projects are becoming increasingly large and complex
  - Contributions demands time/effort often not available

- Alternative is to use Crowd Sourcing platforms

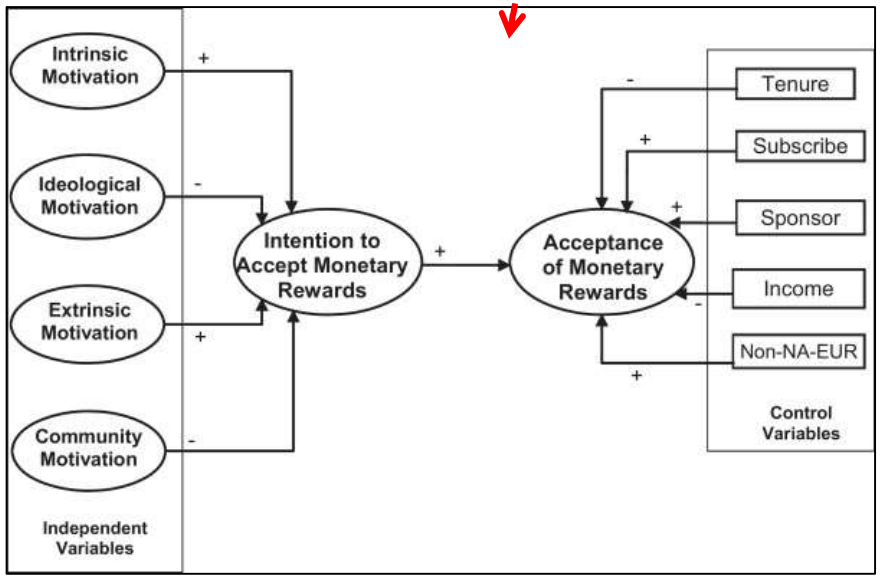
PATREON | catarse

Buy me a coffee Flattr

Support Me on Ko-fi

Donate with LiberaPay

GOTEO www.goteo.org



Source: Krishnamurthy, Sandeep, Shaosong Ou, and Arvind K. Tripathi. "Acceptance of monetary rewards in open source software development." *Research Policy* 43.4 (2014): 632-644.

What About Competitors?

Platform	Open Source	Pay-in Model	Pay-out Method	Crypto payout	Sponsoring Structure	Sponsoring Individuals Proportionally	Fees* Structure	Architectu-re
Patreon	✗	One-off contribution	Direct transfer, Payoneer, PayPal	✗	1 to 1 (individuals)	✗	5%, fixed	Centralized
Buy me a coffee	✗	One-off contribution	Direct transfer, PayPal, Stripe	✗	1 to 1 (individuals)	✗	5%, fixed	Centralized
Flattr	✗	Flat fee subscription	Direct transfer	✗	1 to N (multiple projects)	✗	5%, fixed	Centralized
Ko-Fi	✗	One-off contribution	PayPal, Stripe	✗	1 to 1 (individuals)	✗	0%, 5% gold members	Centralized
Liberapay	✗	One-off contribution	PayPal, Stripe	✗	1 to 1 (individuals)	✗	Up to 5%	Centralized
Goteo	✓	One-off contribution	Direct transfer, PayPal	✗	1 to 1 (individuals)	✗	5%, fixed	Centralized
GitHub Sponsors	✗	One-off contribution	Direct transfer	✗	1 to N (multiple projects)	✗	0%, 10% companies	Centralized
FlatFee-Stack	✓	**Flat fee subscription**	**Stripe, Crypto**	✓	**1 to N (multiple projects)**	✓	**5%, fixed**	**Centralized**

Major competitor

Unique Selling Points

Opportunity

*Does not account additional fees of payment provider

Flatfeestack*

- Mission/Goal
  - Provide a donation platform enabling developers (contributors) to live (financially) based on their contributions to open-source projects

Strong USP / Novelty

Unique Selling Points (USP)

- Incentive mechanism (*i.e.,* contribution engine) based on individual sponsoring based on the relevance of contributions
- Crypto payouts minimizes fees for contributors and allows for reinvesting income
  - Why not decentralized? Would also be a strong USP / Novelty

- Major sponsors (customers):
  - **Companies** relying on open-source software

Easier based on a decentralized solution

Social impact

- Need a robust strategy to attract sponsors (e.g., ecological/sustainable stamps or certifications exploitable in marketing campaigns

FlatFeeStack's (Current) Design

Frontend
- Interface to register and sponsor projects

Auth
- Identify users and grants access to the API services
- Token-based

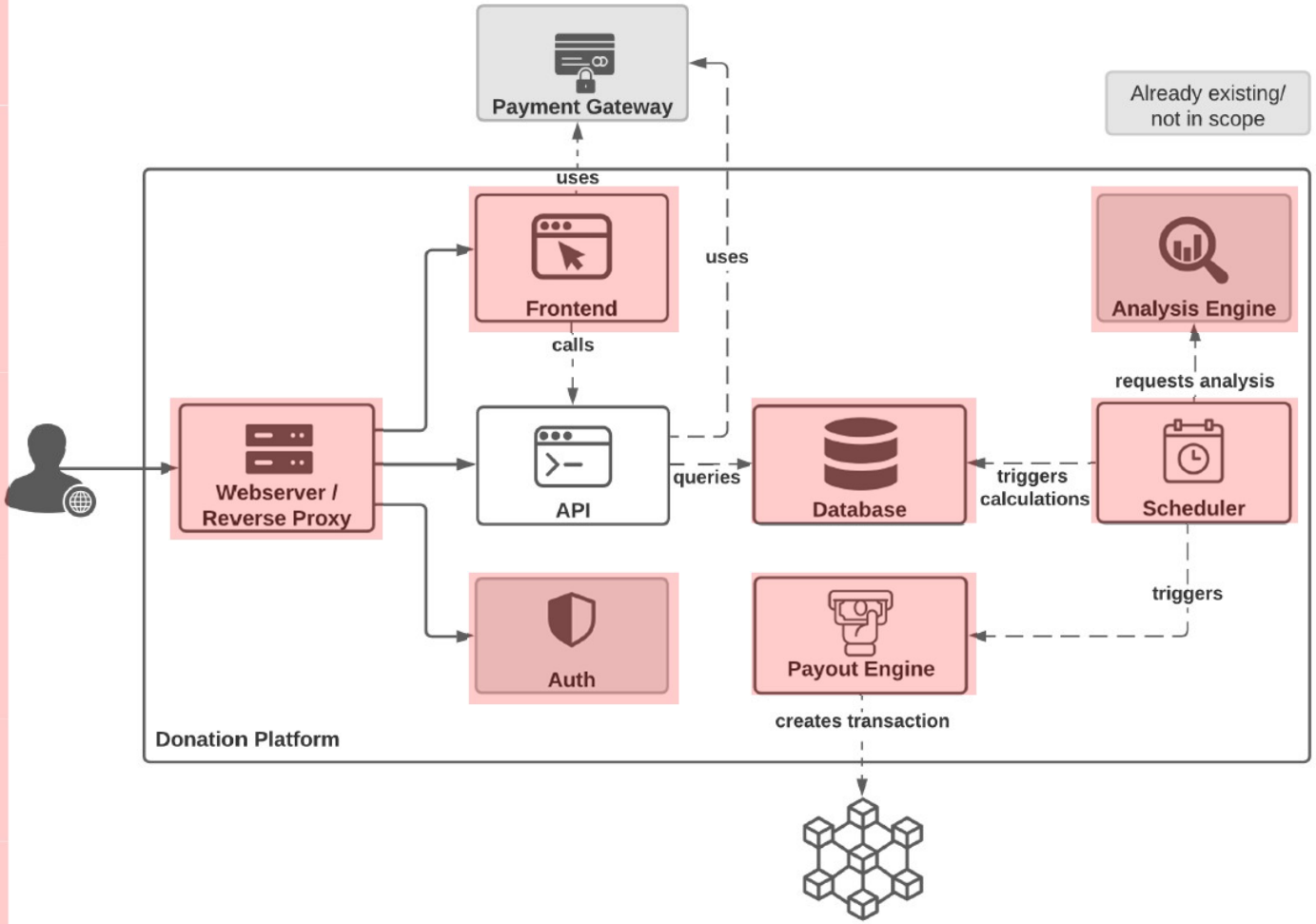
Reverse-proxy
- Entry-point
- Redirects to: frontend, auth, and API

Database
- Stores user and project-related data.
- Intermediate calculations such as daily balances and data related to the payout.

Analysis Engine
- Analysis of individual contributions
- Based on the commits of sponsored projects

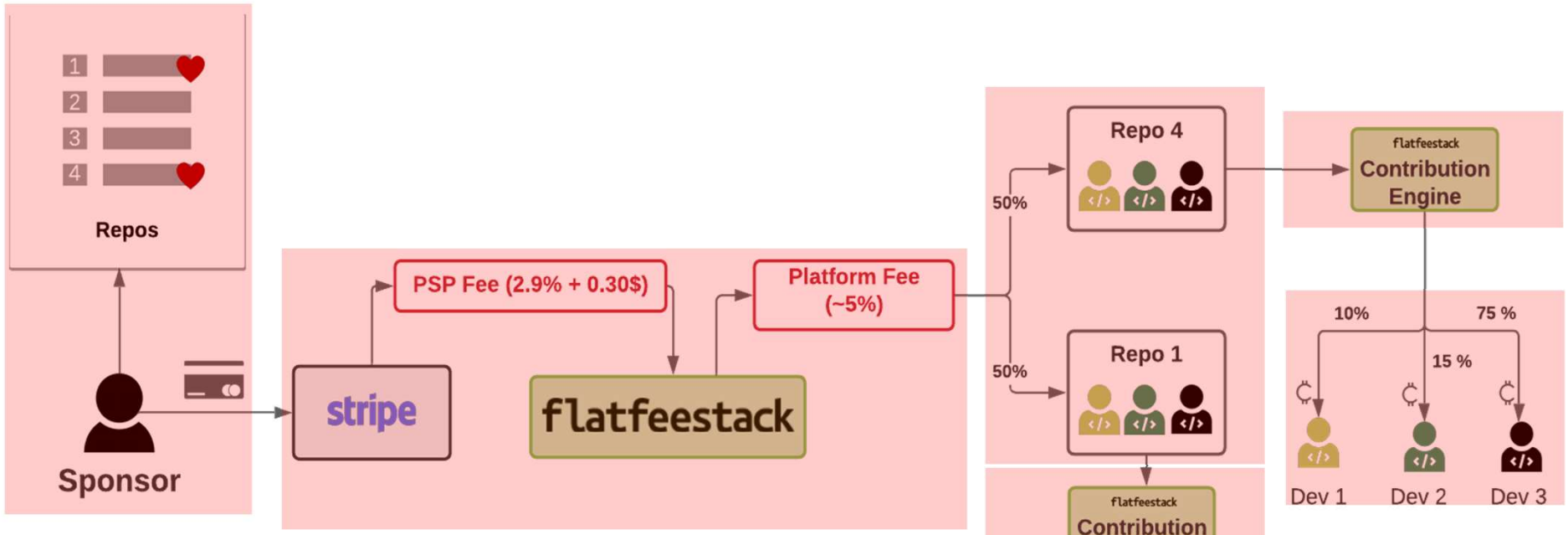
Scheduler
- Trigger repeating tasks (e.g., payouts).

Payout Engine
- Connects to a smart contract to provide



By Jonas Brunner, Eder Scheid, Guilherme Machado, Thomas Bocek

Sponsoring Flow



1. Sponsor

- Selects which projects/repositories to sponsor
- Realizes the Payment

2. FlatFeeStack

- Handles donation
- ☺ ☺ ☺ Charges fee ☺ ☺ ☺

3. Multi-stage Processing of Payout

- [**Split**] Donation is split between selected repositories according to selected percentages
- [**Process individual contributions**] Top contributors receives proportionally according to the relevance of their contribution
- **[Aggregate individual contributions]** for individual contributors participating of different projects

Contribution Engine

- Assumption: git-based repositories
- How to measure the relevance of contributions?
  - Formula weighting different types of contribution
    - Can be implemented on a smart contract
- Who/how to determine weights?
  - As of now, flatfeestack
    - Maybe an issue?
- Who/how to determine code quality?
  - Really subjective metrics
    - Example: clear, simple, refactored, bug-free



Git Driven	Project Driven
Commits	Issues (is)
Merges (mrg)	Issue comments (isc)
Added / deleted Lines of code (loc)	Pull requests
Reverts	Pull request comments
Dangling commits	Code reviews
Squashes	

Calculation of Weights

- Current formula per category:

1.Weight per category

3. Averaging contributions Per metric within a category

$$C_{developer} = \sum_{c=0}^{c_{total}} \omega_c \times \frac{\sum_{m_c=0}^{m_{c,total}} \omega_{m_c} \times \theta_{m_c}}{\sum_{m_c=0}^{m_{c,total}} \omega_{m_c} \times \Theta m_c}$$

2. Example: additions might have higher weight than deletions

2. Each metric has an individual weight

4. Representation is weird because, at first, it is needed to calculate individual contributions.
→ Better to represent in two equations

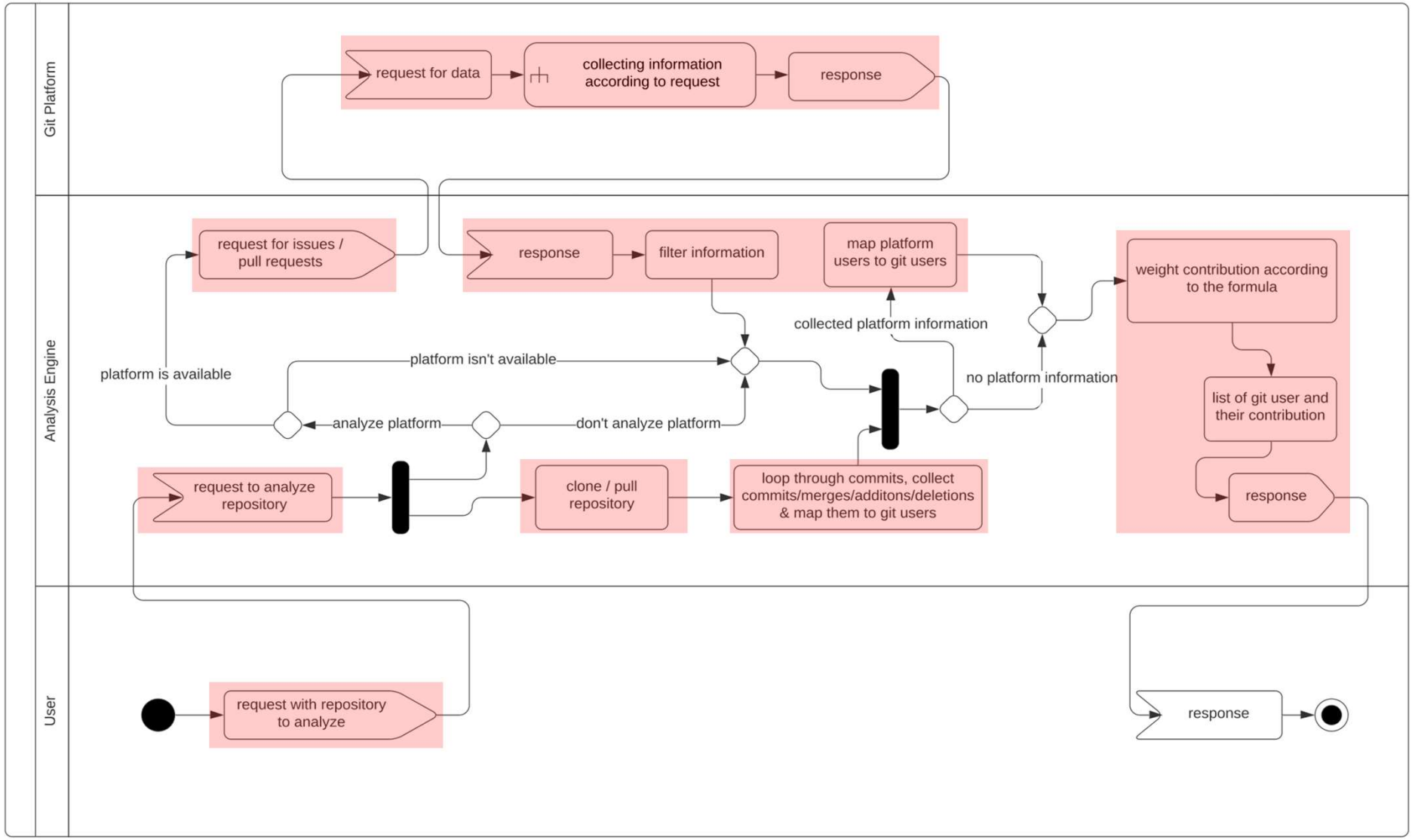
where $C_{developer}$ = Calculated contribution of the developer
c_{total} = Total amount of categories
ω_c = Weight of the category c
c = Index of the category
m_c = Index of the metric inside category c
$m_{c,total}$ = Total amount of metrics within category c
ω_{m_c} = Weight of the metric inside category c
θ_{m_c} = Value of the metric for the analyzed developer
Θ_{m_c} = Total value of the metric for all developers

Categories and Metrics

- Changes
  - Additions
  - Deletions
- Git History
  - Commits
  - Merges
- Issues
  - Author of Issues
  - Comments on Created Issue
  - Written Comments
- Pull Request
  - Author of Pull Request
  - Activity on Created Pull Request
  - Performed Code Reviews

By Calvin Falter, Eder Scheid, Guilherme Machado, Thomas Bocek

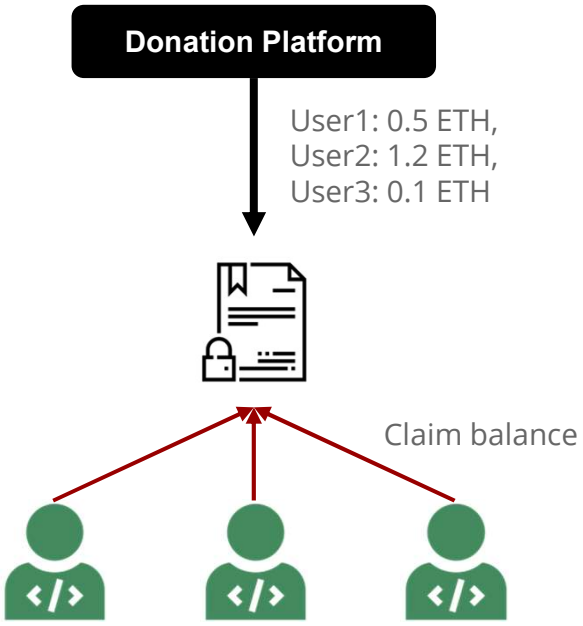
Implementation Details



Crypto Payout

- First prototype based on Ethereum
  - ETH-based solution is very expensive
    - Average transaction fee, USD | 0.01 ETH ($40.47) (as of 05.09)
  - Fees are paid by the platform (cash in) and the contributor (cash out)

- Major requirement
  - Shift to a relatively (to the sponsored amount) low fee blockchain platform
    - Tezos, Cardano (limited SC, limited hype)



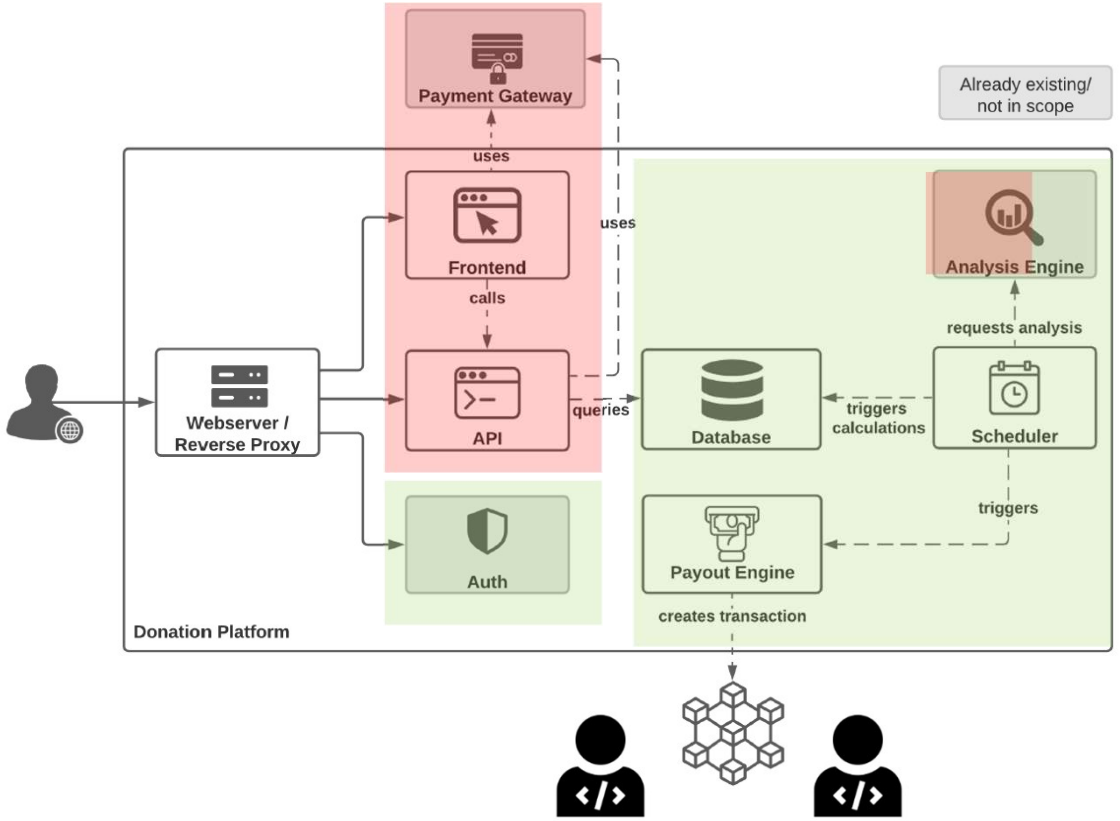
By Jonas Brunner, Eder Scheid, Guilherme Machado, Thomas Bocek

Discussion (1)

- Decentralization
  - Not possible to decentralize all components
  - It would be a huge USP and marketing leverage
- Contribution engine
  - (At least) Community-driven adjustment of parameters
- Crypto payout
  - High fees or volatily could hinder the project
  - Driving question: How to ensure stable income for contributors?

Discussion (2)

- Full decentralization is difficult, if FlatFeeStack acts as an exchange
  - Rely on an exchange API could allow for decentralization
- Logic of the Contribution engine could be **partially** in a Smart Contract
  - Gits/repositories are oracles (trusted by the community)
- Scheduler is not an issue
  - Contributors request the payout

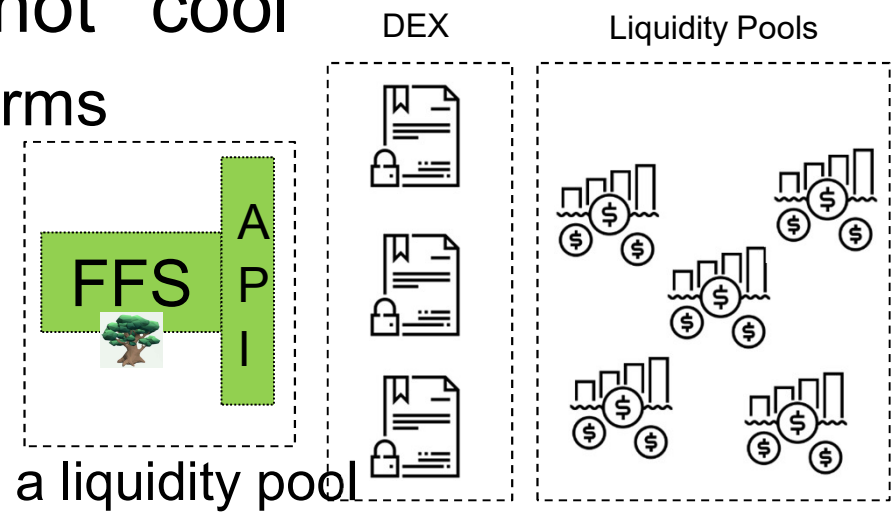


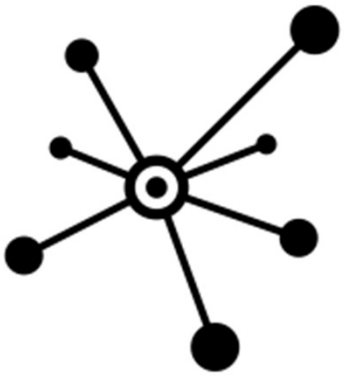
Discussion (3)

- Methodology in the contribution engine could raise concerns
  1. Determine code quality
  2. Associate a price (to be paid) per contribution
- Suggestion: shift the problem back to their respective community/project
  - Quality of contribution should be assessed within the repository by the community
  - Price tag could be based on allocated hours and equally determined by the community
- Related parameters can be adjusted in a smart contract and transparent to everyone

Discussion (4)

- Single currency is “nice” but not “cool”
  - Run away from high-fees platforms
  - Smart contract and low fees
- Opportunity on DeFi
  - Earning interest
    - Staking directly or adding funds to a liquidity pool
  - Lack of liquidity could be an issue
    - Multiple pools
  - LiquidSwap is an interesting alternative
    - High-fees in layer 2 swaps could be a stepping-stone
    - Examples: **Thorchain**, **o3swap**, **Chainflip**, **Algorand**
- Crypto payout only?





Blockchain Research Theses at CSG

CSG Team



Universität Zürich[UZH]

CSG
Communication Systems Group

Open Student Projects/Theses

- https://www.csg.uzh.ch/csg/en/theses.html
- Specifically, Blockchain-related
  - Secure Multi-Party Computation for Cryptocurrency Key Management
  - Design and Implementation of a Blockchain Analytics and Visualization Hub