



University of
Zurich^{UZH}

FCC: Forensic Chain of Custody

Sophy Chhong
Zurich, Switzerland
Student ID: 12-930-194

Supervisor: Bruno Rodrigues, Eder Scheid
Date of Submission: November 2, 2019

Zusammenfassung

In der heutigen Zeit erleben wir einen Überfluss an Informationen und Telekommunikationen. Die entstehenden Daten können im Rahmen eines kriminellen Aktes zu Beweismitteln werden und haben deswegen einen grossen Stellenwert. Die Forensik erwirbt, bewahrt und analysiert diese Daten um diese in einem Gerichtsfall präsentieren zu können. Die digitalen Beweismitteln sind schwierig zu erwerben und handzuhaben, da sie aufgrund ihrer elektronischen Natur auf Verfälschungen anfällig sind. Die Aufgabe des forensischen Mitarbeiters ist es die Integrität während des gesamten forensischen Prozesses aufrecht zu halten. Eine starke Kontrollkette (Chain of Custody) kann die Integrität von digitalen Beweismitteln unterstützen. Die vorliegende Arbeit versucht diese Kontrollkette mit einem Blockchain Service zu verstärken. Die Funktionalitäten wurden in einem Smart Contract verwirklicht, welches nur registrierte Adressen zulässt. In der Blockchain sind lediglich Metadaten abgelegt, welche nicht auf involvierte Personen zurückgeführt werden kann. Die digitalen Beweise nehmen viel Speicher ein und werden deswegen in einer vertraulichen Server Umgebung abgelegt. Eine Fallstudie und das STRIDE Threat Modell evaluiert zum Schluss den Prototypen. Die Untersuchungen zeigen, dass alle Anforderungen zufriedenstellend erfüllt worden sind und dass der Validierungs-Mechanismus die Sabotageversuche sowohl entdecken als auch verhindern kann. Diese Arbeit zeigt, dass die Vollständigkeit, die Echtheit, die Gültigkeit und die Prüffähigkeit mit Blockchain erreicht werden kann.

Abstract

In an era where information and telecommunication keeps expanding, generated data can become important pieces of evidence for an alleged crime. Forensic science collects, preserves and analyzes such data. Their function is crucial as they can make a big difference in a lawsuit. The acquisition of digital evidence differs from physical evidence, due to their fragile electronic nature. As they are vulnerable to tampering, keeping the integrity of digital evidence becomes crucial. Hence, a Chain of Custody (CoC) can provide the proof of integrity, if well maintained. This work aims to improve the chain of custody of an existing workflow by proposing a blockchain service. The functionalities have been implemented in a smart contract, that allows registered accounts to interact with the evidences. Only relevant metadata are inserted to the blockchain so that involved individuals cannot be traced. The actual digital evidence, that takes up a lot of space, is stored in a separate server environment. A use case study along with the STRIDE threat model were conducted to evaluate the prototype. It can be concluded that all requirements have been sufficiently met and that the verification mechanism can detect, and therefore prevent tampering attempts. This work argues that the integrity, authenticity, validity and auditability of evidences can be achieved with a blockchain-based approach.

Acknowledgments

I would like to express my gratitude to my supervisor, Bruno Rodrigues, who continuously guided me throughout this work. His knowledge and inputs helped me progress during the last six months.

I would also like to thank Prof. Dr. Burkhard Stiller, the head of the Communication System Group (CSG) at the University of Zurich, for allowing me to research and write about this topic.

I sincerely like to thank my boss and the colleagues at the company, who allowed me to investigate and ask questions about their workflow. Without their help, none of this would have been possible.

Last but not least, I am thankful to my friends and family for their support and to those, who took the time to proof read this work.

Contents

Zusammenfassung	i
Abstract	iii
Acknowledgments	v
1 Introduction	1
1.1 Motivation	1
1.2 Description of Work	2
1.3 Thesis Outline	2
2 Background and Related Work	3
2.1 Blockchain Technology	3
2.1.1 Types of Blockchain	4
2.1.2 Consensus Mechanisms	5
2.1.3 Smart Contracts	6
2.2 Digital Forensic	6
2.2.1 Digital Evidence	7
2.3 Threat Modeling	7
2.4 Related Work	10

3	Design and Implementation	13
3.1	Workflow	13
3.2	Requirements	14
3.3	Architecture	15
3.4	Implementation	18
3.4.1	Functions	20
4	Evaluation	23
4.1	Use Case Study	23
4.1.1	Action 1: Adding and Removing Accounts	23
4.1.2	Action 2: Registering, Verifying and Deleting an Evidence	24
4.2	Threat Analysis	26
4.2.1	Action 1: Adding and Removing Accounts	28
4.2.2	Action 2: Registering, Verifying and Deleting an Evidence	30
4.2.3	Discussion	31
5	Summary and Future Work	33
5.1	Future Work	34
	Bibliography	35
	Abbreviations	37
	List of Figures	39
	List of Tables	41
A	Installation Guidelines	43
A.1	Setting up Ganache	43
A.2	Install Solidity	43
A.3	Install Truffle	44
A.4	Setting up Prototype	44
B	Contents of the CD	45

Chapter 1

Introduction

1.1 Motivation

As the world experiences a continuous expansion in information and telecommunication processes, a massive amount of data is generated everyday. The acquisition of such data becomes important as they become evidence for possible illegal actions. The increasing complexity of forensics in such digitized era becomes relevant not only due to the great diversity of electronic devices [7], but also due to the increasing complexity of protocols and means of communication. Collecting digital evidences is a complex task due the electronic nature of devices, which are relatively vulnerable to tampering (causal and temporal), and fragile in nature. Thus, it is of important to ensure the integrity of digital evidence during its whole life cycle in any forensic investigation [10]. A digital forensic Chain of Custody (CoC) is defined as a process used to maintain and document the chronological history of handling digital evidence [7, 10], being critical to determine responsibilities of evidences reported in such chain. Indeed, if a CoC is not well maintained, investigators might get accused of deliberate corruption of the investigation.

Many non-law enforcement institutions typically store primary copies of forensically acquired digital evidence in laboratorial environments rather than retaining the original media. To increase the authenticity and audibility of digital evidences, it is proposed to undertake research into and create a service that enables the inclusion of activities to the creation and preservation of digital evidence in blockchain transactions. Blockchain provides a trustworthy, decentralized, and publicly available data storage, making it an interesting opportunity for organizations to increase business agility and reduce costs by removing intermediaries in distributed applications (*i.e.*, by involving multiple and initially non-trusted stakeholders) [9, 11].

The disintermediation allows, for example, that two or more parties are able to conduct an exchange upon an agreement without requiring the presence of a third party acting as an intermediary. Also, the ability to track and trace the origin of documents that provides the digital evidence to guarantee integrity, authenticity, and auditability of digital evidence as it moves along different levels of hierarchy in the CoC [10]. Such elements are also of equal relevance, for instance, in the gem industry [3], in which blockchain has a strong potential

to reinforce claims by providing an immutable record of a product's history that can be verified. Another example is observed in the IoT (Internet of Things) area, in which Ryu et al. [14] use blockchain to track digital evidences of devices in a decentralized manner.

In this regard, this work aims to investigate the digital forensic evidence workflow in a company to identify one or several use-cases where blockchain does provide a benefit in terms of transparency and traceability of digital assets. These use-cases are evaluated concerning its feasibility and necessity of a blockchain implementation. Also, as the main outcome of such use case analysis, a Proof of Concept (PoC) of a blockchain-based CoC solution will be implemented.

1.2 Description of Work

This bachelor thesis demands, in an initial stage, to acquire the conceptual elements that are involved in the proposal, checking whether blockchain does provide an effective approach to ensure transparency and increase trust in a CoC. The theoretical background cover the fundamentals of blockchains as well as custody chains as a basis for a further analysis. In this regard, the analysis determines not only whether the blockchain-based approach is useful or not, but which type of blockchain deployment is most suitable in a positive case. Therefore, it is paramount to elicit specific application details and requirements, which are investigated along with a company. Therefore, the goal of this thesis is to improve the integrity, validity and authenticity of the digital evidence, acquired by the digital forensic team. A PoC is implemented covering the life cycle of the digital evidence in an investigation. An evaluation should assess whether the requirements have been accomplished and what kind of threats could pose problems.

1.3 Thesis Outline

The remaining work is divided into 4 sections. Chapter 2 will introduce the concept of blockchain technology and digital forensic - both relatively new and not much researched yet. Furthermore, threat modeling with focus on STRIDE [6] will be explained as it will be used as evaluation tool later on. Chapter 3 will describe the architecture of the prototype. It includes a section where the requirements are elicited from a real life company. Following is a section on the design decisions. In chapter 4, the PoC undergoes a use case study and an evaluation of possible threats. Chapter 5 concludes this bachelor thesis with a summary and proposition for future work.

Chapter 2

Background and Related Work

2.1 Blockchain Technology

The revolutionary paper "Bitcoin: A peer-to-peer electronic cash system" by Satoshi Nakamoto [11] introduced a system without an intermediate party for the first time. Underlying the bitcoin application is a public distributed ledger termed blockchain which relies on a distributed network whose participants or nodes have the exact copy of the blockchain locally. Participants issue transactions which then are collected in blocks. Each block is time-stamped and linked to the previous one with a cryptographic hash and hence make up the blockchain.

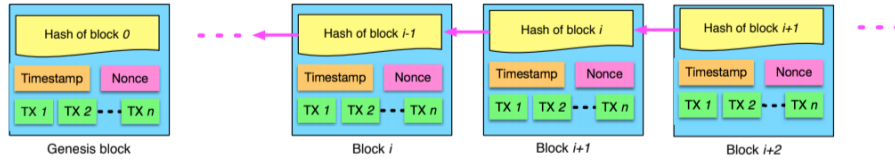


Figure 2.1: Example of blocks in a blockchain [21]

Since every node has a copy of the distributed ledger, a single main chain must be guaranteed. This is the responsibility of the consensus mechanism. Every node needs to verify the sequence of the blocks or transactions. Depending on the kind of consensus mechanism, the correct sequence of the main chain is chosen by the majority of nodes with the same result, or the node who solved a complex mathematical problem. Either way, once a correct sequence has been found, the changed state of the blockchain is broadcast to every participant in the network. Due to this, blockchain is suitable as a system for parties, who do not trust each other to have a trustworthy exchange.

Basically, a blockchain works similar to a traditional database. With the exception, that anything that is put into a blockchain cannot be deleted or changed, and, therefore, allows for permanent storage of information. Every node in the blockchain network has a full replication of the blockchain, traditional databases have a master/slave relationship,

resulting in the slave database to synchronize to the master database. The consensus of a blockchain is being determined by the majority of the nodes agreeing on an outcome, whereas in the traditional way the transactions are simply distributed amongst participants. In a blockchain every peer can validate a transaction.

Properties	Blockchain	Traditional database (DB)
Operations	Insert, read	insert, read, update, delete
Replication	Full replication	Master-Slave
Consensus	Peers agree on an outcome of transactions	Distributed transactions
Invariants	Any peer can validate a transaction	-

Table 2.1: Comparison between blockchain and traditional databases [13]

With a technology not relying on a trusted third party, the financial sector saw the opportunity and developed the first cryptocurrency, as proposed by Satoshi [11]. Without an intermediary, transactions across borders were more direct and faster. Following the first success, many more cryptocurrencies appeared on the web and as time went on, other sectors were interested in the perspectives of blockchain applications [12] beside money transaction.

The capacity to provide a trustworthy, decentralized, and publicly available data storage makes blockchain an interesting opportunity for organizations to increase business agility and reduce costs by removing intermediaries in distributed applications (*i.e.*, by involving multiple and initially non-trusted stakeholders) [12]. The immutability is one of the essential attributes of this novel technology, which brings benefits to many applications that demand non-repudiation features to avoid forging data. Once transactions are written in a block, distributed over the blockchain network, and chained with other blocks, it is not possible to forge the respective transactions on the entire blockchain network. Therefore, the combination of strategies for digital verification and validation of documents with blockchain can be an ideal solution to combat the increase of fraudulent activities in this context.

2.1.1 Types of Blockchain

Different kinds of blockchain exist, each suitable for specific use cases. Rodrigues et. al [13] propose a differentiation of four types, each defined by write (x-axis) and read (y-axis) access as seen in figure 2.2.

A public blockchain is open to any node, allowing them to view any transactions in the network. Above all, anonymity is another crucial feature. Cryptocurrencies, like bitcoin, are typically public since they allow anyone to join without disclosing their identity. Most often, Proof of Work (PoW) is applied to public blockchains. This consensus mechanism includes every node and has them work on a computational problem. A transaction can therefore take a long time to be validated, *e.g.* a transaction in bitcoin takes approximately 10 minutes to be included.

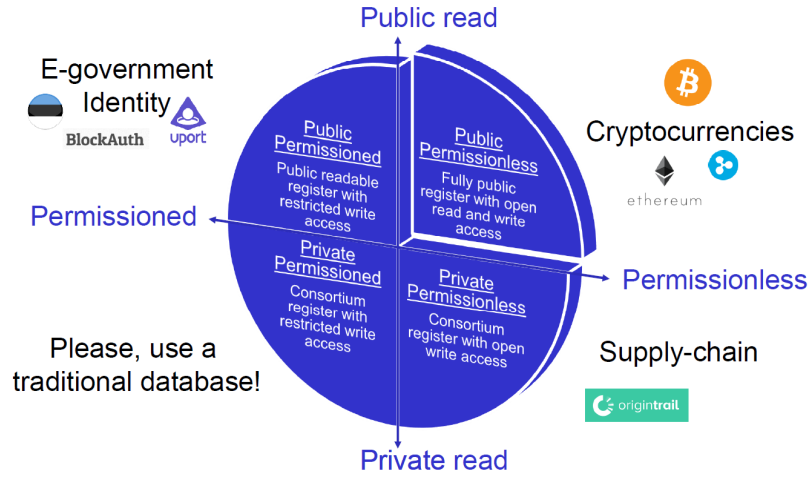


Figure 2.2: Types of blockchain [13]

Only authorized nodes can join a private blockchain, hence nodes are known amongst each other. Due to restricted joining, less nodes are present than in a public blockchain. Private blockchains most likely use the Practical Byzantine Fault Tolerance (PBFT) as consensus, resulting that transactions take less time as only a selection of nodes validate them. Private blockchains have been proposed in the supply chains, *e.g.* for the gem industry [3], where only a defined group, such as the end stakeholder selling the goods, should have access to the data. Information, including origin, way of transportation, middlemen etc., would allow for higher control and quality assessment.

A permissionless blockchain allows anyone to write, and therefore issue a transaction. This is a necessary requirement for cryptocurrencies since anyone joining should also be able to transfer money to anyone in the network. On the other hand, a permissioned blockchain has a defined group of participants who have the permission to write. An example is an e-government, where only government employees may upload information to the network.

2.1.2 Consensus Mechanisms

Proof of Work (PoW) was introduced in Bitcoin [11], being the first consensus algorithm completely decentralized. In PoW, nodes known as miners participate in a hash competition in which a miner needs to solve a partial hash collision competition, *i.e.* miners are required to find a hash with a minimal number of zeros in the input, satisfying a certain target. This mechanism introduces a computational overhead between miners to find the target hash in minimal time.

Proof of Stake (PoS) is an alternative to PoW in which, instead of requiring a computational effort, miners are able to create blocks based on the amount of resources they have at stake [22]. Based on such approach, PoS is able to reduce the energy costs of the expensive PoW mining processes as well as the dependence on specialized hardware. However, in a PoS-based blockchain, where no resource expenditure is required, the network is more susceptible to attacks.

Delegated Proof of Stake (DPoS): This mechanism projects a representative democracy where stakeholders elect nodes to generate and validate blocks [22]. Malicious nodes will not pose a problem as they can be easily removed from the delegation.

Practical Byzantine Fault Tolerance (PBFT) is designed to tolerate Byzantine failures [22]. Nodes are organized to operate in rounds, where in each round a primary node is chosen to insert the next block to the chain. The process consists of three phases: Pre-prepared, prepared and committed. A node must receive $2/3$ votes from all nodes to move from one phase to another. The total number of nodes must be known for this consensus mechanism to work.

Proof of Authority (PoA) belongs to the same family as the PBFT [4]. A group of nodes are recognised as authorities and identified by a unique id. It is assumed that the majority can be trusted. The mining happens in rotation, where the creation of blocks is evenly distributed among the authorized nodes. PoA is most often present in permissioned blockchains.

2.1.3 Smart Contracts

Smart contracts are computer codes that are stored in the blockchain. As the name suggests, it acts like a traditional contract where two or more parties are involved and sign a contract with defined inputs and outcomes. Smart contracts are only executed when the terms are fulfilled and can be invoked by any node that is part of the blockchain network. A simple example of a smart contract: An object, *e.g.* a car, is being auctioned and participants Bob and Alice bid money on it. The auctioneer sets the car's minimum bidding value at 500 which is not known by any participant. Bob bids 400 and Alice 600. Once the bidding deadline has been reached, the contract identifies the highest bidder who also fulfills the min. bidding value. Alice outbid her competitor and is the new owner of the car. The amount is being subtracted from Alice's e-wallet and added to the car's owner. Consider a more complicated case where Alice only has 550 in her wallet and still bids 600. This situation makes the bidding invalid and therefore would make Bob the new owner of the car. However, he has not bid the minimum amount of 500. Therefore, the auction is terminated as the contract has not been satisfied by any party. The auctioneer has to re-transact the bidding contract for the car.

2.2 Digital Forensic

Over the years, digital devices, such as computers, mobile phones, internet of things (IoT), became so sophisticated that we cannot imagine our lives without them anymore. They are present in our private life as tools for communication, entertainment, and as the main tool in the work environment. Just as it improved everyday life it also opened the doors to crimes. Cybercrimes, which is the term for any crime involving computers and networks, have also become more sophisticated and therefore make it difficult for investigators to uncover the truth. The term, digital forensic, is part of the forensic science that tries to

solve cybercrimes by recovering relevant information that indicates an alleged crime. Even though digital forensic has come far, tools constantly need to be improved or invented as technology progresses [5]. Over the years, many digital forensic models have been proposed but no general consensus has decided what kind of model to use. Yusoff et. al [20] examined selected models and tried to abstract and generalize the common processes into 5 steps:

- **Pre-process** Everything that needs to be done prior the actual investigation, *e.g.* getting legal approval and preparation of tools.
- **Acquisition and preservation:** The phase where relevant tasks revolving data is performed.
- **Analysis** The main part of any investigation, where investigators systematically analyze data to discover possible evidence, suggesting a crime or a person connected to a crime.
- **Presentation** Findings are reported to the authority, which help refute or prove an alleged crime.
- **Post-process** Involves the closure of the investigation and the proper storage of documents and data involved.

Just like in traditional investigations, digital forensic must follow the legal framework to admit digital evidence as credible in front of the court [2]. Processes and documents during the investigation must follow the standard procedures for evidence in court. Such documentation that archives the handling of evidence is called chain of custody (CoC).

2.2.1 Digital Evidence

Digital evidences are like traditional evidence (*e.g.* a piece of an object) that might be subject to an alleged crime. In theory, there is no difference in evidence being physical or digital, but in reality, the handling of digital evidence is more complex. A typical approach involves storing a forensic image - a bit-by-bit copy - of the digital device which is done during the acquisition and preservation phase of digital forensic. Since certain device types can store terabytes of data, a corresponding environment to be able to store such large data is needed. Digital evidence is also more fragile and fleeing in nature than physical ones and therefore face the challenge of possible manipulation, *e.g.* due to acquisition technology. To make sure that evidence is valid, techniques like sha algorithm is used for the original data and the forensic image to assure that the copy is not corrupted.

2.3 Threat Modeling

Threat modeling is a process which identifies possible threats or vulnerabilities in the system and assesses their danger. The goal of threat modeling is the prioritization of threats,

so that appropriate mitigations can be chosen. Many threat models exist, like STRIDE [6], LINDDUN [19] or PASTA [17], each of them having advantages, disadvantages and specific purpose. STRIDE is the most used and matured tool for threat modeling. It is easy to use and focuses on the security of its products. LINDDUN explores privacy threats and PASTA is a risk-centric model that brings business and technical requirements together. STRIDE is the first formally written threat model by Microsoft. The name itself is an acronym for the following threat types:

- **Spoofing** The attacker impersonates another person or uses their password to act as that person. Spoofing is a threat to authenticity.
- **Tampering** It is the act of purposefully modifying data and violates the integrity of data.
- **Repudiation** Untraceable illegal actions fall into repudiation threats. A user can dispute his crime since no proof can be given otherwise.
- **Information disclosure** Nowadays known as privacy breach, is the threat where sensitive information is visible to people that are not supposed to see it. Confidentiality is desired to counter information disclosure.
- **Denial of service (DoS)** A threat where the system becomes temporarily unavailable. These kind of attacks lower the reliability of the system.
- **Elevation of privilege (EoP)** A person gives himself unlawful privilege to restricted actions which can compromise the whole system. Authorization is the desired property to suppress such threats.

The first step in the STRIDE model is to create a Data Flow Diagram (DFD) of the system. It will help frame the threat analysis as concrete boundaries between the system, its actors (internal and external), the processes and the components that are drawn. Figure 2.3 shows an example of what a DFD looks like. Each component - external entity, processing node, data store and flow - will be analyzed for possible vulnerabilities and attacks. According to STRIDE [8], external entities are prone to spoofing and repudiation, data flow and storage to tampering, information disclosure and denial of service, and process node to all threat categories as seen in 2.2. Repudiation may be a potential threat if data storage contains logging or audit data. The list is complete once sufficient threats have been found. Following the threat finding is the risk assessment, which orders the threats according to their risk impact and probability. Mitigation options are created for the highest prioritized threats as they pose the most imminent danger to the system.

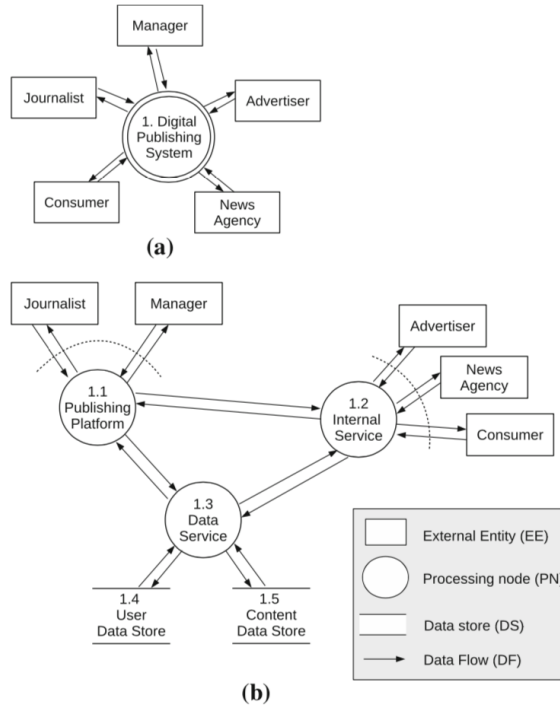


Figure 2.3: A DFD for a digital publishing system. **a)** Level 0 (context diagram) **b)** Level 1 [15]

	S	T	R	I	D	E
EE	X		X			
DF		X		X	X	
DS		X	(X)	X	X	
PN	X	X	X	X	X	X

Table 2.2: DFD elements mapped to STRIDE [8]

Although no formal threat model for smart contracts or blockchain have been published yet, threats and vulnerabilities still exist and need to be explored. Known threats that have occurred during the life of blockchain technology, have been documented to prevent future network failures. Contributors to the Decentralized Application Security Project (DASP) [1] have collected ten known threat types for smart contracts, in regards to Ethereum and Solidity, the main language for smart contracts in Ethereum.

- **Reentrancy** This famous threat led Ethereum to lose 50 Million USD, resulting in a hard fork of the network. In a reentrancy attack, a new call is allowed to a calling contract while it is still executing. The state of the called contract may change abruptly during said execution.
- **Access control** This is a common threat to all kinds of technologies. Vulnerabilities in the logic may give attackers access to private variables or logic and therefore the possibility to exploit the system.

- **Arithmetic** Integer underflow and overflow are common problems in computer science. Smart contracts are even more prone as they work with unsigned integer, whereas most programmers are used to the simple int types.
- **Unchecked low level calls** Low level calls in Solidity have a different error behaviour than other Solidity functions. As they do not propagate, a total reversion of an execution may not be possible. They will return a call with value *false* and let the code continue.
- **Denial of services** An overlapping vulnerability to STRIDE that has the possibility to kill a smart contract for good.
- **Bad randomness** Randomness is hard to achieve in Ethereum. It is either under influence of miners or viewable by the public and therefore predictable.
- **Front running** In public blockchains like Ethereum, everyone can see each transaction. If one node solves a puzzle, another node may steal the solution and re-enter that transaction with a higher fee. This transaction would then be mined faster than the original.
- **Time manipulation** Smart contracts may rely on time for their fulfillment and these timestamps are conveyed by miners. Avoiding trusting the miners' timestamp will reduce the risk of time manipulation threats.
- **Short addresses** The Ethereum Virtual Machine (EVM) has a problem where wrongly padded arguments are processed. Poorly coded clients are being exploited with special addresses that result in incorrect arguments inserted to the blockchain. There are different opinions whether this issue should be corrected by the smart contract or the EVM.
- **Unknown unknowns** It is necessary for the community to be aware that smart contract is not at its desired stability yet, and therefore highly susceptible to threats. More threats will likely appear, people involved in the blockchain should keep an eye for any unknown threat.

2.4 Related Work

Cartier et al. [3] proposed blockchain to trace gems along their supply chain. Due to the complicated and fragmented industry, not much is known about the mining, manufacturing and selling of specific gem materials. This could *e.g.* result in immorally mined diamonds such as blood diamonds. They argued that transparency in the CoC would reinforce the claims by providing immutable records, which blockchain could offer. A potential small QR code could be imprinted on the diamond itself. When scanned, information such as origin, quality etc. could be shown. However, they also mention that blockchain is only as strong as the data is. The event resulting in the supplied data must still be supported with standard procedures in the supply chain.

Giova [7] believes that the golden age of digital forensic has come to an end, and efficiency and coordination among the investigation teams must be improved. He claims that the CoC of digital evidence must be strengthened to appear more admissible in court. Therefore, he proposes to use the Resource Description Framework (RDF) and a flexible metadata schema such as the AFF4 format to create a more reliable framework. The automated framework would take over the interaction between humans and the CoC software.

ProvChain, designed by Liang et al. [9], is a secure architecture to collect and verify cloud data provenance. Cloud data provenance security is important, since it records the history, creation and operations of cloud objects. ProvChain data is embedded in blockchain transactions, making records tamper-proof, transparent and help improve privacy and availability. The architecture consists of three phases: 1. data collection, 2. data storage and 3. data validation. Once a user operates on a cloud object, it collects the operation metadata and inserts it into the blockchain and the provenance database. For the validation, the blockchain receipt is requested which consists of the transaction metadata and the merkle proof. The proof recreates the merkle tree, which is a complex hash node structure, to its root node. The calculated hash will be compared to the root hash, that is anchored in a specific block. If they are identical, the transaction can be assumed to be authentic.

IoT devices such as smart cars, smart homes, smartphones etc. can become part of cybercrimes. It is expected that 26 billion IoT devices will exist until 2020, prevailing digital forensic methods would not be able to handle it. Ryu et al. [14] proposed blockchain as a way to deal with IoT digital forensic. Current digital forensic methodology is based on centralized architecture and therefore vulnerable to data tampering, transparency and reliability issues. Blockchain as a distributed and immutable system could improve those vulnerabilities. Since IoT devices communicate with each other, the idea is to include communication data between the device in the blockchain. Transaction would be verified by a group of people, like the manufacturer, device user, service provider and the investigator. When a cybercrime is identified, the investigator may request access to this forensic blockchain.

Lone and Mir [10] have proposed a blockchain approach to improve the CoC in digital forensic. They evaluated the feasibility of blockchain in CoC by applying the flow chart from Wüst et al. [18]. The flow concluded that public or private permissioned blockchain is appropriate, depending whether public verification is needed. For their prototype, they opted for the Hyperledger Composer and smart contracts that allow participants to interact with the digital evidence. Four basic functions have been implemented for the prototype: create, transfer, delete and view evidence. The work concludes that blockchain increases trust and reduces conflicts without the need of a third party. Their future plans involve a blockchain plug-in for automated forensic tools that would support the forensic process during each step.

Tian et al. [16] go even further and created a framework for digital evidence, called Block-DEF. The framework separates the evidence, stored in a database, and the evidence information, stored in the blockchain. It supports evidence collection, storage, verification and retrieval. The Block-DEF verifies the integrity and validity of the evi-

dence whenever it is retrieved. Since blockchain have scalability issues, namely blockchain bloat, a lightweight blockchain and hybrid block structure has been chosen. A blockchain bloat occurs, because each node of the networks stores all blocks. As the chain grows, so must the storage capacity of each block. The study concludes that Block-DEF can satisfy the scalability, integrity, privacy, validity and traceability requirements of a CoC.

Chapter 3

Design and Implementation

This thesis uses the framework and workflow of an existing company, as shown in figure 3.1, and tries incorporate a blockchain service. The resulting prototype should cover all defined requirements.

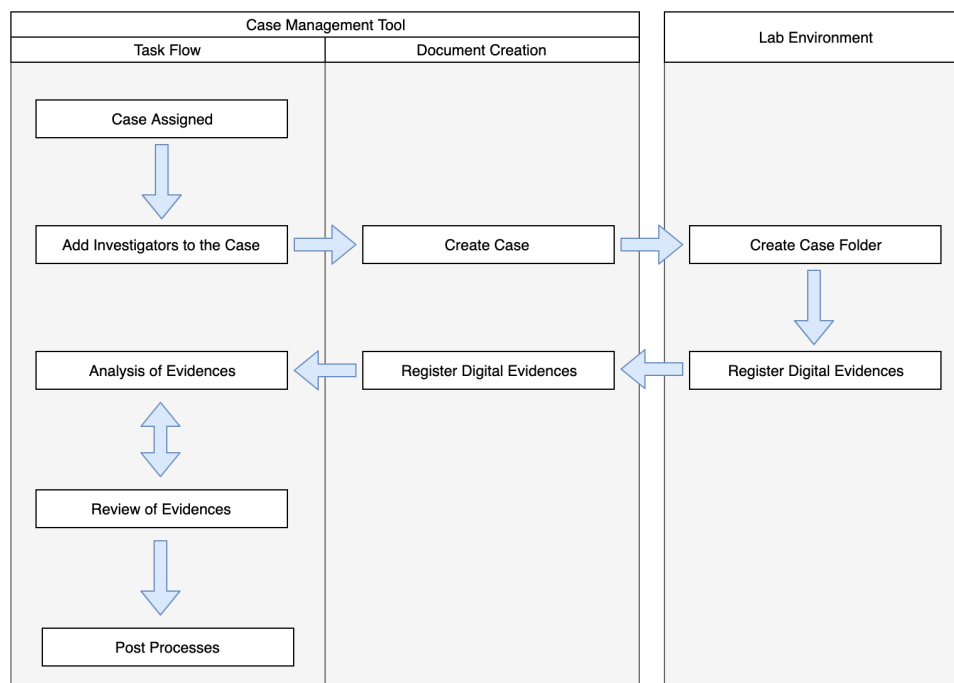


Figure 3.1: Workflow for the evidence and case management

3.1 Workflow

The workflow starts with the assignment of an investigation case. A group of people - the so called team leaders - are responsible in assigning their investigators to cases. Each team is defined by a geographical area, *e.g.* the team Asia consists of people working in

countries on the Asian continent. In case of an absence, the team leader may temporarily delegate his privileges to a fellow team member. Once an investigator is being assigned, he or she can register a case in the case management tool, and open a case folder in the lab environment. The lab holds multiple servers where all the digital data is being stored. An investigator assigned to a case has the permission to assign more investigators to the case, just like the team leader. The team leader has additional rights, such as adding himself to cases. Only those actively involved with the case have the access to the corresponding data in the lab.

As part of the preparation, for each subject and device potentially involved in a crime, a consent form must be created. Without a consent or a legal approval, a device cannot be acquired. Once the consent has been signed by the subject, the device or devices are acquired and a forensic image, which is an identical copy of the device, is produced. Most of the time, the forensic image is on a retention media which then must be transferred into the lab environment. A validation is run to see whether the acquisition might have corrupted or modified the data. If that is not the case, the data is transferred to the lab and the retention media can be safely emptied. The verified data is then converted and pre-processed, *e.g.* removing redundant data like installation files. As soon as the evidence is registered in the lab environment, an entry for the digital evidence must be added in the case management tool as well. Once all data has been processed in the lab, the case goes into analysis and review mode. In the analysis phase, investigators go into in-depth search for potential evidence. Lost or deleted files are recovered during the processes. The review phase, which can be executed in parallel with the analysis, has the investigators and externs, *e.g.* lawyers, conduct searches for information that can be used against or in favour of an alleged crime.

At the end of the life cycle, a case is closed. The digital data of the devices must be legally stored until the end of the current year. Data retention can be extended if the data is part of an ongoing lawsuit. If no data retention has been issued for digital data, the data can be safely removed. Other data, *e.g.* case information in form of a document, must be kept for twenty years after closing the case.

3.2 Requirements

The goal of this thesis is the attempt to improve the integrity, validity and authenticity of the digital evidence acquired by the digital investigations team. A proof of concept will be implemented that covers the life cycle of the digital evidence in an investigation. The prototype shall have the following properties:

Functional requirements:

- (R1) Only registered **accounts** should be able to interact with the smart contract.
- (R2) The smart contract shall allow different **actors**. From the section beforehand, three distinct actors can be identified: the global leader, the team leaders and the investigators.

- (R3) Implementation of the different **tasks** that are restricted to the roles.
- (R4) A **data structure** that maps the life-cycle of an **evidence**.
- (R5) Allow access to shared data.

Non-Functional requirements:

- (R6) Use a **permissioned blockchain** and **smart contracts**.
- (R7) **Off-chain (dummy) evidence** that can be hashed and uploaded into the blockchain.
- (R8) Assure the **integrity** of the **evidence**.
- (R9) Assure the **integrity** of **investigators' actions**.

3.3 Architecture

Since only authorized accounts should be able to interact with the service (R1), all permissionless blockchains drop out. Permissioned blockchain allow for controlled joining of nodes. Smart contracts enable to implement scenario-specific functionalities, and control the input as well as the outcome of such contracts (R3).

As there already exists a system to store the actual digital evidence, there is no need to upload the whole data to the blockchain. This comes in handy since the blocks are limited in size. The network should only occupy himself with the hash and the relevant metadata of a digital evidence which results in (R7). Since the solution deals with sensitive data, integrity of each evidence must be guaranteed (R8). Blockchain as a persistent database does not allow modification on data included in the network. The integrity of the evidence can be checked by its state in the blockchain. Above all shall the blockchain solution assure the validity of the investigators' work (R9). Investigators shall prove their actions the same way as the integrity of evidence.

The use case diagram 3.2 shows the three actors and their possible actions, reflecting (R2) and (R3). The investigator may register, verify and view evidences in the blockchain. Cases, as described in the workflow, will not exist, instead anyone from the same team can interact with the respective data in the blockchain. For simplicity, investigators will not be able to add new team members to their own team, only the team leader has the authorization to add investigators to the team. The team leader has the full authorization of the team and may remove an investigator or an evidence that has been registered by one of his team members. In case when the team leader is absent, *e.g.* vacation, he has the option to delegate the privilege to another team member. As the name suggests, the global leader has the highest authorization and assigns the team leaders. He may also assign investigators to any team.

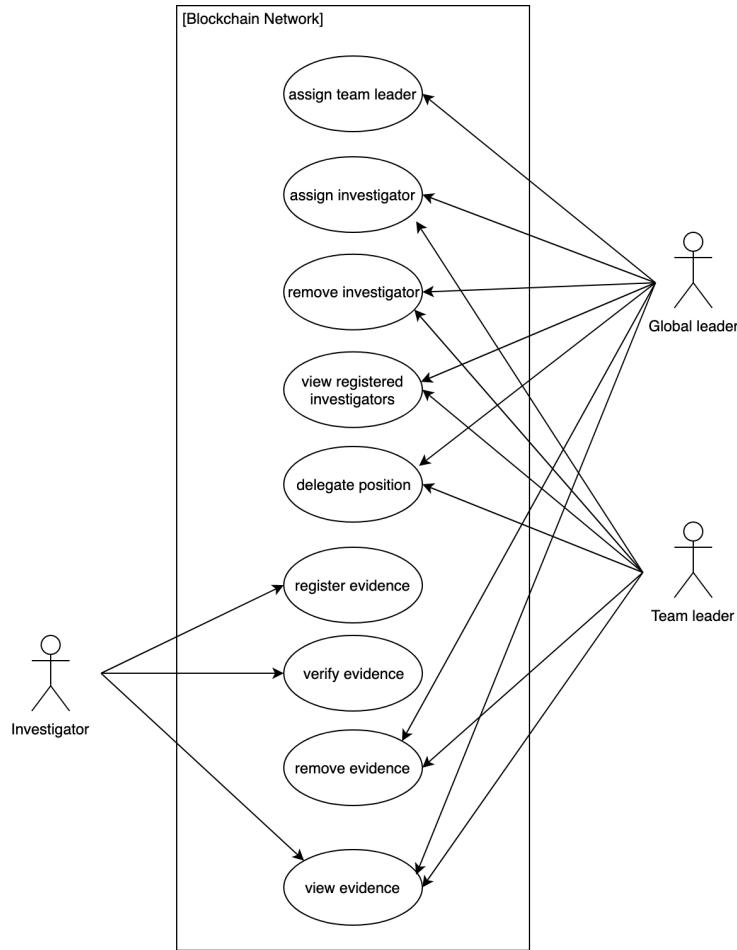


Figure 3.2: Use case diagram

The ER model 3.3 displays the relationships between the actors and the different entities. Every investigator and team leader belongs to a certain team which is characterized by the geographical area. Investigators and team leaders can only interact with evidences or other team members in their geographical area, this includes assigning and removing an investigator, and registering, verifying, viewing and deleting an evidence. The global leader has a special role as he assigns team leaders and is not restricted to any geographical area. He has full access to all evidences and investigators.

The figures 3.4 and 3.5 show the process involving registering and verifying evidence as required by (R4). Unlike the use case diagram 3.2, we distinguish between the investigator/sender and the verifier as they have distinctive tasks. For a specific evidence, the sender cannot take the role of the verifier at the same time. They both belong to the group investigator. Figure 3.4 shows the context in which the evidence is being handled. The investigator acquires the appropriate legal approval for the digital device. The digital device can then be collected and a forensic image is acquired. The path splits into two where the digital data itself is transferred to a server environment for storage and preservation, and the metadata is inserted to the blockchain. The metadata includes the unique identifier of the digital data and the SHA-1 value. The verifier's job is to verify that evidence. Step 1 is to run an additional SHA-1 encrypter on the data and acquire

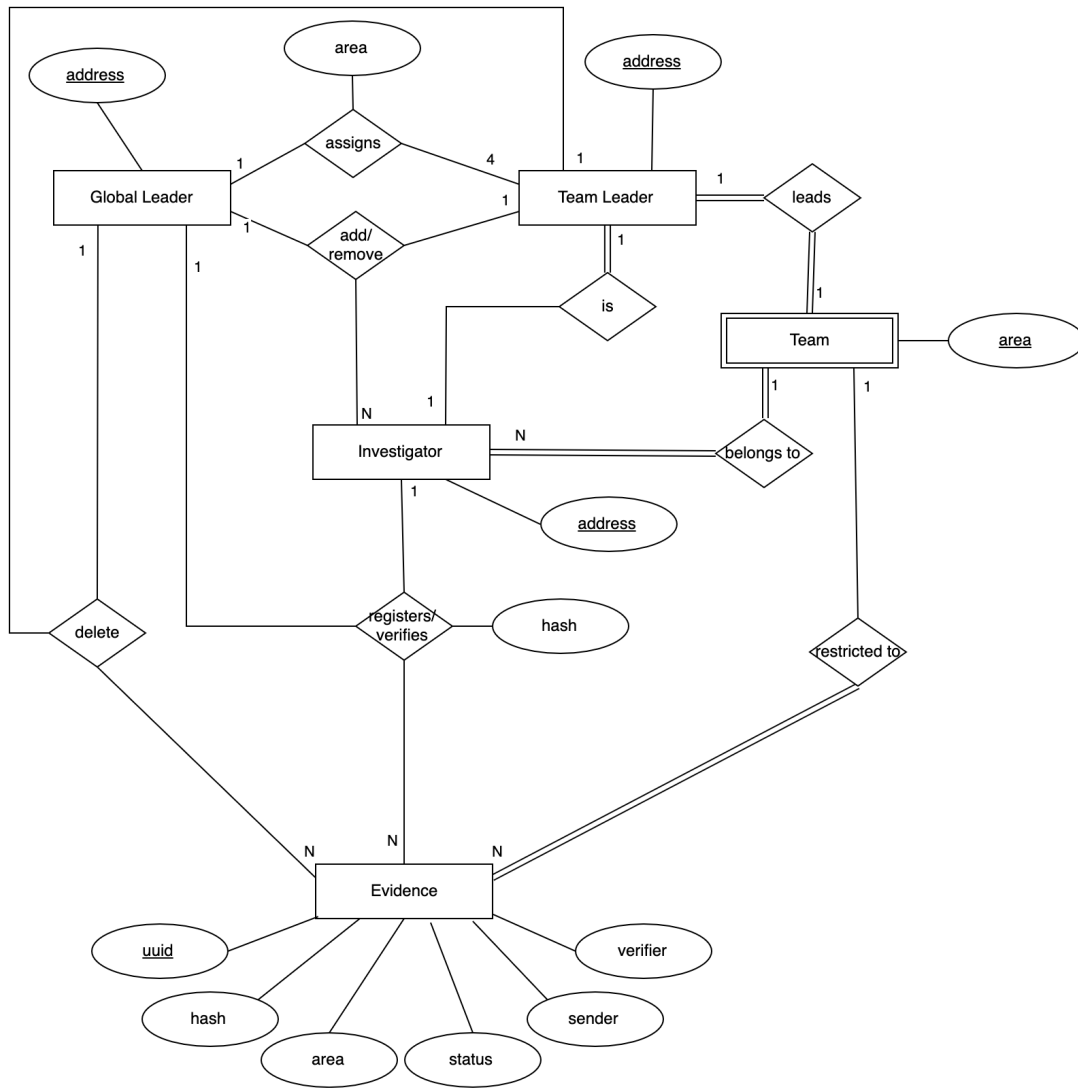


Figure 3.3: Entity relationship model

the hash. Step 2 is the comparison between the stored hash in the blockchain and the new calculated one. Ideally, they would be identical which would mean that the data in the server environment has not changed since the upload of the initial hash.

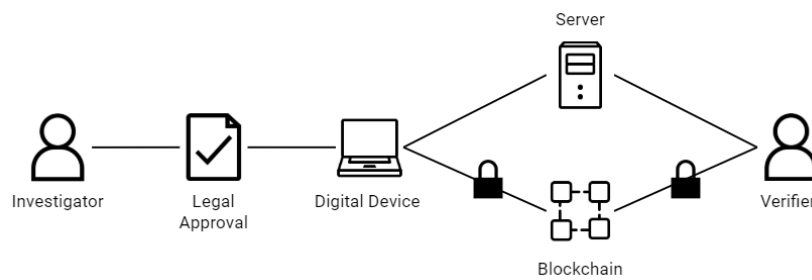


Figure 3.4: Evidence process

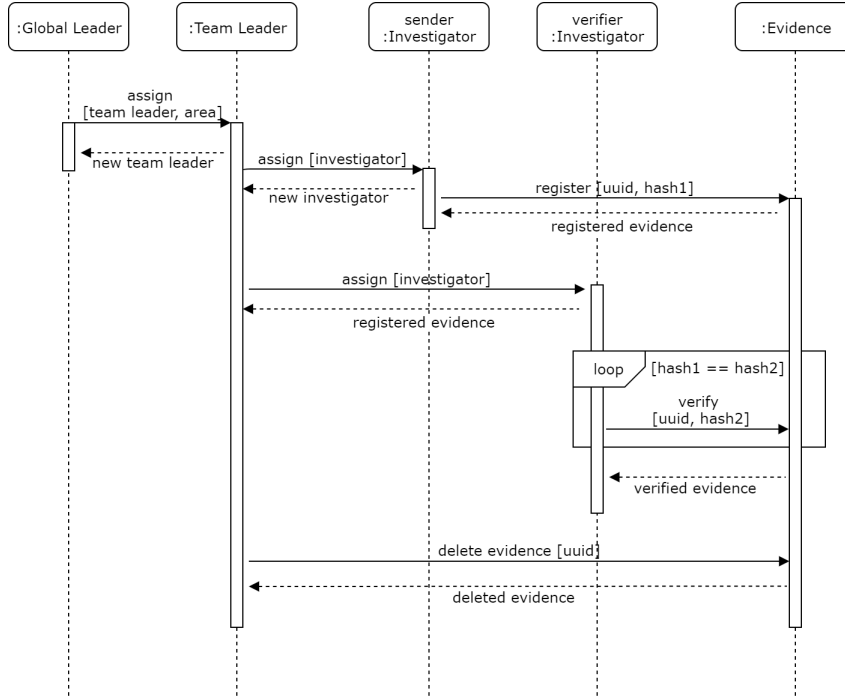


Figure 3.5: Sequence diagram

The sequence diagram 3.5 shows an ideal situation for the handling of an evidence. Both the sender and the verifier are investigators assigned by their team leader. The sender registers the evidence by uploading the hash of the evidence data with the unique identifier of the evidence to the blockchain. The actual evidence data is transferred to the existing laboratory as seen in figure 3.4. The verifier - another investigator from the same team - has the duty to verify the evidence by providing another hash. The hash is gathered by recalculating the hash from the laboratory residing evidence. If the hash is identical to the existing hash in the blockchain, proof exists that there has been no tampering between the registering and verification of the evidence. Due to legal reasons, the evidence must be retained until the end of the year after closing the case. After the passing of the retention time, the team leader is responsible for the appropriate deletion of the evidence in the laboratory as well as in the blockchain. There are other use cases in the evidence process, which would lead to a slightly different sequence diagram. Since the global leader has the highest authorization, he could replace the team leader in the sequence diagram, meaning he would assign the investigators and be responsible for the evidence. Team leaders and the global leader may also take on the role of an investigator, but then are restricted to the sender/verifier distinction.

3.4 Implementation

Truffle is used as framework for the smart contract project. The created prototype is written in Solidity, an object-oriented programming language for smart contracts, most notably seen in Ethereum. The code is then uploaded to Ganache, which simulates a

local running Ethereum network without actual miners. Therefore, every transaction is guaranteed to be mined.

The prototype consists of one smart contract called FCC.sol which consists of global variables and functions. The variables are the *global leader*, *allInvestigators*, *teamLeaders* and *evidences*. Every team leader added to the contract is automatically part of the mapping *allInvestigators* as well. The variable *evidences* consist of evidence structures. Each evidence is defined by a unique identifier (uuid), a hash, a geographical area or team, a status, and the addresses of the sender and the verifier.

The functions are similarly built to Java functions, consisting of a function name plus the arguments it expects. The function *addInvestigator* requires two arguments, the address of the new investigator and an integer *i*. The function call also includes a modifier named *onlyIfLeader* which has been defined prior this function. It states that the function can only be called by a team leader or the global leader. In the body, an additional requirement, that the new investigator has not been registered yet, is defined. If that is the case, the investigator is added to team leader's team. The integer *i* comes only into play when the caller of this function is registered as the global leader. Since he technically is not assigned to an area, he must explicitly convey the argument in form of an integer. The areas are numerically listed. At the end, the event *AddedInvestigator* is triggered and a boolean value is returned.

```

1      function addInvestigator(address addr, uint i) public onlyIfLeader
      returns(bool success){
2
3      require(allInvestigators[addr].invAddress == address(0), "This
      investigator already exists.");
4
5      GeographicalArea a;
6      if(globalLeader == msg.sender){
7          a = GeographicalArea(i);
8      } else {
9          a = allInvestigators[msg.sender].area;
10     }
11     allInvestigators[addr].invAddress = addr;
12     allInvestigators[addr].area = a;
13
14     emit AddedInvestigator(addr, areaAsString(a));
15     return true;
16 }

```

Listing 3.1: Adding an investigator to a team

The function *verifyEvidence()* takes two arguments as well. The modifier *onlyIfAuthorized* states that only accounts which have been registered in the smart contract, either as investigator/team leader or global leader may interact with this function. In contrast to *addInvestigator*, *verifyEvidence()* has more requirements. First of all, the evidence must exist in the network. Second, it must not have been verified yet, and at last, the caller of this function must not be the register of this evidence. A team leader has an additional condition, where he must be assigned to the same geographical area the evidence has been registered in. The global leader omits this part as he is not assigned to any area. Is the provided hash identical to the one uploaded to the blockchain, the status will be changed

to *Verified* and the event *VerifiedEvidence()* is emitted. On the contrary, if the hash is different, the status will not be changed and the event *FailedVerification()* is triggered.

```

1  function verifyEvidence(string memory uuid, string memory hash)
    public onlyIfAuthorized returns(bool success){
2
3      require(evidences[uuid].sender != address(0), "The evidence does not
        exist or has been deleted.");
4      require(evidences[uuid].sender != msg.sender, "The sender is not
        allowed to verify the evidence.");
5      require(evidences[uuid].status != EvidenceStatus.Verified, "The
        evidence has already been verified.");
6
7      if(globalLeader != msg.sender){
8          require(evidences[uuid].area == allInvestigators[msg.sender].area,
9              "You do not have the authorization for this action.");
10     }
11     Evidence storage ev = evidences[uuid];
12
13     if(!isSameString(ev.hash, hash)){
14         ev.status = EvidenceStatus.FailedVerification;
15         emit FailedVerification(uuid);
16         return false;
17     } else {
18         ev.verifier = msg.sender;
19         ev.status = EvidenceStatus.Verified;
20         emit VerifiedEvidence(uuid, msg.sender);
21         return true;
22     }
23 }

```

Listing 3.2: Verifying an evidence

3.4.1 Functions

All functions in the smart contract (with the exception of functions used for demo purposes) are listed and explained followingly:

- *event*: Emits an output to mark an event.
- *modifier*: Functions that check whether the requirements have been fulfilled. They are extensions of a function call. Four modifiers have been implemented in this smart contract: *onlyGlobalLeader*, *onlyTeamLeader*, *onlyIfLeader* and *onlyIfAuthorized*.
- *assignTeamLeader(address addr, uint i)*: Has a modifier which restricts access only to the global leader. The account address of the team leader and the integer (mapped to the teams/geographical areas) are the arguments for the function. The successful function emits the event *AssignedTeamLeader(address, area)*.
- *addInvestigator(address addr, uint i)*: Extended by the modifier *onlyIfLeader*. The uint will only be processed if the function caller is the global leader. Same as above, the integer responds to the mapping of the teams. The function produces the event *AddedInvestigator(address)* at success.

- *removeInvestigator(address addr)*: Removes the investigator from the contract and emits *RemovedInvestigator(address)*. Only team leaders or the global leader are authorized for this action.
- *delegateTeamLeader(address newLeader)*, *delegateGlobalLeader(address newLeader)*: Delegation of the leader position to another investigator. In the case of a team leader delegation, a team member must be chosen. The global leader may choose any investigator to take over his role. Both produce the event *DelegatedLeaderPosition(msg.sender, newLeader)*. They are restricted by the modifier *onlyTeamLeader*, respectively *onlyGlobalLeader*.
- *registerEvidence(string memory uuid, string memory hash, uint i)*: Registers an evidence by mapping the evidence to a unique identifier *uuid* and the hash of the evidence data. It is extended with the modifier *onlyIfAuthorized*. The global leader evokes an additional function where the integer *i* responds to the mapping of the geographical area.
- *verifyEvidence(string memory uuid, string memory hash)*: Verification of an evidence by passing the unique identifier - to look up the evidence with the uploaded hash - and the newly calculated hash. A failed verification emits a *FailedVerification* event, a successful one a *VerifiedEvidence* event. Only authorized accounts may interact with this function.
- *showEvidence(string memory uuid)*: An evidence can be looked up when the unique identifier is known. The function prints information about the evidence, such as id, SHA-1 hash, area, status, address of the sender/registerer and verifier.
- *deleteEvidence(string memory uuid)*: The global leader and the team leaders may delete an evidence. The team leaders are restricted to evidences registered in their geographical region. The event *DeletedEvidence(uuid)* is emitted at success.
- *isSameString(string memory a, string memory b)*: It takes two strings and compares them. If identical, *true* is returned, *false* otherwise.
- *areaAsString(GeographicalArea a)*: A helper function that takes the area and returns the string of it.

Chapter 4

Evaluation

This chapter will evaluate the prototype and assess whether the requirements have been accomplished by conducting a use case study. It consists of two main actions: 1. adding and removing accounts, and 2. registering, verifying and deleting evidence. Afterwards, a threat analysis with the STRIDE threat model will determine the potential threats. The ten smart contract threats by DASP [1] will also be considered.

4.1 Use Case Study

4.1.1 Action 1: Adding and Removing Accounts

Since the prototype is deployed locally, the testing of the functionalities is also done on a local machine. The smart contract recognizes the local machine as the global leader and assigns the connected address to the variable *global leader*. Taking on the role, additional accounts must be added so that the evidence process can proceed. With the function *assignTeamLeader(address, uint)*, a team leader - uniquely identified by the account *address* - is assigned to one of the four teams, mapped by an unsigned integer (*uint*). Ganache initializes for each local network ten addresses, whereas one is the address of the local machine. One address each will be assigned to team America and team Europe with *assignTeamLeader(address, uint)*. Both teams will be populated with two investigators, again uniquely identified by their account address, with the function *addInvestigator(address, uint)*. Each of these calls triggers an event in the smart contract to show a successful registration of the account. In total, two teams, each consisting of two investigators and their respective leader, have been initialized in the smart contract.

At some point in time, an investigator may leave the company and the address, connected to the investigator, must be removed from the smart contract. This can be done with the call *removeInvestigator(address)*. The successful removal is again presented by an event in the network. Applied to team Europe, *e.g.* the account address

"0xAbe8E39c83C8A50546061aacca25ab31011943dc"

```

[truffle(development)> fcc.showInvestigator("0x2f87a865F00D85d953949DEaF790E46c7404C9dE")
Result {
  '0': '0x2f87a865F00D85d953949DEaF790E46c7404C9dE',
  '1': true,
  '2': 'Europe' }
[truffle(development)> fcc.showInvestigator("0xD65B3CB50B72c839dD636aF696Ed0744d033bc80")
Result {
  '0': '0xD65B3CB50B72c839dD636aF696Ed0744d033bc80',
  '1': false,
  '2': 'Europe' }
[truffle(development)> fcc.showInvestigator("0xAbe8E39c83C8A50546061aacca25ab31011943dc")
Result {
  '0': '0xAbe8E39c83C8A50546061aacca25ab31011943dc',
  '1': false,
  '2': 'Europe' }

```

Figure 4.1: Team Europe: Shows the address ('0'), a boolean indicating a team leader position ('1') and the team ('2')

is removed. The account is now invalid and no longer has the right to interact with the smart contract.

```

logs:
[ { logIndex: 0,
  transactionIndex: 0,
  transactionHash:
    '0x5e5c3cd986689a63588e66c6e55baba642df8a8fe4ea6c8c7daa83a815dccc35',
  blockHash:
    '0x9d5b2323c87f623707a1524edc287180708cc4775f59fad994eda12888dfefbc9',
  blockNumber: 203,
  address: '0xbB01e58F0cA5b09bBb697eeD1f783e880610Df40',
  type: 'mined',
  id: 'log_6cb0d55a',
  event: 'RemovedInvestigator',
  args: [Result] } 1 }

```

Figure 4.2: Log: Removed an investigator from team Europe

Taking on the role of the team leader of Europe, the empty spot needs to be filled. To add an additional investigator, the same function, *addInvestigator(address, uint)* is used. The smart contract identifies the local address as a team leader and therefore ignores the second argument, which would manually define the team assignment. In this case, the investigator is automatically added to the team leader's team, team Europe. Removing an investigator is just as straightforward, *removeInvestigator(address)* will remove an address from the smart contract. As opposed to the global leader, team leaders may only remove their own team members from the contract.

4.1.2 Action 2: Registering, Verifying and Deleting an Evidence

This scenario shall test the functionality of registering, followed by verifying of an evidence as part of the chain of custody. For this, three investigators and their team leader are involved under following assumptions:

- All investigators belong to the same team. Two of them have the role of the sender and the verifier - as specified in figure 3.5. The third investigator has no specific role, he shall be called the bystander.
- They have been registered in the smart contract and therefore either been added by their team leader or the global leader.

- The team leader is responsible for the three investigators.

Off-chain dummy evidence in form of a .txt file has been created to be able to test the functionalities. Multiple tools for SHA-1 encryption were available, the easiest one was a simple online encrypter which took the .txt content as input. The hash output was then manually saved in the same .txt file. Another tool, called EnCase imager, was used to generate E01 files, which is a standard forensic image format for hardware. A USB stick with random files has been formatted with the EnCase imager. Basically, it did the same as the SHA-1 online encrypter, with the exception, that additional metadata was produced that might be needed in a later stage of the forensic process.

Assuming the role of the sender, the hash, from a E01 or .txt file, would be uploaded to the blockchain with *registerEvidence(uuid, hash)*, whereas the uuid is the unique identifier. For the sake of this study, a uuid has been invented, in the actual workflow a uuid would be automatically assigned during the acquisition. The raw digital evidence would then be transferred to a laboratory. For testing purposes, the data simply resides on the local machine. The evidence (see figure 4.3) is now registered with the uuid, the hash, the area, and the sender's address. The verifier's address is empty since it has not been verified yet.

```
truffle(development)> fcc.showEvidence("de1")
Result {
  '0': 'de1',
  '1': '*****',
  '2': 'Europe',
  '3': 'Registered',
  '4': '0xEAc8EC8462f3F4d9CD4F82BbC5860d70Ce06F2bF',
  '5': '0x0000000000000000000000000000000000000000' }
truffle(development)>
```

Figure 4.3: The registered evidence "de1"

The verifier would now recalculate the SHA-1 on the digital evidence in the laboratory, here the local machine, and afterwards verify the evidence by uploading the hash with *verifyEvidence(uuid, hash)* to the blockchain. If the hashes are identical, the state of the evidence is changed to *Verified*, and the verifier's address is added. If not, the status changes to *FailedVerification*, indicating tampering of the evidence. Other accounts, who are not registered as team members, will get an error message if they try to verify an evidence they have no access to.

```
truffle(development)> fcc.showEvidence("de1")
Result {
  '0': 'de1',
  '1': '3fcd8234e87ea5ae2872ba1ee4f4111baf41d65a',
  '2': 'Europe',
  '3': 'Verified',
  '4': '0xEAc8EC8462f3F4d9CD4F82BbC5860d70Ce06F2bF',
  '5': '0x1fBd830eEeca902b4cB9c2a597194a27CDBCb791' }
truffle(development)>
```

Figure 4.4: The verified evidence "de1"

The bystander can look up the evidence with *showEvidence(uuid)* at any point of the process. A *FailedVerification* status will tell the bystander, that during the registration

and the verification tampering (deliberate or not) has occurred. Even if an evidence has been successfully verified, the bystander may check the integrity of the evidence. He would run the same hash algorithm on the evidence and look up the uploaded hash in the blockchain for comparison. This mechanism does not allow for untraceable modification, even after the initial verification of the evidence.

Under the assumption that the evidence has been verified correctly and no other incident has occurred and the legal retention time has passed, the team leader of Europe wishes to delete the evidence from the smart contract. The evidence can be deleted by the function *deleteEvidence(uuid)*, whereas the evidence must have been registered (and verified) as part of team Europe. An error appears whenever an unauthorized evidence is called or when an unauthorized personnel tries a deletion action. The deletion is straightforward, the *uuid* serves as look up key for the evidence, then the removal from the contract is initialized. *DeletedEvidence* marks the event.

```
logs:
  [ { logIndex: 0,
    transactionIndex: 0,
    transactionHash:
      '0x06b397e46fa46e39657b63a376420e0ca307bf38861c5b4834de8ea6a8831a6c',
    blockHash:
      '0x1b7105e1c75a9c05d8e05970bb74406ce0ab1944ed28eaedd4df1fcdbe4c73f4',
    blockNumber: 212,
    address: '0x30c2785d2912B30818F86Ef0c07B78c0dca82020',
    type: 'mined',
    id: 'log_e460b960',
    event: 'DeletedEvidence',
    args: [Result] } ] }
truffle(development)> fcc.showEvidence("del")
Thrown:
Error: Returned error: VM Exception while processing transaction: revert The evidence does not exist
or has been deleted.
```

Figure 4.5: Log: deleted evidence "del"

4.2 Threat Analysis

The STRIDE model is used to assess possible threats. The use case scenarios described above, will be the context for threat modeling. For step 1 of threat modeling, a data flow diagram (DFD) is created. The first diagram 4.6 shows a level 0 DFD, which provides a general overview of the involved actors with the system. The three actors, the investigator, the team leader and the global leader, can be seen interacting with the smart contract. Level 1 4.6 looks deeper into the actual processes of the system. The processes have been divided into 4 elements, processing nodes concerning the investigators (3.3), the team leaders (3.4) and the evidences (3.1), and the data store blockchain (3.2). The dotted lines draw the boundaries of the system.

Step 2 of threat modeling would be the identification of threats, whereas each identified component of the DFD is prone to specific threat types as seen in 2.2. In the resulting table (see 4.1), tampering, information disclosure and DoS appear to affect most of the components. Processing nodes in particular seem to be prone to threats.

According to STRIDE, the next step would determine the risk for each threat by calculating the impact and the probability. Since the goal of this thesis was to implement a

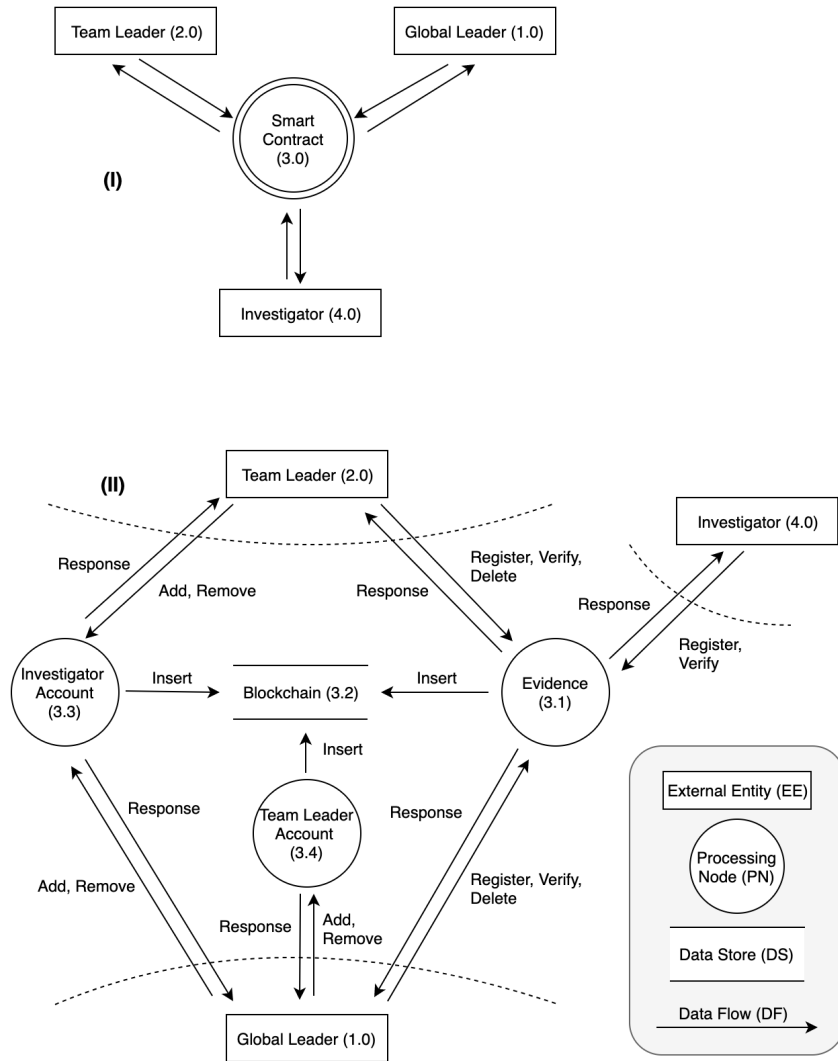


Figure 4.6: Level 0 (I) and Level 1 (II) Data Flow Diagram

PoC, a complete risk assessment does not seem appropriate. Also, planning the mitigation, the last step of STRIDE, will not be done in its entirety.

Nonetheless, the threats will be analysed and interpreted in regards to the described use case. Additionally, since the prototype is a smart contract, threats concerning this technology must be considered as well. However, not all of the ten threat types [1] are applicable. Bad randomness, time manipulation and unchecked low level calls will not be considered in the analysis, as the source code does not use any of the properties that could lead to such threats. Even though an unsigned integer is used in the solution, there is no danger for an integer over- or underflow. The integer is merely the mapping key to a list. Front running threats involve competition, which is not the case for this prototype. Since the solution shall be part of a permissioned blockchain, consensus mechanism such as PoW is not applied and therefore front running threats will not occur. DoS is already covered in the STRIDE model. Threats such as reentrancy, access control, short addresses and the unknown unknowns remain.

Threat Type (STRIDE)	DFD Item Numbers
Spooofing	External entities: (1.0), (2.0), (4.0) Processes: (3.1), (3.3), (3.4)
Tampering	Processes: (3.1), (3.3), (3.4) Data stores: (3.2) Data flows: (1.0 $\rightarrow$ 3.1 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.3 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.4 $\rightarrow$ 1.0), (2.0 $\rightarrow$ 3.1 $\rightarrow$ 2.0), (2.0 $\rightarrow$ 3.3 $\rightarrow$ 2.0), (4.0 $\rightarrow$ 3.1 $\rightarrow$ 4.0), (3.1 $\rightarrow$ 3.2), (3.3 $\rightarrow$ 3.2), (3.4 $\rightarrow$ 3.2)
Repudiation	External entities: (1.0), (2.0), (4.0) Data store: (3.2)
Information Disclosure	Processes: (3.1), (3.3), (3.4) Data stores: (3.2) Data flows: (1.0 $\rightarrow$ 3.1 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.3 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.4 $\rightarrow$ 1.0), (2.0 $\rightarrow$ 3.1 $\rightarrow$ 2.0), (2.0 $\rightarrow$ 3.3 $\rightarrow$ 2.0), (4.0 $\rightarrow$ 3.1 $\rightarrow$ 4.0), (3.1 $\rightarrow$ 3.2), (3.3 $\rightarrow$ 3.2), (3.4 $\rightarrow$ 3.2)
Denial of Service	Processes: (3.1), (3.3), (3.4) Data stores: (3.2) Data flows: (1.0 $\rightarrow$ 3.1 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.3 $\rightarrow$ 1.0), (1.0 $\rightarrow$ 3.4 $\rightarrow$ 1.0), (2.0 $\rightarrow$ 3.1 $\rightarrow$ 2.0), (2.0 $\rightarrow$ 3.3 $\rightarrow$ 2.0), (4.0 $\rightarrow$ 3.1 $\rightarrow$ 4.0), (3.1 $\rightarrow$ 3.2), (3.3 $\rightarrow$ 3.2), (3.4 $\rightarrow$ 3.2)
Elevation of Privilege	Processes: (3.1), (3.3), (3.4)

Table 4.1: Threats to the prototype

4.2.1 Action 1: Adding and Removing Accounts

The scenarios for action 1 involve the external entities (1.0) and (2.0), the processes (3.3) and (3.4), the data store (3.2), and the data flows (1.0 $\rightarrow$ 3.4 $\rightarrow$ 1.0), (3.4 $\rightarrow$ 3.2), (1.0 $\rightarrow$ 3.3 $\rightarrow$ 1.0), (2.0 $\rightarrow$ 3.4 $\rightarrow$ 2.0) and (3.3 $\rightarrow$ 3.2). The threats in this scenario are the most crucial ones, as attacks could render the whole smart contract unusable very quickly. However, the threat scenarios are not likely, as there is not much to gain from these actions. There is no data yet that can be stolen and no ongoing process that can be interrupted. The biggest weak point in the scenario would be the global leader (1.0), as he is the starting point of the whole system. He initializes the setting of the contract. If the global leader is impersonated by a malicious user, none of the implemented functionalities can be used. According to STRIDE, authentication is the best approach to mitigate spoofing threats.

From the list of possible threats 4.2, the team leader account processing node (3.4) is prone to a variety of attacks. Even though not in the list, the process (3.3) is assumed to be just as susceptible. Since the logic and the functionality of the prototype resides in the processing nodes, bad coding would lead to exploitation by malicious users. Utmost care must be taken to implement code so that threats are less likely to occur. Automated as well as manual testing should help assess the quality of the code. Repudiation and tampering

threats would make the prototype less credible, since our goal was the permanent and unmodified chain of custody of evidence. A complete audit trail, which can be proven to be unaltered, can make a strong point in a lawsuit. Therefore, non-repudiation services and integrity are the ideal properties which should be adopted.

Component	Threat Category	Description
Global Leader	Spoofing	Impersonation of the global leader
Global Leader	Repudiation	Not traceable addition/removal of an investigator or team leader
(1.0 → 3.1)	Tampering	Adding an unauthorized person as investigator
(1.0 → 3.1)	Denial of service	Adding an investigator is unavailable
(1.0 → 3.3)	Tampering	Adding an unauthorized person as team leader
(1.0 → 3.3)	Denial of service	Adding a team leader is unavailable
Blockchain	Tampering	Modification of data in the blockchain
Blockchain	Denial of service	Blockchain is unavailable
Team Leader Account	Tampering	Modification of the process/logic
Team Leader Account	Repudiation	Non-traceable actions
Team Leader Account	Denial of service	Process is not available
Team Leader Account	Reentrancy	An external call is allowed while the contract is still running, an investigator may end up never been registered
Team Leader Account	Elevation of privilege	Malicious user takes control of the process
Team Leader Account	Short addresses	Special forged addresses produce incorrect arguments being inserted to the blockchain

Table 4.2: Possible threat scenarios for action 1

Reentrancy, along with DoS threats, are overall problematic in smart contracts. Reentrancy threats exploit the call logic in smart contracts, *e.g.* as a contract is running and registering a new account, another external call is made to the running contract. The contract is interrupted and may end up corrupting the account and the smart contract. These threats could potentially lead to wrong transactions and documentation of the actions. DoS is not a new kind of threat, it affects many technologies that are dependent on the availability of their services. As it is, blockchain and smart contracts serve clients. A DoS attack can, depending on the situation, kill the whole contract. Access control goes hand in hand with the spoofing. As mentioned before, the prototype is dependent on the global leader and can only function if he gives the authorization to other accounts. If a malicious user is impersonating the global leader and takes all the control, the smart contract is rendered unusable and must be terminated, possibly forever. Short addresses is a special kind of threat, which involve special crafted addresses. These addresses call

a smart contract and lead to incorrectly processed arguments inserted to the blockchain. This would again lead to faulty transaction and chain of custody, and to less credibility of the prototype.

Component	Threat Category	Description
Investigator	Spoofing	Impersonation of the investigator
Investigator	Repudiation	Not traceable registration/verification of an evidence
Team leader	Spoofing	Impersonation of the team leader
Team leader	Repudiation	Not traceable deletion of an evidence
(4.0 $\rightarrow$ 3.1)	Tampering	Unauthorized registration/verification of an evidence
(4.0 $\rightarrow$ 3.1)	Denial of service	Registration/Verification of evidence is unavailable
Blockchain	Tampering	Modification of data
Blockchain	Repudiation	No traceable transaction data
Blockchain	Denial of service	Blockchain is unavailable
Evidence	Tampering	Evidence data modification
Evidence	Repudiation	Non traceable actions
Evidence	Denial of service	Process is unavailable
Evidence	Reentrancy	An external call is allowed while the contract is still running a verification, the evidence may falsely get verified
Evidence	Access control	Total control of the registration and verification of evidences
Evidence	Short addresses	Special forged addresses produce incorrect arguments

Table 4.3: Possible threat scenarios for action 2

4.2.2 Action 2: Registering, Verifying and Deleting an Evidence

Action 2 associates with the entity (2.0) and (4.0), the process (3.1), the data store (3.2), and the data flow (4.0 $\rightarrow$ 3.1 $\rightarrow$ 4.0) , (2.0 $\rightarrow$ 3.1 $\rightarrow$ 2.0) and (3.1 $\rightarrow$ 3.2). In comparison to action 1, threat scenarios are more likely here. A malicious user may gain more in attacking the components involved with the evidence management. Presumably, these attacks would be motivated by a specific purpose. A person directly connected to an evidence or an investigator connected to the subject of an evidence may try to corrupt the investigation in their favour. These threats would for example include spoofing of investigators or the team leader himself. Especially a team leader, who has high privileges, is susceptible to spoofing threats. Tampering and repudiation attacks could occur when sufficient knowledge about the process exist, which would mean that these threats would likely originate from within the digital forensic team itself.

As seen before, the processing node (3.1) is more susceptible to threats than other DFD components. The attack on this processing node is more likely than an attack on the processing node (3.3) and (3.4) as reasoned before. Reentrancy attack for example could corrupt the processing nodes by allowing a call to the running contract, which was in the process of verification of an evidence. The attack could lead to the contract falsely concluding that the evidence is unmodified and therefore gets verified. Access control can be deadly in combination with spoofing. Elevated privileges, that are granted through spoofing, allow a malicious user total control which ends up killing the contract. Even though the chain of custody is still in the blockchain, evidence data cannot be interacted with, since it resides in the smart contract.

4.2.3 Discussion

The goal of the thesis was to create a PoC and to analyze the feasibility and technicality of extending an existing workflow with blockchain. The assessment of the use case study shows, that the required functionalities work as they are supposed to. The prototype was implemented in Solidity, which is the language for smart contracts in Ethereum. Ethereum is per se a permissionless blockchain, but does allow permissioning in form of smart contracts. Modifiers for example, can impose requirements on the smart contract, that would only allow registered accounts. The main advantage of Ethereum is the ability to locally simulate it with Ganache, and therefore simplify testing. In that sense, requirement (R6) has been fulfilled.

The three different actors (R2) and their respective tasks (R3) have been realised with modifiers, whose responsibility was to restrict functions to the three roles. This functionality was useful as it checks the requirement right when the function is called and only allows authorized accounts to interact with it (R1). Modifiers also made sure that evidence, which have restricted visibility, can only be accessed by accounts belonging to the same team (R5). Solidity is an object oriented language, which made creating structures for the investigator/team leader and the evidence (R4) easy.

Furthermore, the use case study investigated whether the integrity of evidence (R8) and the integrity of the investigators' actions (R9) can be upheld with the implemented solution. The threat analysis indicates that threats are more likely to occur regarding the registration, verification and deletion of evidence. It is assumed that a potential attacker must know the evidence process and is motivated by an agenda. Hence, it is likely that threats occur from within the company. In any other cases, the attacker would not gain much from attacking this contract. Even if a malicious investigator would try to corrupt the forensic process, it would be very difficult to achieve. The features of blockchain, the immutability and the permanent storage make a tamper-free environment that is unlikely to break. Also, the verification mechanism checks the integrity of the evidence, between the verification and its initial registration process. Even after that, any team member can recalculate the SHA-1 on the evidence in the laboratory and compare it to the stored hash in the blockchain. Indeed, once an evidence is hashed and inserted to the blockchain network, integrity check can be done at any time in the CoC. This thesis would argue that because of these points, any accusation of corrupting the evidence by investigators

can be refuted with the blockchain chain of custody. In the end, requirement (R8) and (R9) can be assumed to be accomplished.

Dummy evidences has been created off the chain by using a SHA-1 online encrypter and the EnCase imager. Both took data as input and calculated the SHA-1 key. In case of the online encrypter, the hash needed to be added manually to the evidence file itself or to a separate key file. The EnCase imager automatically included it in the generated E01 file. Requirement (R7) is therefore fulfilled.

The solution is intended to be incorporated as an automated service in the existing workflow. Hence, no graphical user interface has been implemented and command line testing was sufficient. Threats involving external entities will likely be less probable as the interface would not allow human interactions. Especially tampering or repudiation attacks would be more difficult to achieve. Information disclosure threats are a big part of threat modeling, although due to the involvement of blockchain, revelation of sensitive data does not occur. The information inserted into the network is connected to the evidence only and can not be used to track specific people involved.

Reentrancy has not been considered much during the implementation, and may be a big problem as it affects all processing nodes of the DFD. Proper testing, automated as well as manual with focus on reentrancy threats, should be used to achieve a good test coverage. Spoofing in combination with access control is deadly, especially when certain roles have elevated privileges. The best approach would be to include authentication, to assure that only authorized accounts can access restricted functions and data.

Improving the prototype will introduce other threats as well. As a possible part of future work, the evidence structure could have a timestamp that indicates the legal retention time that it must meet. *DeleteEvidence* must check the timestamp before it can delete an evidence from the smart contract, therefore it introduces time manipulation threats.

The big unknown is certainly intimidating. Smart contract are still in their early stages, it will take time for it to mature and be rooted. However, the future will bring more knowledge and tools that help with understanding smart contracts as well as the danger it brings. For now, the smart contract community has to keep an eye open to any new threats and always be ready to adapt to make smart contract more secure.

Chapter 5

Summary and Future Work

As there is an abundance of information and telecommunication processes, generated data may shed light on alleged crimes. Forensic science has the duty to acquire forensic data, preserve and analyze it. Ultimately, their goal is to present evidence to refute or prove an accusation. The large variety of electronic devices and protocols make that task complex. Also due to their fragility, digital evidences are prone to tampering and therefore difficult to collect. Hence, its integrity must be proven at any point during a forensic investigation. A Chain of Custody (CoC) documents the chronological handling of an evidence, and if not well maintained, it might rule the evidence as inadmissible. The goal of this thesis was to analyse an existing digital forensic process in a company, and investigate whether a blockchain approach would improve the integrity, validity and authenticity of digital evidences.

The architecture of the prototype consists of a smart contract, which was deployed to a local Ethereum network as a PoC, and an external server environment. Only crucial metadata representing the evidences are inserted to the blockchain. The evidence itself resides in the confidential external server environment. The prototype expresses allowed accounts to be registered and removed from the smart contract, and once registered, evidence could be registered, verified and deleted. Among the tasks, the verification is the most important action, as it examines the integrity of the evidence. A failed verification indicates a tampering of data between the registration and the verification process of the evidence.

A use case study, consisting of two distinctively different actions, was conducted to evaluate the prototype. The assessment of the study concluded that all functional and non-functional requirements have been sufficiently accomplished. The verification mechanism is able to detect tampering attempts even after the initial verification process. Furthermore, this thesis demanded a threat analysis to assure that the chain of custody cannot be corrupted. In this regard, the STRIDE threat modeling has been chosen whereas the use case study served as context for the analysis. In addition to the six STRIDE threat categories, smart contract threats such as reentrancy, access control and short addresses were also analyzed. The findings suggest that processes of a data flow diagram, therefore, the functionalities of a smart contract, are the most susceptible to threats. Tampering, repudiation, DoS, reentrancy and access control attack appear to have the highest impact

on the system. The PoC was designed in consideration of embedding it in an automated workflow. Hence, no graphical user interface has been implemented. It can be assumed, that attacks caused by humans are more difficult to achieve and do not need to be considered as much. Indeed, this work would argue that because of those reasons the claim of integrity, validity and authenticity can be improved with a blockchain approached CoC.

5.1 Future Work

On a conceptual level, roles could be extended or added to broaden the use of the prototype. Further investigation of the workflow could result in requirements that might improve the CoC even more. Furthermore, smart contract specific threats, such as reentrancy and access control should be considered during the next step of implementation. The original idea is to embed the service in the existing automated workflow, therefore, it is crucial to analyze currently used tools to understand how and where the intended blockchain service could be included. Each device type requires a different acquisition tool, but not all have an active internet connection. Integration of the blockchain service in such tools would be difficult if not impossible.

Bibliography

- [1] <https://dasp.co>. Accessed: 2010-09-30.
- [2] James Tetteh Ami-Narh and Patricia AH Williams. Digital forensics and the legal system: A dilemma of our times. In *Australian Digital Forensics Conference*, page 41, 2008.
- [3] Laurent E Cartier, Saleem H Ali, and Michael S Krzemnicki. Blockchain, chain of custody and trace elements: An overview of tracking and traceability opportunities in the gem industry. *Journal of Gemmology*, 36(3), 2018.
- [4] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. 2018.
- [5] Simson L Garfinkel. Digital forensics research: The next 10 years. *digital investigation*, 7:S64–S73, 2010.
- [6] Praerit Garg and Loren Kohnfelder. The Threat to Our Products. *Microsoft*, pages 1–8, 1999.
- [7] Giuliano Giova. Improving chain of custody in forensic investigation of electronic digital systems. *International Journal of Computer Science and Network Security*, 11(1):1–9, 2011.
- [8] Michael Howard and Steve Lipner. *The security development lifecycle*, volume 8. Microsoft Press Redmond, 2006.
- [9] Xueping Liang, Sachin Shetty, Deepak Tosh, Charles Kamhoua, Kevin Kwiat, and Laurent Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*, pages 468–477. IEEE Press, 2017.
- [10] Auqib Hamid Lone and Roohie Naaz Mir. Forensic-chain: Blockchain based digital forensics chain of custody with poc in hyperledger composer. *Digital Investigation*, 28:44–55, 2019.
- [11] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

- [12] Bruno Rodrigues, Thomas Bocek, and Burkhard Stiller. The use of blockchains: application-driven analysis of applicability. In *Advances in Computers*, volume 111, pages 163–198. Elsevier, 2018.
- [13] Bruno Rodrigues, Eder John Scheid, Roman Blum, Thomas Bocek, and Burkhard Stiller. Tutorial 1. blockchain and smart contracts - from theory to practice. In *IEEE International Conference on Blockchain and Cryptocurrency*.
- [14] Jung Hyun Ryu, Pradip Kumar Sharma, Jeong Hoon Jo, and Jong Hyuk Park. A blockchain-based decentralized efficient investigation framework for iot digital forensics. *The Journal of Supercomputing*, pages 1–16, 2019.
- [15] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. A descriptive study of microsoft’s threat modeling technique. *Requirements Engineering*, 20(2):163–180, 2015.
- [16] Zhihong Tian, Mohan Li, Meikang Qiu, Yanbin Sun, and Shen Su. Block-def: A secure digital evidence framework using blockchain. *Information Sciences*, 491:151–165, 2019.
- [17] Tony UcedaVelez and Marco M Morana. *Risk centric threat modeling*. Wiley Online Library, 2015.
- [18] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE, 2018.
- [19] Kim Wuyts, Riccardo Scandariato, Wouter Joosen, Mina Deng, and Bart Preneel. LINDDUN: A Privacy Threat Analysis Framework. *DistriNet*, pages 1–23, 2019.
- [20] Yunus Yusoff, Roslan Ismail, and Zainuddin Hassan. Common phases of computer forensics investigation models. *International Journal of Computer Science & Information Technology*, 3(3):17–31, 2011.
- [21] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *Work Pap.-2016*, 2016.
- [22] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564. IEEE, 2017.

Abbreviations

CoC	Chain of Custody
DB	Database
DFD	Data Flow Diagram
DPoS	Delegated Proof of Stake
EVM	Ethereum Virtual Machine
IoT	Internet of Things
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoC	Proof of Concept
PoS	Proof of Stake
PoW	Proof of Work
RDF	Resource Description Framework
SC	Smart Contract

List of Figures

2.1	Example of blocks in a blockchain [21]	3
2.2	Types of blockchain [13]	5
2.3	A DFD for a digital publishing system. a) Level 0 (context diagram) b) Level 1 [15]	9
3.1	Workflow for the evidence and case management	13
3.2	Use case diagram	16
3.3	Entity relationship model	17
3.4	Evidence process	17
3.5	Sequence diagram	18
4.1	Team Europe: Shows the address ('0'), a boolean indicating a team leader position ('1') and the team ('2')	24
4.2	Log: Removed an investigator from team Europe	24
4.3	The registered evidence "de1"	25
4.4	The verified evidence "de1"	25
4.5	Log: deleted evidence "de1"	26
4.6	Level 0 (I) and Level 1 (II) Data Flow Diagram	27

List of Tables

2.1	Comparison between blockchain and traditional databases [13]	4
2.2	DFD elements mapped to STRIDE [8]	9
4.1	Threats to the prototype	28
4.2	Possible threat scenarios for action 1	29
4.3	Possible threat scenarios for action 2	30

Appendix A

Installation Guidelines

The installation guidelines show the steps to successfully run the FCC prototype on a local machine with a macOS. A UNIX based machine would follow a similar installation pattern, the steps can differ on a Windows machine. The source code of the prototype can be found on the enclosed CD.

A.1 Setting up Ganache

To set up Ganache on the local machine, it can be downloaded and installed from the trufflesuite website.

```
https://www.trufflesuite.com/ganache
```

Clicking "Quickstart" button sets up a fresh network with ten accounts, each with 100 ether.

A.2 Install Solidity

Solidity can be installed with the node package manager. Enter this command in your command line interface (CLI) of your choice.

```
npm install -g solc
```

A.3 Install Truffle

Truffle can also be installed with the node package manager. Enter following command in your CLI of your choice.

```
npm install truffle -g
```

A.4 Setting up Prototype

To start, navigate to the prototype root folder in your CLI.

```
cd FCC
```

To run the prototype, the smart contract has to be compiled and migrated to Ganache.

```
truffle compile  
truffle migrate --reset
```

The smart contract may not be able to connect to Ganache as the port and the network ID might differ from the project configuration. They can be found under the settings of Ganache. The port and the network ID must be replaced in the following file:

```
cd FCC/truffle-config.js
```

The migration was successful, if the contract is displayed as deployed in the contract tab of Ganache.

To start interacting with the deployed contract, the truffle console must be opened first. Following command opens the console.

```
truffle console
```

Initialize the contract to access the functions.

```
let fcc = await fcc.deployed()  
bcc.assignTeamLeader("0xAC83120d09F05C7d861Aca378dFFeF48c2103CeC", 2)
```

Appendix B

Contents of the CD

- FCC Prototype (Source Code)
- LaTeX Source Code of Bachelor Thesis Document
- Bachelor Thesis Document (.pdf)
- Abstract English (.txt)
- Abstract German (.txt)
- Intermediate Presentation (.pdf)
- Final Presentation (.pdf)