TU DELFT

MASTER THESIS

MediTrail: a blockchain-based tamper-proof auditable access log for medical data

Author:
Angela PLOMP

Supervisor:
Dr. ir. Johan POUWELSE

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Distributed Systems Group
Computer Science

February 20, 2019

TU DELFT

Abstract

EEMCS
Computer Science

Master of Science

MediTrail: a blockchain-based tamper-proof auditable access log for medical data

by Angela PLOMP

After several incident in which the privacy of patients was violated, the need for logging of access to medical data is evident. In this master thesis, the first prototype is presented for a blockchain-based access log for medical data. This prototype uses a novel blockchain to create a tamper-proof log. Patients each have their own blockchain in which health care providers can participate as well. At any time, a user can check the access logs of their data on the online *MediTrail* portal. Users can also validate an entry by placing a digital signature. The prototype performs well when no consensus algorithm is used. Achieving consensus with Proof of Elapsed Time slows adds overhead. The prototype succeeds in providing the patient with knowledge and power over their data in the form of a tamper-proof auditable access log.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. ir. J.A. Pouwelse for his enthousiastic personality and his innovative spirit. Thank you also to Dr. J. Rellermeyer and Dr. ir. Van den Akker for the valuable feedback.

My gratitude goes out to my parents, Maria and Arie, who were always generous in their support, and encouraged me to pursue whatever goal I had in mind. Thank you for believing in my abilities even when I did not.

Thank you to my boyfriend, Elmer, for bearing with me with incredible patience and keeping me focused on all the good things that will come. It was not the easiest time and you dragged me through it.

I would also like to thank my friend Martin van Hensbergen for being almost as concerned with my thesis as I was myself, and simultaneously performing the roles of Python master, math oracle and positivity coach.

Contents

Chapter 1

Introduction

1.1 The leading use case

In January 2018, Dutch reality star Samantha "Barbie" de Jong received acute medical care in the Haga Hospital in The Hague (De Telegraaf 2018). Her hospitalization was met with great interest from several media companies, who speculated about possible causes. A few weeks later, it turned out that an abnormally large number of Haga Hospital employees had looked into one partical patient's files: Ms. De Jong's. In order to look into the files, they even ignored a warning screen. All doctors and nurses have access to the electronic medical record of all patients currently or previously hospitalized. Haga Hospital apologized for the incident and started an investigation, sending an official warning to the illegitimate readers of the records. This news resparked a debate about the merits and risks of storing medical data the way we do.

1.2 Digitalized medical records

Medical records have existed for ages and are now not only useful for individual patient tracking, but for research on populations as well.

1.2.1 History of medical records

One of the oldest medical documentations found is an Egyptian papyrus, dating from 1600 BC. It contains a didactic recording of a surgery (Gillum 2013). Later, in Ancient Greece, Hippocrates (460-370 BC), one of the most famous doctors of world history made a big contribution to medical records. He documented many medical case studies, notes and philosophical ponderings, bundled in the Hippocratic Corpus. As the interest in natural science in general and human anatomy in particular rose during the 17th and 18th century in the Western world, more and more records were kept on the suspected origin and possible treatment of diseases. Still, these records were kept for educational purposes and not to track individual patients' health trajectory. This only started to change in the 20th century. Most governments of european countries started requiring physicians to keep records on their patients in a specific format. With the rise of (affordable) personal computers in the 1960s, medical records moved towards digital files that can be shared between health care providers such as GP's, hospitals and specialized clinics.

1.2.2 Data ownership

A central question with regard to medical information is ownership of the data. Many patients feel that they do not control access to their data, but would like to be able to access the data themselves, look at the history of data access and give or deny access permissions to healthcare providers (World Economic Forum 2012). The data is about them, so they feel they should have ultimate control over it. Patients that doubts the confidentiality of their records may not make completely honest disclosures, holding back potentially crucial information. Without complete and accurate information, health care providers may misdiagnose the patient or provide inadequate treatment. On the other hand, the data has been collected and stored by the healthcare providers. They invest time and money into this process. The burden of coming up with policies and implementation of these policies also lies on the health care provider (Kostkova et al. 2016).

1.3 Cyber crime and other concerns

As in any digital information system, providing adequate protection of the data is a serious concern. Medical data have qualities that make them particularly attractive for cyber criminals and fraudsters.

1.3.1 Data theft or leakage

EMRs may contain extremely sensitive data: It is assumed that most people would not want others to know if they suffer from stigmatized illnesses like sexually transmittable diseases or mental disorders. In a more practical way, information about someone's medical history may for example have a negative effect on their chances of being hired for a job. In some parts of the world, medical identity theft is a problem, which is a special case of identity theft. This is when a person uses another person's identity to fraudulently receive health care or prescription drugs. According to a study on medical identity theft from 2016, the last years showed an upward trend in the number of medical identity theft cases in the USA. The main causes for this identity theft are the stealing or abusing of credentials of family members, a data breach at a healthcare provider or the submission of credentials on a phishing page (Ponemon Institute 2016).

1.3.2 Privacy concerns around EMRs

The United Kingdom launched NHS Care.Data in 2013, an initiative to centralize patient health care data. Patient information could be legally shared with stakeholders outside of the NHS or medical research community. A report found multiple severe problems with this system in terms of privacy and patients' power over their own data (Presser et al. 2015). Data was processed without properly consulting or even informing patients. Sometimes, data was optimistically categorized as anonymous or pseudonymous even though there exist known techniques to deduce personal information from it (. Li, T. Li, and Venkatasubramanian 2007). GPs were required to send records to the central database, but were simultaneously required by another law to keep the records confidential, which led to legal complications. Another system by the NHS, the Detailed Record System, was classified by researchers as "almost certainly illegal under human rights or data protection law" (Anderson et al. 2009).

1.3.3 Impact of the GDPR

In May 2018, the General Data Protection Regulation (GDPR) came into effect in the European Union, as a replacement of the Data Protection Directive (DPD) of 1995. The DPD already forced EU member states to take into account data protection on computers and other electronic devices (Calder 2016). The GDPR presents six principles that should be adhered to when collecting, storing and processing data. These mainly concern the proportionality of the data gathering for a certain goal and transparency of and consent for the use of the data. Organisations are held responsible for proving that they comply with the rules. The GDPR is not specifically designed for medical data. There may exist conflicting objectives when it comes to ensuring privacy rights versus providing adequate access to data (European Society of Radiology 2017). The GDPR requires healthcare providers to grant patients access to their files, as long as the access requests are not 'manifestly unfounded or excessive'. The Regulation provides several exemptions and derogations for the use of health data, if applying the law would prevent or seriously impair research (McCall 2018).

Chapter 2

Problem statement

The central problem of this thesis is to present a new method of logging all access to medical files in a tamper-proof way, focusing on the principles of non-repudiation and patient power. The patient has access to all data in our model.

2.1 Barbie's medical records in HiX

The hospital where Ms. De Jong was treated for her medical problems, Haga Hospital, uses ChipSoft's HiX software for the storage and processing of patient's medical records (HagaZiekenhuis 2016). HiX does record the access of users to the digital files. During a routine check, the access to Ms. De Jong records by staff who were not treating her came to light. This violation of her privacy is deemed unacceptable by many people. Physicist and philosopher Vincent Icke attributes the incident not only to the snooping employees, but to the design of EMR systems as well. He proposed that an app should be made to let patients see the access log of their medical data (Icke 2018). In this research, my hope is to contribute to the development of a more secure medical record file system in which the patient's involvement is central. Ms. De Jong should have easy access to the log of persons who viewed the record herself. Additionally, she should know the exact contents of these records and agree with their storage.

2.2 Research goal

The goal of this master thesis project is to research the possibilities of expanding patients' power over and knowledge about their medical records. This power consists of two parts:

1. Accountability on access: knowing who has accessed the file;
2. Validation of EMR entries by both the health care provider and the patient.

In addition to this, the traditional security goals for any information system containing sensitive data still stand: confidentiality, integrity and availability. In the earlier days of medical record systems, there was a lack of clear security policies for these kinds of systems, as a consequence of little awareness of the ethical and legal duties for medical data protection. Anderson (1996) presented a security policy model for clinical information systems, consisting of nine principles. In the next paragraph, the relevance of these principles and other frameworks for accountability on access and validation of entries in medical systems will be explored.

2.2.1 Accountability on access

In 2007, Scotland dealt with a very similar case to De Jong's when over 50 employees of an NHS hospital illicitly looked into a celebrity's medical record (Carvel 2017). This scandal occurred just before upgrading the medical file systems to a new and controversial version, in which data would be accessible for some companies as well. However, an NHS spokesperson stated something very interesting: *"The reality of the situation is that, for the first time in the history of medical records, the new IT systems being implemented across the NHS have a fully integrated audit trail that tracks access to any care record to safeguard and maximise patient confidentiality."* The fact of the matter is that the new system which provided the audit trail, made it possible to hold the health care providers accountable for their privacy invasion. Accountability on access means that a patient can verify who has accessed accessed a file, and when. There should be no way for someone to access the file without leaving a trace. When a patient questions the legitimacy of an access event, the person who looked into the file can be asked for an explanation. One of the aforementioned Anderson's nine principles is stated as follows: *"All access to clinical records shall be marked on the record with the subject's name, as well as the date and time. A audit trail must also be kept of all deletions"* (R. J. Anderson 1996). A recent paper that points out the lack of patient-centered transparency requirements for medical data systems, states that transparency is needed for accountability. The authors define ex-post transparency as *"enabling the patient to be informed or get informed about what happened to his/her medical and personal data"* (Spagnuelo and Lenzini 2016). In order to fulfill this ex-post transparency goal, a number of transparency requirements were formulated. When it comes to the relation between transparency and accountability, the most relevant of these requirements are:

1. The medical record system must provide the patient with accountability mechanisms.
2. The medical record system must provide the patient with evidence regarding permissions history for auditing purposes.
3. The medical record system must provide the patient with evidence of security breaches.

These requirements guide the design of an EMR system that center the patient's need of privacy and power over their own data. Thus, these criteria will be used in the set up of the requirements for the system presented in this thesis.

2.2.2 Validation of EMR entries

According to University of Leeds researchers, an EMR is valid if all events have been recorded and all records signify an event. Additionally, it should be clear what every record means (Neal, Heywood, and Morley 1996). Later, researchers have extended this definition to: *"Medical records, whether paper or electronic, record health events. Records are valid when all those events that constitute a medical record are correctly recorded and all the entries in the record truly signify an event"* (Hassey, Gerrett, and Wilson 2001). In this master thesis, validation of EMR entries means that an entry becomes official only when both the patient and the health care provider have agreed to the entry. This is similar to a person sending a registered letter and the recipient signing for delivery. The patient cannot claim not to know the content of the entry. Research found out that there are significant discrepancies between health care reported by physicians themselves, patient surveys, and written medical records (Stange et al. 1998). Another interpretation of the concept of validation of EMR entries is to verify whether the content of the records, e.g. lab results, are actually accurate. This is not related to patient power over data and therefore out of scope for this research.

2.3 Research question

Taking the aforementioned considerations into account, the research question for this thesis project is as follows:
R: *"How can blockchain technology be used to design an Electronic Medical Record (EMR) system, that guarantees accountability on access and validation on entry addition?'*

This question can be split into two subquestions:
R1: *"How can blockchain technology be used to guarantee accountability on access in an EMR?"*
R2: *"How can blockchain technology be used to validate entries in an EMR?"*

When the two subquestions are answered, the main research question can be answered as well. The proposed solution will be supported by a simple prototype as a proof-of-concept.

2.4 Requirements

Before making a design, it should be clear what the requirements are. These are used for both the design of the system and the validation after building the prototype. Some requirements are general, others are specifically needed for answering the research questions.

2.4.1 Requirements for accountability and validation

The proposed system should fulfill the following requirements directly related to the research questions:

1. Accountability on access: Every access to an entry in the EMR system is recorded. The log contains information on the name of the user who accessed the file, the name of the file itself, and the timestamp of the event.

2. Validation of entries: A user should be able to sign an entry with a secure digital signature. The digital signatures should be verifiable by anyone in the system.

2.4.2 Requirements from the CIA triad

A standard in the field of information security is the CIA triad. This stands for the security goals of confidentiality, integrity and availability that any secure information system should meet. Based on these goals, the following additional requirements are constructed:

1. Confidentiality: Information stored in the EMR system itself as well as the event log should only be accessible to the users it is intended for.

2. Integrity: Information stored in the EMR system cannot be changed by and adversary without being noticed.
3. Availability: Information stored in the EMR system is available for the users whenever they need or want to access it.

2.4.3 Requirements for the user experience

The prototype for the proposed system is not intended as a ready-to-use system for the real world. The user experience has a low priority as it is not really needed to demonstrate the qualities of the system for its intended goal. However, there are some minimal requirements:

1. The user should be able to navigate between the functionalities of the system without effort;
2. The information displayed to the user should be clear and easily understandable;
3. The user should be able to easily verify that the access log has not been tampered with.

2.5 Research method

First of all, a literature study is conducted on the topic of EMRs and the state-of-the-art of blockchain-based medical systems. The focus lies on the use of blockchains to improve patient's power and knowledge over their data. Then, possible architecture and implementation choices for a system that fills the requirements as stated in this chapter will be explored. Two aspects are taken into account. First, the desired functionality and ideas found in previous work by researchers that touch upon this subject. Second, the technologies should be available in practice. NASA distinguishes 9 Technology Readiness Levels (TRLs), ranging from the observation of basic principles to a system being successfully used in operations (Mankins 1995). Recent developments in computer science are sometimes only described in theory or as a proof of concept, and are still in a low TRL and therefore not ready to use in the system. After analyzing the architecture and implementation options, the prototype will be made. When the prototype is tested, it will be validated by checking it against the requirements stated in this chapter.

Chapter 3

Background

This chapter gives an introduction to the theory that is needed to understand literature on the related work.

3.1 Introduction to blockchain

Blockchain is a relatively new technology, celebrating its 10th birthday this year. It is best known as the driving force behind cryptocurrencies as Bitcoin, being the first implementation of blockchain in 2008, and Ethereum. At the core of a blockchain, there is a distributed ledger that is tamper-proof under the right circumstances. Essentially, blockchain is a peer-to-peer distributed ledger, which can only be updated via consensus (Nakamoto 2008). Blockchains can be public, private or semi-private. Anyone can participate in a public (or permissionless) blockchain: all participants hold a copy of the ledger but none of the participants actually own the ledger. This ensures the decentralized nature of the blockchain. A private blockchain is open only to an organization or consortium. Semi-private blockchains are a combination of a public and private part (Bashir 2017).

3.1.1 Blocks

As the name implies, a blockchain is in essence a chain of blocks. A block minimally consists of:

1. The hash of the previous block;
2. A nonce (number used only once);
3. A bundle of transactions.

The first block in a blockchain is called the genesis block. This is hardcoded at the time the blockchain was started. To add a block to the blockchain, the nodes must agree on a single version of truth. This is achieved using a consensus algorithm.

3.1.2 Tamper-proof qualities of blockchains

The longer a block has been in the blockchain, the more permanent its status is. The probability of another version of the blockchain becoming the largest chain, not containing this particular block, becomes smaller and smaller as the chain grows longer. Because every block contains a hash pointer to the previous block, one can access the previous information, but also verify that it has not changed. Tampering is evident because the hash of the changed information would change, too. A binary tree with hash pointers is called a Merkle tree. An essential quality of a Merkle tree is that it can hold many items, but one just needs to remember the root hash in order to verify membership of the tree in just *O(log n)* time and space (Szydlo 2004). Although data can be stored in a blockchain directly, a blockchain is not suitable to store large amounts of data. This is why many blockchain-based systems use a distributed hash table (DHT) that only stores pointers to the actual data.

3.2 Consensus algorithms

The goal of a consensus algorithm is to achieve consensus between honest nodes. Consensus algorithms are used in all kinds of distributed systems. In the case of blockchain systems, the need for consensus is centered around the question of which blocks should be added to the chain. There are roughly two categories of consensus mechanisms: Byzantine fault tolerance-based or proof- and leader-based algorithms.

3.2.1 Byzantine Generals Problem

The need for consensus can be illustrated with the classic Byzantine Generals Problem (Lamport, Shostak, and Pease 1982). This problem describes a war situation, in which a group of generals must agree on whether to attack the enemy or to retreat. If some generals attack but others retreat, the consequences will be poor, especially for the attacking generals. To ensure an agreement, one commanding general must send an order to the other (lieutenant) generals, such that:

1. All loyal lieutenant generals obey the same order;
2. If the commanding general is loyal, all loyal lieutenant generals obey his order.

Note that the problem includes the possibility that the commander general is not loyal himself, and that the goal is not to reach a specific outcome but merely to have all generals agree to the outcome. It turns out that if there are m malicious generals and more than $3m$ honest generals, the loyal generals can reach a consensus applying the following algorithm:

1. The commander sends his value to the lieutenants: either *attack* or *retreat*.
2. Each lieutenant adapts the value he received from the commander. If he did not receive a value, he adapts the default value *retreat*. Each lieutenant acts now sends his value to the remaining lieutenants.
3. Each lieutenant chooses the majority value of the values he received from the commander and the other lieutenants. If there is a tie, he falls back on the default value *retreat*.

The Byzantine generals problem is of great importance in distributed systems. When applied to blockchains, the nodes can be seen as generals who have to agree on whether they should add a new block to the chain or not. In a public blockchain, an attacker could create as many nodes as possible to make sure that there are $3m$ or fewer honest nodes. The above algorithm would not be correct anymore. This is called a Sybil attack.

3.2.2 Byzantine Fault Tolerant protocols

Byzantine fault tolerance means that a system has a mechanism in place to overcome failures or malicious nodes like described in the Byzantine Generals Problem.

Practical BFT

Almost two decades ago, researchers presented Practical BFT (PBFT) (Castro, Liskov, et al. 1999). It was a major improvement upon earlier protocols, in the sense that it previous protocols assumed synchrony. The protocol tolerates a number of malicious or faulty nodes f that is strictly fewer than $1/3$ of the total number of nodes n, such that $N \leq 3f + 1$. PBFT is a weakly synchronous protocol, which relies on some timing assumptions. Liveness is only guaranteed when the network behaves as expected. The algorithm works with a leader node which gathers votes and broadcasts the result to client nodes. In a nutshell, the algorithm works as follows:

1. A client node sends a request to the leader node;
2. Pre-prepare phase: The leader node assigns a sequence number to the request and broadcasts this to all other nodes (replicas);
3. Prepare phase: The replica nodes acknowledge this sequence number (as long as it is valid) by sending a prepare message to all other nodes;
4. Commit phase: The client waits for $f + 1$ replies from different replicas with the same result.

When $f + 1$ nodes have voted for a certain result, this is accepted as the result of the operation.

HoneyBadger BFT

HoneyBadger BFT is the first practical asynchronous BFT protocol that guarantees liveness without making any timing assumptions (Miller et al. 2016). Earlier BFT systems assume weak synchrony, in the sense that every message is is guaranteed to be delivered with a maximum delay Δ. HoneyBadger BFT however, does not care about the underlying network. It is even suitable for an asynchronous setting as long as each pair of nodes is connected by an authenticated point-to-point channel, that does not drop messages. Additionally, nodes may interact with a trusted dealer during the protocol-specific setup phase. Just like in the PBFT protocol, it is necessary that the $3f + 1 \leq N$ threshold is upheld.

3.2.3 Proof-of-work-based consensus algorithms

Bitcoin uses the proof-of work consensus mechanism to prove that enough computational resources have been spent in order to be trusted to propose an addition to the blockchain. Nodes can compete with each other to be selected in proportion to their computing capacity. For Bitcoin, the proof-of-work requirement is to solve the following problem (Bashir 2017):

H (N | | P_{hash} | | Tx | | Tx | | . . . Tx) < **target** where
H is an ideal hash function,
N represents a nonce,
P_{hash} is the hash value of the previous block, and
Tx are the transactions in the proposed block.
The hash value of these concatenated fields should be smaller than the set **target** for difficulty.

An ideal hash function h satisfies three requirements (Paar and Pelzl 2009):

1. Preimage resistance: given a hash output z, it must be computationally infeasible to find an input message x such that $z = h(x)$
2. Second preimage resistance: it must be computationally infeasible to create two different messages $x_1 \neq x_2$ with equal hash values $z_1 = h(x_1) = h(x_2) = z_2$
3. Collision resistance: it should be computationally infeasible to find two different inputs $x_1 \neq x_2$ with $h(x_1) = h(x_2)$

If H is an ideal hash function, it should be computationally impossible to construct a hash output that satisfies the target as set in the proof-of-work problem. Solving the problem is therefore done in a brute-force way, letting nodes sacrifice CPU power in exchange for trust. The high costs of creating malicious pseudonymous identities prevents Sybil attacks (Vukolić 2015). A drawback is that it is (obviously) computationally intensive, and therefore uses much energy, which is a strain on the environment. The consumption is around 500 MW at the moment, and researchers suspect that it may grow even much larger in the coming few years (Fairley 2017).
Adding a block to the blockchain is done through the following consensus algorithm (Nakamoto 2008): new transactions are broadcast to all nodes; each node collects transactions into a block; in each round, a random node (selected by the proof-of-work) gets to broadcast its block; other nodes accept the block if and only if all transactions in it are valid; nodes express their acceptance of the block by including its hash in the next block they create.

3.2.4 Proof-of-elapsed-time

The aforementioned proof-of-work based consensus algorithm relies on a kind of lottery, in which having a large amount of CPU power increases the chances of winning. A similar thought is the basis of proof-of-elapsed-time algorithms (Intel Corporation 2017). The strategy is as follows: Each node chooses a random time to sleep. The first node to wake up, may propose the next block to be added. There are two fundamental requirements in order for this to work. The first one is that the time to sleep for all nodes are indeed randomly chosen. The second requirement is that all nodes respected this chosen sleeping time. In theory, PoET is designed for an abstract TEE (trusted execution environment), so it is flexible for multiple TEE implementations. Intel has developed chips equipped with SGX (Software Guard Extensions) instructions which provide an attestation that these requirements are actually fulfilled. Equipping a blockchain with trusted computing technology has two major advantages with respect to other PoW consensus mechanisms. One is the high efficiency, nodes do not have to solve NP-complete computational problems like in Proof of Work schemes. Another is the fairness: it achieves the "one CPU, one vote" principle better than Proof of Work schemes.

3.2.5 Other types of consensus algorithms

The proof-of-stake algorithm uses the stake that a user has in the system, for example invested time, to trust that the benefits of performing malicious activities would not outweigh the benefits of staying in the system as a trusted member (Kiayias et al. 2017).
Deposit-based consensus requires putting in a deposit before proposing a block to be added to the blockchain. In case the block is rejected by others, the user loses its

deposit (Solat 2017). Reputation-based mechanisms let members elect a leader node, based on the reputation it has built on the network. When a transaction is added to a block, it should be clear who has performed this transaction.

3.3 Identity and verification

Particularly in the medical use case, any access to the EMR should be linked to an identity. A digital signature confirms the identity, under the condition that such a signature can be verified but cannot be forged. Digital signatures can be issued using different algorithms. Bitcoin uses the Elliptic Curve Digital Signature Algorithm (ECDSA).

3.3.1 Identities and signatures

Accountability on access can only be established when it is guaranteed that the person being recorded as accessing or modifying the file is indeed the person who is doing so. An identity should have a one-on-one relation to a person.

3.3.2 Identification of users

This means that the proposed system will need a solid identification and authentication method for the file system. Traditionally, this goal has been attained by using username/password systems. There are several drawbacks to this system, especially as the number of username/password combinations users have to memorize grows larger. Many users consider this as very inconvenient and this sometimes leads to irresponsible password behaviour (Adams and Sasse 1999). Another issue is that a user has to create a new identity for each application. These identities only exist within the context of each specific website or application, leading to great volumes of data duplication (Tobin and Reed 2016).

3.3.3 Digital signatures

As paperwork has been replaced by digital entries, digital signatures have taken over the role of traditional signatures. A digital signature provides proof of the integrity of the authorship, because anyone can verify that the signature is based on the author's public key. On the other hand, only the person who creates the message should be able to generate a valid signature. In general, the steps to create a digital signature are as follows:

1. The signature algorithm is a function of the signer's private key k_{pr}. Hence, only one person can sign a message x, assuming that the private keys are kept secret.

2. The message x is an input to the signature algorithm as well, to make sure that the signature is related to the message and cannot be re-used.

3. A digital signature algorithm is run with the right inputs, which yields signature s. Then, s is appended to x and the pair (x, s) can be sent.

Digital signatures can be created using a range of different algorithms, based on for example Digital Signature Algorithm (DSA), prime factorization (RSA-based signatures) or the discrete logarithm problem (ElGamal-based signatures) or on the elliptic curve discrete logarithm problem.

3.3.4 Elliptic Curve Digital Signature Algorithm

Elliptic curves have some advantages over RSA and discrete logarithm-based schemes. Threshold versions of DSA are unusable in practice (R. Gennaro, Goldfeder, and Narayanan 2016). One of these advantages is that a small key length provides the same security as other schemes, but with a shorter processing time. The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined over prime fields as well as over Galois fields. Here, the procedures for the more popular version over prime fields are given (Paar and Pelzl 2009).

1. For key generation, an elliptic curve E is chosen with modulus p, coefficients a and b and a point A which generates a cyclic group of prime order q. Choose a random integer d such that $0 < d < q$. Compute the new point $B = dA$.
   $k_{pub} = (p, a, b, q, A, B)$
   $k_{pr} = (d)$

2. In order to generate a signature, an integer such that $0 < k_E < q$ is chosen as an ephemeral key. Compute $R = k_E A$. Let $r = x_R$ (the x-coordinate of point R) and compute the signature $s \equiv (h(x) + d \cdot r)k_E^{-1} \bmod q$.

The main analytical attack against ECDSA, assuming that the parameters are chosen correctly, is trying to solve the elliptic curve discrete logarithm problem. Considering that this is an NP-complete problem, it is extremely unrealistic to solve this in time.

3.3.5 Elliptic curve threshold signatures

Similarly to the threshold encryption schemes discussed before, threshold cryptography can be applied to digital signatures. A scheme to achieve this was first presented in 1992 by Desmedt & Frankel. This method was based on the RSA signature scheme (Desmedt and Frankel 1991). Since then, many papers have been published presenting threshold signature schemes. One of them was a robust Elliptic Curve threshold DSA scheme (. Gennaro et al. 1996). For this project, the focus will be on Elliptic Curve threshold signature schemes, because of the previously mentioned advantages. Specifically, a scheme is needed which is fit to execute on a distributed system.

3.3.6 Threshold ECDSA in a fully distributed system

In 2015, researchers at the Worcester Polytechnic institute presented a fully distributed signature system for threshold ECDSA, named *Nephele* (Green and Eisenbarth 2015). This system is mainly built to protect the key from side-channel attacks and is designed in such a way that a private key never even needs to appear in memory. The key generation as well as the signature generation algorithm is fully distributed. It also allows for fully distributed key re-sharing.

3.3.7 Identity-based signatures

Considering the wish for transition to self-sovereign identities as explained in paragraph 3.3.1, the possibility of using identity-based signatures should be researched. Because the core goal of this project is to design a system with patient's power in mind, it would be fitting if patients do not have to rely on an external party to provide their identification. The idea of identity-based signatures is a public key cryptosystem in which the users do not have to exchange public keys because the public key of a user is simply a person's email address or other personal identification (Shamir 1984). Requirements for this identification is that it uniquely identifies the user in a way that cannot be denied afterwards, and that the information is available to anyone within the system. A trusted party computes the private key for every user and issues the keys on a smart card.

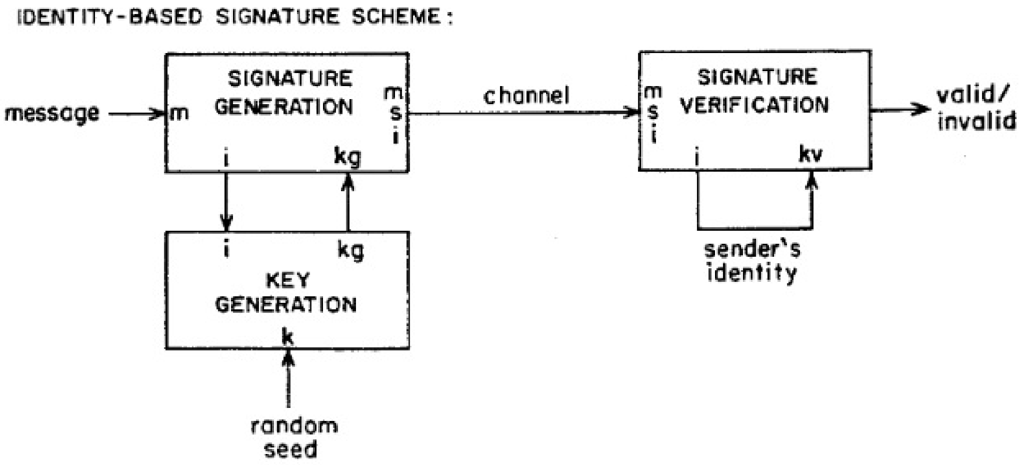


FIGURE 3.1: Identity-based signature scheme (Shamir 1984)

3.3.8 Schnorr signatures

Since a few years, some Bitcoin enthusiasts have been lobbying for the usage of Schnorr signatures to sign transactions. The Schnorr signature scheme is an elegant signature scheme based on the discrete logarithm problem. It is efficient because it minimizes the message-dependent computation needed to generate a signature (Schnorr 1991). Most of the computational effort can be done in a pre-processing stage whenever is convenient for the processor. One of the major challenges for blockchains in general is scalability. Consider the scenario that a user would like to send a certain amount of bitcoins from multiple accounts to one account. In the current system, the transaction from each source account to the destination account requires its own signature. However, if it is just one user sending the transaction, they should be able to place just one signature for the combined transactions. Schnorr signatures enable users to do this. Cutting superfluous signatures could potentially achieve a significant reduction in bandwidth, which in turn makes up space for more transactions: increasing scalability.

Chapter 4

Related work

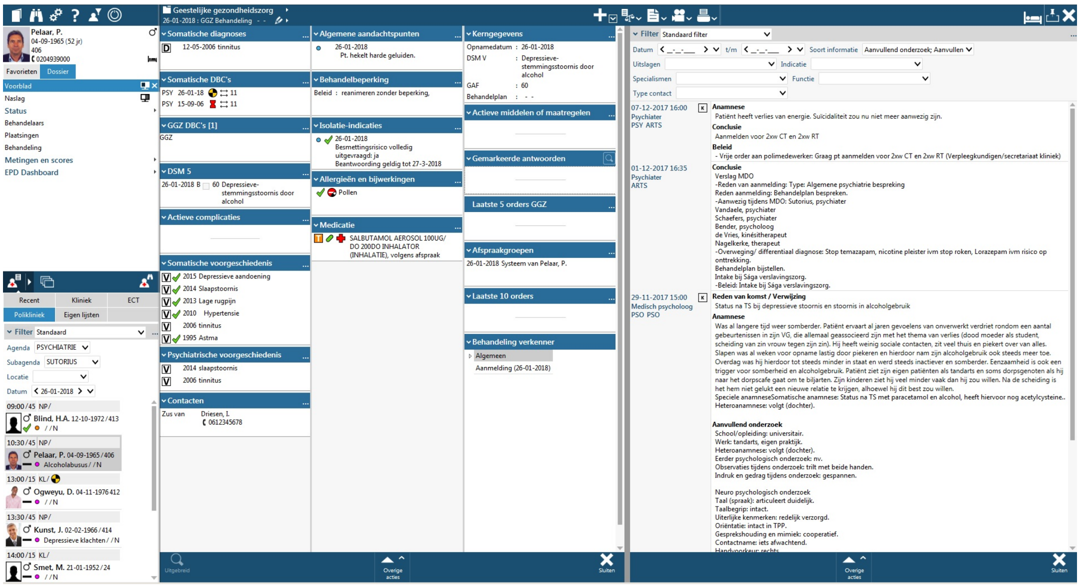
Almost every hospital in the world uses an EMR system to handle patient data. A considerable amount of research has been conducted to study possible improvements on the protection of privacy in these systems.

4.1 Access logging in current widely-used EMR systems

The need to have an auditable access trail for EMR systems is not new, and current systems already have some kind of access logging. In The Netherlands, the EMR market is dominated by two parties: Chipsoft and Epic.

4.1.1 Chipsoft

The hospital where Ms. De Jong received care uses Chipsoft's HiX (Healthcare information eXchange). Chipsoft is a dutch EMR developer and delivers several versions of their Microsoft-based HiX software. On their website, there are no implementation details about some form of logging. An email requesting information about their implementation and handling of logging never received a response. It would be safe to assume that Chipsoft does not use blockchain technology for access logging yet.



Pelaar, P.
04-09-1965 (52 jr)
406
0204939000

Filter: Standaard
Agenda: PSYCHIATRIE
Subagenda: SUTORIUS
Locatie:
Datum: 26-01-2018

09:00 / 45 NP/
Blind, H.A. 12-10-1972 / 413
/ / N

10:30 / 45 NP/
Pelaar, P. 04-09-1965 / 406
Alcoholabusus / / N

13:00 / 15 KL/
Ogweyu, D. 04-11-1976 412
/ / N

13:30 / 45 NP/
Kunst, J. 02-02-1966 / 414
Depressieve klachten / / N

14:00 / 15 KL/
Smet, M. 21-01-1952 / 24
/ / N

Geestelijke gezondheidszorg
26-01-2018 : GGZ Behandeling - -

Somatische diagnoses
D 12-05-2006 tinnitus

Somatische DBC's
PSY 26-01-18 11
PSY 15-09-06 11

GGZ DBC's [1]
GGZ

DSM 5
26-01-2018 B 60 Depressieve-stemmingsstoornis door alcohol

Actieve complicaties

Somatische voorgeschiedenis
V ✓ 2015 Depressieve aandoening
V ✓ 2014 Slaapstoornis
V ✓ 2013 Lage rugpijn
V ✓ 2010 Hypertensie
V 2006 tinnitus
V ✓ 1995 Astma

Psychiatrische voorgeschiedenis
V 2014 slaapstoornis
V 2006 tinnitus

Contacten
Zus van Driesen, I.
0612345678

Algemene aandachtspunten
26-01-2018
Pt. hekelt harde geluiden.

Behandelbeperking
Beleid : reanimeren zonder beperking,

Isolatie-indicaties
✓ 26-01-2018
Besmettingsrisico volledig uitgevraagd: ja
Beantwoording geldig tot 27-3-2018

Allergieën en bijwerkingen
✓ Pollen

Medicatie
T SALBUTAMOL AEROSOL 100UG/DO 200DO INHALATOR (INHALATIE), volgens afspraak

Kerngegevens
Opnamedatum : 26-01-2018
DSM V : Depressieve-stemmingsstoornis door alcohol
GAF : 60
Behandelplan : - -

Actieve middelen of maatregelen

Gemarkeerde antwoorden

Laatste 5 orders GGZ

Afspraakgroepen
26-01-2018 Systeem van Pelaar, P.

Laatste 10 orders

Behandeling verkenner
Algemeen
Aanmelding (26-01-2018)

Filter: Standaard filter
Datum: _-_-__ t/m _-_-__ Soort informatie: Aanvullend onderzoek; Aanvullen
Uitslagen: Indicatie:
Specialismen: Functie:
Type contact:

07-12-2017 16:00
Psychiater
PSY ARTS

Anamnese
Patiënt heeft verlies van energie. Suïcidaliteit zou nu niet meer aanwezig zijn.
Conclusie
Aanmelden voor 2xw CT en 2xw RT
Beleid
- Vrije order aan polimedewerker: Graag pt aanmelden voor 2xw CT en 2xw RT (Verpleegkundigen/secretariaat kliniek)

01-12-2017 16:35
Psychiater
ARTS

Conclusie
Verslag MDO
-Reden van aanmelding: Type: Algemene psychiatrie bespreking
Reden aanmelding: Behandelplan bespreken.
-Aanwezig tijdens MDO: Sutorius, psychiater
Vandaele, psychiater
Schaefers, psychiater
Bender, psycholoog
de Vries, kinésitherapeut
Nagelkerke, therapeut
-Overweging/ differentiaal diagnose: Stop temazapam, nicotine pleister ivm stop roken, Lorazepam ivm risico op onttrekking.
Behandelplan bijstellen.
Intake bij Sága verslavingszorg.
-Beleid: Intake bij Sága verslavingszorg.

29-11-2017 15:00
Medisch psycholoog
PSO PSO

Reden van komst / Verwijzing
Status na TS bij depressieve stoornis en stoornis in alcoholgebruik
Anamnese
Was al langere tijd weer somberder. Patiënt ervaart al jaren gevoelens van onverwerkt verdriet rondom een aantal gebeurtenissen in zijn VG, die allemaal geassocieerd zijn met het thema van verlies (dood moeder als student, scheiding van zin vrouw tegen zijn zin). Hij heeft weinig sociale contacten, zit veel thuis en piekert over van alles. Slapen was al weken voor opname lastig door piekeren en hierdoor nam zijn alcoholgebruik ook steeds meer toe. Overdag was hij hierdoor tot steeds minder in staat en werd steeds inactiever en somberder. Eenzaamheid is ook een trigger voor somberheid en alcoholgebruik. Patiënt ziet zijn eigen patiënten als tandarts en soms dorpsgenoten als hij naar het dorpscafe gaat om te biljarten. Zijn kinderen ziet hij veel minder vaak dan hij zou willen. Na de scheiding is het hem niet gelukt een nieuwe relatie te krijgen, alhoewel hij dit best zou willen.
Speciele anamneseSomatische anamnese: Status na TS met paracetamol en alcohol, heeft hiervoor nog acetylcysteine..
Heteroanamnese: volgt (dochter).

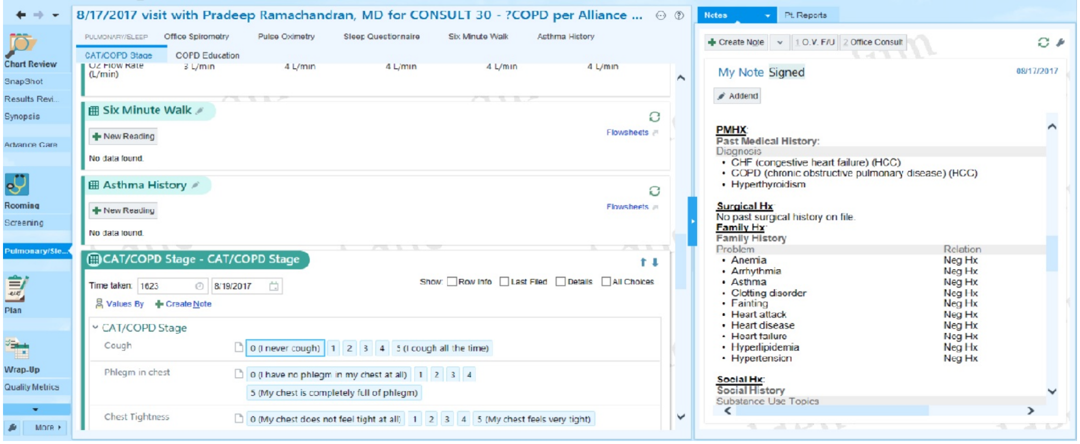
Aanvullend onderzoek
School/opleiding: universitair.
Werk: tandarts, eigen praktijk.
Heteroanamnese: volgt (dochter).
Eerder psychologisch onderzoek: nv.
Observaties tijdens onderzoek: trilt met beide handen.
Indruk en gedrag tijdens onderzoek: gespannen.

Neuro psychologisch onderzoek
Taal (spraak): articuleert duidelijk.
Taalbegrip: intact.
Uiterlijke kenmerken: redelijk verzorgd.
Oriëntatie: intact in TPP.
Gesprekshouding en mimiek: cooperatief.
Contactname: iets afwachtend.
Handvoorkeur: rechts

FIGURE 4.1: Screenshot of HiX software (Chipsoft 2018)

4.1.2 Epic

Epic is an EMR software developer based in the United States. Their website does not mention any logging functionality, but the software does provide a patient environment in which patients can schedule appointments, complete questionnaires and message their doctor. Additionally, there is an app that patients can use on their tablet when they are in the hospital. The app enables them to check their care schedule and access patient education material.



8/17/2017 visit with Pradeep Ramachandran, MD for CONSULT 30 - ?COPD per Alliance ...

O2 Flow Rate (L/min)	3 L/min	4 L/min	4 L/min	4 L/min	4 L/min

Six Minute Walk

No data found.

Asthma History

No data found.

CAT/COPD Stage - CAT/COPD Stage

Time taken: 1623 8/19/2017

Show: □ Row Info □ Last Filed □ Details □ All Choices

CAT/COPD Stage

Cough	0 (I never cough) 1 2 3 4 5 (I cough all the time)
Phlegm in chest	0 (I have no phlegm in my chest at all) 1 2 3 4 5 (My chest is completely full of phlegm)
Chest Tightness	0 (My chest does not feel tight at all) 1 2 3 4 5 (My chest feels very tight)

My Note Signed 08/17/2017

PMHX:
Past Medical History:
Diagnosis
- CHF (congestive heart failure) (HCC)
- COPD (chronic obstructive pulmonary disease) (HCC)
- Hyperthyroidism

Surgical Hx:
No past surgical history on file.
Family Hx:
Family History

Problem	Relation
Anemia	Neg Hx
Arrhythmia	Neg Hx
Asthma	Neg Hx
Clotting disorder	Neg Hx
Fainting	Neg Hx
Heart attack	Neg Hx
Heart disease	Neg Hx
Heart failure	Neg Hx
Hyperlipidemia	Neg Hx
Hypertension	Neg Hx

Social Hx:
Social History
Substance Use Topics

FIGURE 4.2: Screenshot of Epic software (Ramachandran 2017)

4.2 Blockchain-based EMR systems

There is a high interest in blockchain applications in the health care sector. This is reflected in scientific work, pilots in the public sector and startups offering blockchain-based solutions. Estonia is a pioneer in this field, integrating blockchain technology in their e-health applications.

4.2.1 Scientific work

This research would definitely not be the first to incorporate blockchain into a EMR system, although it may be the first one to use blockchain technology for the specific purpose of empowering patients with knowledge over what happened to their data. Researchers have distinguished five key benefits to the use of blockchain as opposed to distributed database management systems for biomedical and health care applications (Kuo, Kim, and Ohno-Machado 2017). These are:

1. Decentralized management, enabling health care stakeholders to collaborate without needing a central managing party;
2. An immutable audit trail, as a blockchain is append-only;
3. Data provenance, as ownership of digital assets on a blockchain can only be changed by the owner and not by a system administrator;
4. Robustness and availability, having records available on multiple places;
5. Improved security and privacy when using proper cryptographic tools.

Especially the first three listed benefits play a role in the system designed and prototyped for this master thesis. In the remainder of this section, four papers that present blockchain-based EMR systems are studied.

MedRec

MedRec is a EMR system aimed at managing authentication, confidentiality, accountability and data-sharing. The paper in which this system is presented identifies interoperability challenges between healthcare provider systems as a major barrier towards effective data sharing. The authors designed a public key cryptography-based blockchain structure that could be applied to create append-only, immutable, timestamped EMRs (Ekblaw et al. 2016). The block content consists of information about data ownership and viewership permissions. Smart contracts are used to log events such as data retrieval. A prototype was made to demonstrate the qualities of the system.

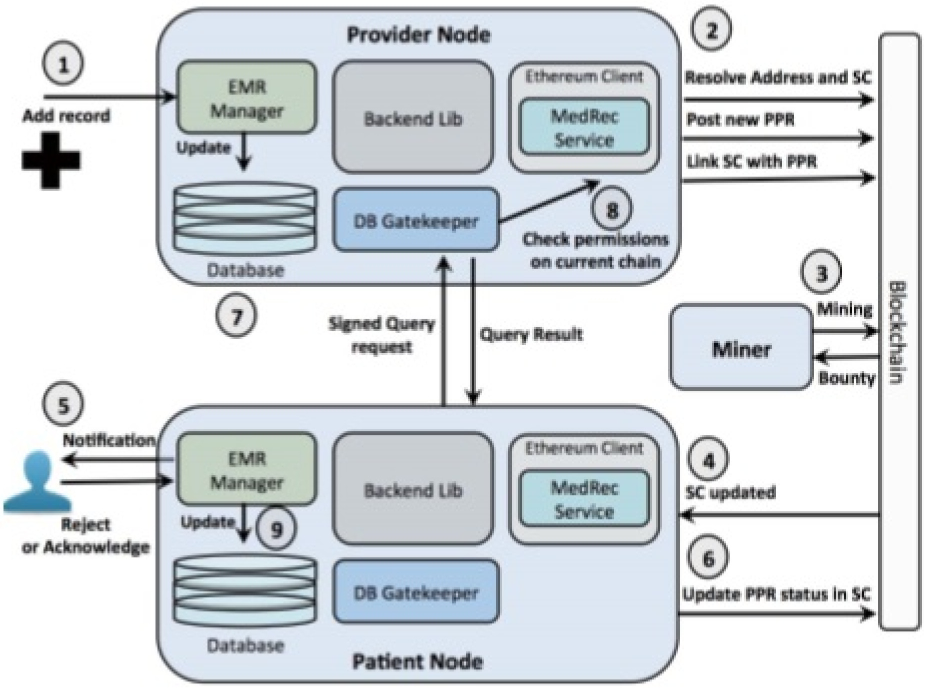


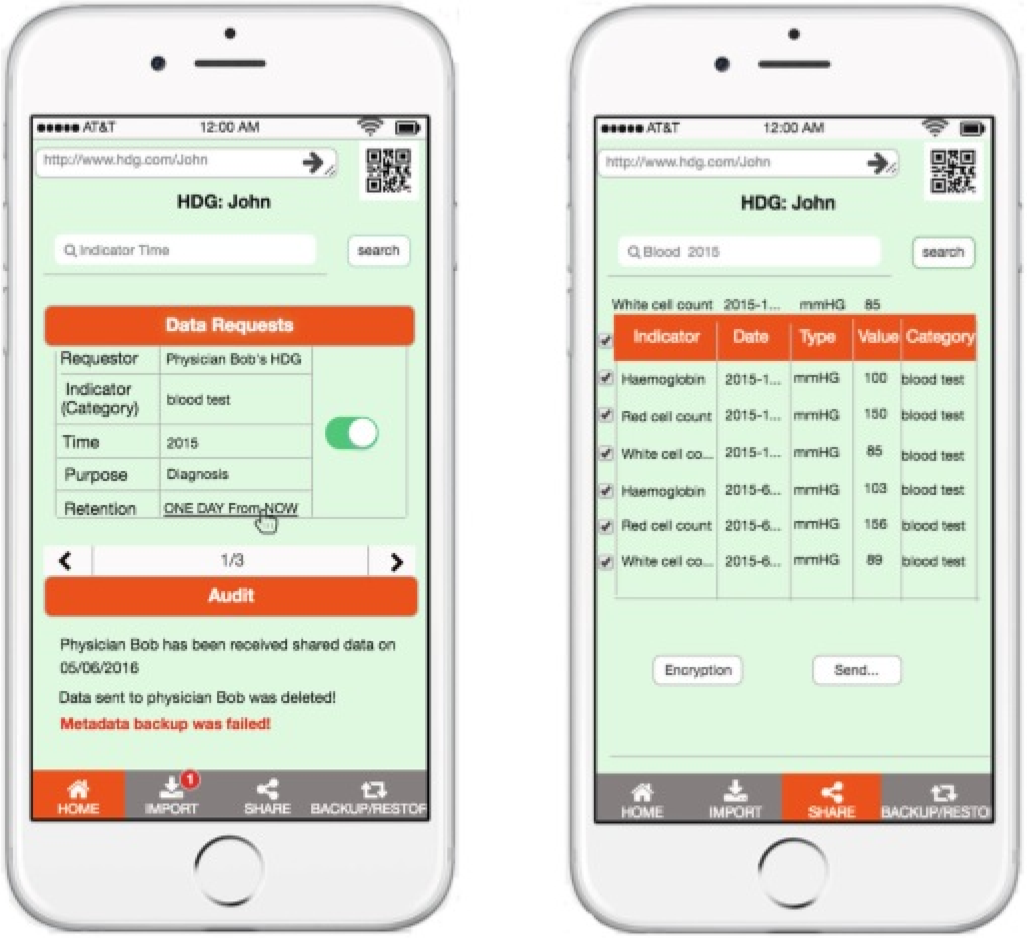
FIGURE 4.3: Overview of MedRec system (Ekblaw et al. 2016)

OpenPDS

Zyskind & Nathan proposed a model called OpenPDS for an information system in which a mechanism for returning computations on the data is included: return answers instead of data itself. The contribution of this paper is twofold: Combination of blockchain and off-blockchain storage to construct a personal data management platform focused on privacy; Perform trusted computing on blockchain-handled data. The proposed systems treats users as the owners of their data and provides them with data transparency and fine-grained access control. A rough sketch of the functionality of the system is as follows: A users installs the application on a smartphone. Data collected on the phone is encrypted using a shared encryption key and sent to the blockchain. The blockchain routes it to an off-blockchain key-value store using a DHT, only retaining a SHA-256 hash pointer. Anyone wanting to access the data can send a request to the blockchain, which in turn verifies the digital signature of the requester as well as the listed permissions for this user (Zyskind, Nathan, et al. 2015). Assuming that users manage their keys in a secure manner, the system provides security and privacy. An adversary cannot really learn interesting information from the blockchain itself, because it only stores hash pointers. Even if it would control a large amount of nodes, the raw data is still encrypted using a key that none of the nodes possess. Adversaries are prevented from posing as a user because of the digitally-signed transactions and the decentralized nature of blockchain.

Healthcare Data Gateway

In 2016, Xiao Yue presented a fairly similar system called the Healthcare Data Gateway app. It is a combination of a traditional database and a gateway. Personal electronic medical data is managed by a blockchain. All data requests are evaluated for permission. In case of a granted permission, secure multiparty computation (sMPC) is used to process patient data without risking patient privacy (Yue et al. 2016).



Requestor	Physician Bob's HDG
Indicator (Category)	blood test
Time	2015
Purpose	Diagnosis
Retention	ONE DAY From NOW

Physician Bob has been received shared data on 05/06/2016

Data sent to physician Bob was deleted!

Metadata backup was failed!

White cell count 2015-1... mmHG 85

☑	Indicator	Date	Type	Value	Category
☑	Haemoglobin	2015-1...	mmHG	100	blood test
☑	Red cell count	2015-1...	mmHG	150	blood test
☑	White cell co...	2015-1...	mmHG	85	blood test
☑	Haemoglobin	2015-6...	mmHG	103	blood test
☑	Red cell count	2015-6...	mmHG	156	blood test
☑	White cell co...	2015-6...	mmHG	89	blood test

(a) Home user interfaces of John's HDG

(b) Data sharing illustration of John's HDG

FIGURE 4.4: Example of HDG screenshots (Yue et al. 2016)

Enigma

Enigma is a computation platform proposed by Zyskind et al. Their paper states that blockchain can neither handle privacy nor heavy computations. Enigma can be connected to an existing blockchain. The goal of the platform is to facilitate developers to build privacy-by-design, decentralized applications without using a trusted third party (Zyskind, Nathan, and Pentland 2015). Just like most blockchain-based systems, it uses a DHT that stores references to the data. sMPC is used by splitting date between nodes and performing computation on these nodes without transferring any information from one node to another. Each node has a piece of seemingly random data, that is useless on its own. In general, sMPC systems are based on secret sharing. This is a category of threshold cryptosystems, in which a secret s is divided into n parts, and at least t shares are required to reconstruct s. Such a system is written as a (t, n) threshold system. Shamir's secret sharing scheme is a famous example of a secret sharing scheme, which uses polynomial interpolation. The Enigma platform provides an API which facilitates the uses of a sharing scheme based on Shamir's scheme. In total, there are three decentralized databases in the system: the public ledger, the DHT and the sMPC database. Nodes are compensated for their computational resources via computation fees.

4.2.2 Startups and industry-based projects

Several startups and government- or industry-based projects have come up in the last few years on the subject of blockchain in healthcare. These range from conceptual frameworks to functioning prototypes. A few Dutch projects are listed here.

Mijn Zorg Log

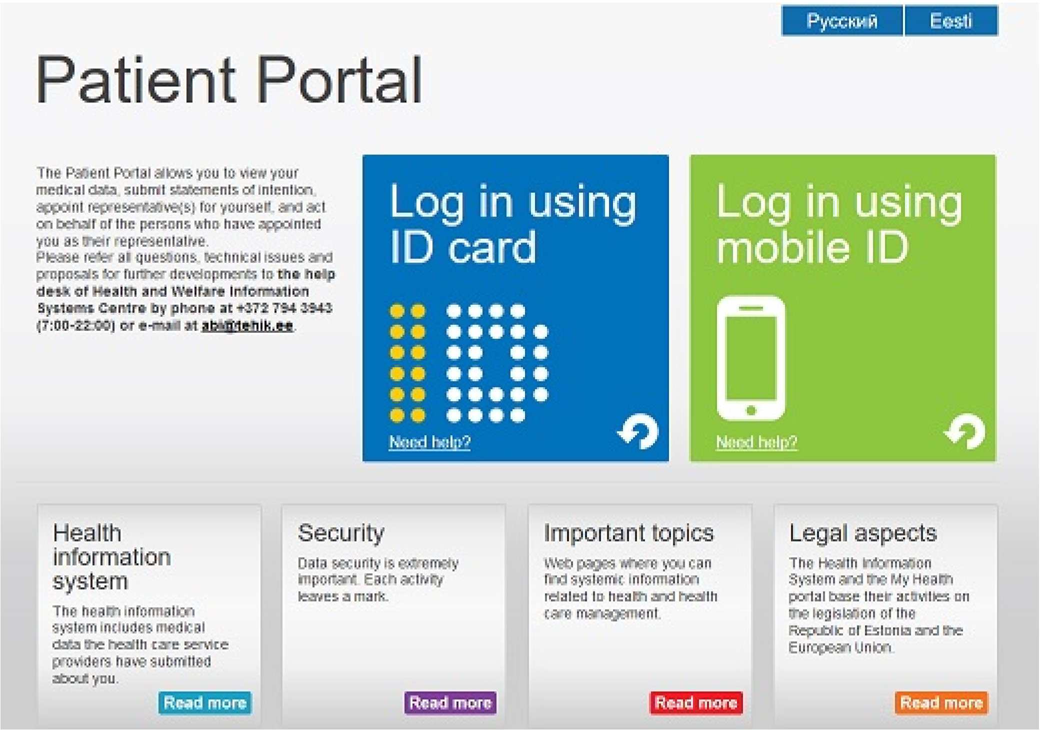
Mijn Zorg Log is a smartphone app, developed by the Dutch Health Care Institute (Dutch: *Zorginstituut Nederland*) in cooperation with blockchain software company Ledger Leopard. This app can be used by people who receive home care to log the hours that the home help spent at their house and the nature of the care. The home care provider can then verify these hours and use them for their administration. A permissioned blockchain is used, with two types of nodes: member nodes and authority nodes. Only authority nodes participate in the mining process. An experiment has been conducted using this app for administration in maternity care. The results were mainly positive, especially concerning the self-reported reduction of the administrative burden (Felix et al. 2018).

MedMij

MedMij is a framework that consists of agreements about how medical data should be exchanged in a blockchain-based healthcare application. It is therefore not a working product in itself. Health care providers that want to develop a digital healthcare application, can hire a MedMij-certified vendor to implement a compliant system. One of the goals of MedMij is to be an aid in the development of personal medical portal for patients (Kusiak 2018).

4.2.3 E-health in Estonia

Estonia is leading in the provision of public digital services to its citizens. Upon the rebirth of this republic in 1991, the digitalization of state administration was deemed essential (Priisalu and Ottis 2017). All patients can see their medical data through the Estonian eHealth Patient Portal after authentication with the national ID card which contains an identification chip. Citizens can deny access to certain medical data to any care provider, inclusing their own GP. Access to the data is recorded and is available to the patient upon request. Almost all prescriptions, hospital discharge letters and insurance claims are digital (Ross 2016). Currently, blockchain software company Guardtime is testing their blockchain implementation for the eHealth sytem and is planning to deploy it in the near future (eHealth Estonia 2018).



Patient Portal

The Patient Portal allows you to view your medical data, submit statements of intention, appoint representative(s) for yourself, and act on behalf of the persons who have appointed you as their representative.
Please refer all questions, technical issues and proposals for further developments to **the help desk of Health and Welfare Information Systems Centre by phone at +372 794 3943 (7:00-22:00) or e-mail at abi@tehik.ee**.

Health information system

The health information system includes medical data the health care service providers have submitted about you.

Security

Data security is extremely important. Each activity leaves a mark.

Important topics

Web pages where you can find systemic information related to health and health care management.

Legal aspects

The Health Information System and the My Health portal base their activities on the legislation of the Republic of Estonia and the European Union.

FIGURE 4.5: Screenshot of Estonian eHealth Patient Portal (eHealth Estonia 2018), screenshot taken by author.

4.3 Discussion of existing systems

In the previous sections, several papers presenting blockchain-based EMR systems were discussed. In this section, we take a look at the question why these systems do not fit the requirements for a tamper-proof logging auditable access log for medical data, as described in Chapter 2.

4.3.1 Implementation details

Unfortunately, not every system presented in section 4.2 is accompanied by a prototype. Three systems provided an implementation or a description of a possible implementation. The following table contains an overview of the implementation details of these systems.

TABLE 4.1: Functionality and choices of current systems

	MedRec	**OpenPDS**	**Healthcare Data Gateway**
Maturity level	Functioning prototype	Functional design	Screen designs only
Goal	Manage: data access	Manage: data ownership, transparency and auditability, access control	Own, control and share own data easily and securely
Blockchain	Ethereum	Not specified, but assumes qualities similar to Bitcoin	Not specified, mentions "private blockchain cloud"
Block content	Data ownership viewer permissions	Hash pointers (Kademlia)	Encrypted healthcare data
Programming language	Python	Not specified	Not specified
Consensus algorithm	Proof-of-Work	Proof-of-Work	Not specified
Mining reward	Access to aggregated anonymized medical data	Not specified	Not specified
Identity confirmation	DNS-like system that maps real-life ID to ETH address	Pseudonymous compound identities	Not specified

Of these three systems, only MedRec features a working prototype, but it is designed to solve other problems than accountability on access. The two other systems do not even have a prototype and are therefore in a very low TRL. Even theoretically, the OpenPDS and Healthcare Data Gateway designs solve other problems than the problem stated in this master thesis. Both systems have the purpose of facilitating secure multiparty computation. The system proposed in this master thesis, will feature a working prototype designed to facilitate the logging of events in an EMR and to validate entries.

4.3.2 Fitness of MedRec for prototype requirements

Out of all the studied literature, MedRec is the system that most closely resembles the envisioned solution for a tamper-proof auditable access log for medical data. It is a very complete system with a good looking architecture. Still, the system has some essential shortcomings when compared with the requirements of Chapter 2. These are the following:

1. Accountability on access: MedRec stores permission information on the blockchain and information for the verification of data integrity. This means that it provides the patient with the power to deny access to some other participant in the system *before* a certain entry has been read. In the MediTrail system, we are not concerned with access control, but with providing a log of the viewership *after* a certain entry has been read.

2. Validation of entries: MedRec informs patients about the content of the entries and therefore provides them with the implicit possibility of verifying the data and taking action if they disagree with an entry. However, there is no official way to validate an entry. Within MediTrail, there should exist an option to explicitly validate an entry.

Considering these shortcomings of even the most related scientific work up to date, it is deemed necessary to construct a new system (prototype) which does fit these requirements.

Chapter 5

System architecture and design choices

This chapter describes possible implementation choices and considerations on why a certain option is chosen for the development of the *MediTrail* prototype.

5.1 From requirements to use cases

Before designing a system, it is important to know exactly what the system should be able to do. This means that the development effort for this prototype will heavily focus on the core functionality and any features that are not deemed absolutely necessary for answering the research questions will be omitted.

5.1.1 Use cases

A straightforward way of describing the actions of users in a system is by creating use cases. Note the requirements as presented in chapter 2:

1. Accountability on access: Every access to an entry in the EMR system is recorded. The log contains information on the name of the user who accessed the file, the name of the file itself, and the timestamp of the event.
2. Validation of entries: A user should be able to sign an entry with a secure digital signature. The digital signatures should be verifiable by anyone in the system.

Based on these requirements, we can distinguish the following use cases:

1. A user uploads a file to the system. They indicate whether they require one or more other users to sign the file. They receive a confirmation of the upload.
2. A user views the list of files that have been uploaded by all users.
3. A user views/downloads any of the files that have been uploaded by all users.
4. A user signs any of the files that have been uploaded by all users. They receive a confirmation that a signature has been placed.
5. A user views the blockchain log, that contains every event of every user uploading, signing or viewing/downloading a file.

These five use cases together form the core functionality of the prototype. Additionally, the system will require some form of identification and authentication to make sure that only authorized users have access to the system and to accurately log the actions of the users.

5.1.2 Use case diagram

Use case diagrams show the interaction of users with the system and the features that users need. The «*extend*» relationship means that the behaviour in the extending use case is supplementary to the extended use case. In this case, a user can upload a file and can choose not to require anyone to sign this file. However, if they choose to ask other user to sign the file, this is supplementary to the basic uploading function. In a similar fashion, viewing/downloading and signing a file extend the function of viewing the list of files.

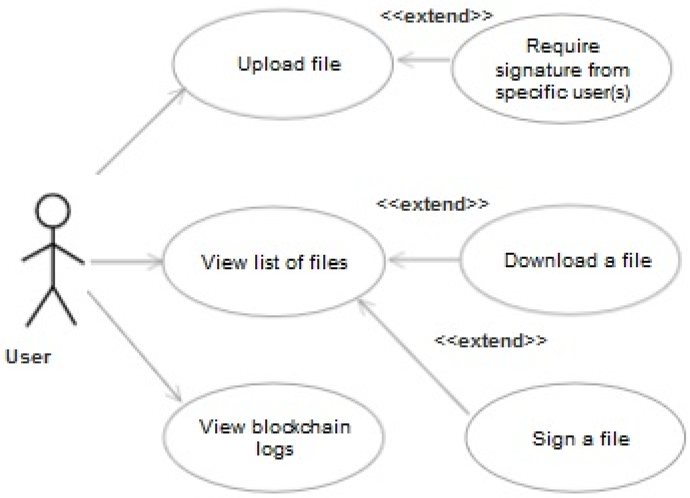


FIGURE 5.1: Use case diagram

In a peer-to-peer system, all users have the same rights. Every node is simultaneously a client and a server. Therefore, this use case diagram is valid for any user in the system, whether they are patients or health care providers. Data on a blockchain is accessible to anyone participating in the network. That is why every patient has their own blockchain in this system, which can be accessed by their health care providers.

5.1.3 Integration with existing tools

The *MediTrail* prototype is not meant as a standalone system, but rather as an extension to existing EMR systems to facilitate the accountability on access and validation of entries. It would be for the system to be compatible with existing tools. However, the implementation details of commercial EMR systems like HiX and Epic are not public. Therefore the goal will be to make conceptual and implementation choices that are mostly platform-independent.

5.2 Selection of blockchain technology

There are several ready-to-use blockchain libraries available that could be used for this project, or a novel blockchain could be built which would be custom to the specific needs of this project.

5.2.1 TrustChain

Researchers at TU Delft developed TrustChain, a scalable blockchain with an emphasis on resilience against one of the primary challenges in permissionless blockchains: Sybil attacks (Otte 2017). A Sybil attack takes place when an adversary forges many fake identities to gain a larger influence of that system than it should actually have (Douceur 2002). The author states that when there is no central trusted authority to assert the one-on-one correspondence between an entity and its identity, it is practically impossible to distinguish identities. This poses a fundamental problem for permissionless blockchains, because they are fully decentralized.

5.2.2 Ethereum

Ethereum is a blockchain that has the possibility of smart contracts as its main feature. Just like Bitcoin, it uses a proof-of-work mining method to make sure that the longest blockchain is the one that has received the greatest investment in terms of computing power (Wood 2014). For Python, there exist several Ethereum libraries, one of which is PyEthereum.

5.2.3 Kademlia

Kademlia is a Distributed Hash Table (DHT) for peer-to-peer networks with an XOR-based metric network topology. A DHT stores (key, value) pairs, the key being a hash, providing a lookup service. Nodes in a Kademlia DHT use UDP to communicate, but Kademlia has mechanisms to overcome packet loss (Maymounkov and Mazieres 2002).

5.2.4 Novel blockchain

Besides using existing blockchains, there is the possibility to create an own novel blockchain from scratch. An advantage of this, is that it provides the researcher with the opportunity to only implement the features that are necessary to achieve the goal. For example, it does not need to handle monetary transactions. The prototype will be the first system to use blockchain technology for tamper-proof logging of access to medical data. It makes sense to create a custom-built blockchain tailored to the requirements of this specific case, instead of using technology that had other purposes. A disadvantage is that it might take more time to write functions that are already defined in the available libraries. This time can then not be spent on the creation or of some features that would be nice to have, or on issues like the security of the system.

5.3 Consensus mechanism

For a permissioned blockchain, Byzantine Fault Tolerance (BFT) protocols are usually used to achieve consensus. Two of these protocols, Practical BFT and HoneyBadger BFT were discussed in chapter 3. Because adversaries are limited in the number of nodes they can create, a tolerance for a lower amount of faulty nodes is accepted. These distributed systems are often small: Google's fault tolerant lock service *Chubby* consists of only five nodes (Burrows 2006). The small size of the *MediTrail* blockchain would thus not be extraordinary.

TABLE 5.1: Comparison consensus algorithms

	PoW	**Practical BFT**	**HoneyBadger BFT**	**PoET**
Synchrony assumption	Asynchronous	Weakly synchronous	Asynchronous	Asynchronous
Additional costs	Potentially high, due to power consumption	Low	Low	Potentially high, due to purchase of Intel chips
Fitness for what blockchain types	Especially fit for permissionless systems	Smaller permissioned systems	Permissioned systems	Permissioned systems
Implementation options	Several python libraries with usable code, own implementation possible	No easy-to-use python library	No easy-to-use python library	Own implementation possible

5.3.1 Proof of work

A Proof of Work based consensus mechanism is a popular choice for permissionless blockchains, because of it resilience against sybil attacks. There are some fundamental differences between a system like Bitcoin and the *Meditrail* portal, which lead to the this high resilience may not be fully needed. For example, Bitcoin is a permissionless blockchain in which any user can enter without restrictions, whereas only the patient and their medical professionals can enter the *Meditrail* system. Another distinction is that the information in the Bitcoin blocks is of a very different nature: it concerns monetary transactions which leads to the need of solving the double spending problem. When considering sustainability, PoW is not the best choice due to the high power consumption as a consequence of competing through the calculation of hashes.

5.3.2 Practical BFT

Practical BFT was a major improvement upon earlier BFT algorithms, but is still only weakly synchronous, which relies on some timing assumptions (Castro, Liskov, et al. 1999). The protocol tolerates a number of malicious or faulty nodes f that is strictly fewer than 1/3 of the total number of nodes. Liveness is only guaranteed when the network behaves as expected. Additionally, it is not as efficient in terms of time consumption as some newer algorithms (Miller et al. 2016).

5.3.3 HoneyBadger BFT

HoneyBadger is a new and very promising BFT algorithm. The researchers state that the protocol is specifically designed for the deployment scenario of a permissioned blockchain (ibid.). HoneyBadger BFT is suitable for even completely asynchronous networks. This is important, because many nodes in the *MediTrail* blockchain will be offline for extended periods of time: patients for example, only have a running node whenever they access the *MediTrail* portal. Unfortunately, the HoneyBadgerBFT API on GitHub does not seem ready to use at all, and making an own implementation of it would be a very complicated and time-consuming endeavor.

5.3.4 Proof of Elapsed Time

One of the goals for developing the PoET consensus algorithm, was to reduce power consumption by miners in PoW-based blockchains (Fairley 2017). It provides a lottery-like system, just like PoW, but not based on hashing power. The mechanism would not be suitable for a permissionless chain, because adversary could then just make multiple identities, each of them receiving an equal chance in the lottery. For a permissioned blockchain, it seems to be a fairly secure and efficient algorithm. This will be evaluated in Chapter 7.

5.4 Digital signature algorithm and implementation

During the literature study phase of this project, research was conducted on various DSA algorithms with the purpose of using these to validate entries in the EMR. To shortly reiterate the use case for these signatures: When uploading a file, users should be able to indicate which other users should sign this file. A file should be marked as *validated* when all the required signers have signed it.

5.4.1 Theoretical considerations on the DSA choice

In a nutshell, a choice has to be made between regular ECDSA and threshold ECDSA. In the table below, a comparison between the fitness of ECDSA and threshold ECDSA for the prototype is made.

TABLE 5.2: Comparison ECDSA and threshold ECDSA

	ECDSA	**Threshold ECDSA**
Initial key distribution	Each user has one private and one public key	Each user has a private key part
Key redistribution	Keys can be kept for an indefinite time	Keys must be redistributed when a new node enters the group
Validity of signature	Signatures can be placed independently	Partial signatures must be remembered in order to construct valid signature

The group of nodes is a dynamic coalition in the sense that health care providers are expected to enter or leave regularly. This makes key (re)distribution hard and time-consuming. Although there exist digital signature schemes that can deal with these challenges (Lubbe, Boer, and Erkin 2014), there are no ready-to-use implementations yet.

5.4.2 Practical considerations on the DSA choice

The second concern is more practical in nature. There does not seem to be a a widely used and thoroughly tested threshold ECDSA library available for Python. Considering the lack of a reliable threshold ECDSA library for Python, the search was extended to libaries that support the creation and verification of regular ECDSA signatures. In the following table, three libraries are compared

TABLE 5.3: Comparison of ECDSA libraries

	python-ecdsa	**python-nss**	**ecpy**
Popularity (16-07-18)	359 GitHub stars	0 GitHub stars	20 GitHub stars
Language	Pure Python	C with Python wrapper	Pure Python
Options	ECDSA only (compatible with OpenSSL)	Supports many network security services	Multiple EC crypto options
Documentation	Abundant documentation with clear examples	Limited and partially outdated documentation	Quality of documentation is sufficient
Speed	0.06-0.6s per generated signature, depending on key length (on laptop from 2008)	Not specified, assumed to be faster because it is written in C	Not specified for signatures
Weaknesses	Vulnerable for timing attacks	Not specified	Not specified

This comparison shows that python-ecdsa is the most popular library and has the most abundant documentation. In turn, the key signature generation is assumed to be slower than for the other libraries. Because the signatures generations are triggered manually in the system and can only be performed on a limited number of entries, it is acceptable to use a slower method. Therefore, python-ecdsa is chosen as the ecdsa library for the prototype.

5.5 Event-based monitoring architecture

One of the core functionalities of the prototype should be the ability to monitor events and save these events to a blockchain. Monitoring sign and upload events is straightforward, using a click on the sign or upload button to trigger the creation of a block that is proposed to be added to the blockchain. Therefore, a method is needed to monitor file events. The first option is to store the files in a directory on the operating system of the user and incorporate a file monitoring function into the prototype to watch for changes. The second option is to let the user download the files on the webpage and use the mouse click event on the download button as the sign that the user has indeed accessed the file.

5.5.1 Monitoring files on OS

Hard disk drives retain data even after the device has been turned off. This retention is called persistent storage. Several operations can be executed on the stored files, as described in the CRUD (Create, Read, Update, Delete) and REST (Representational State Transfer) processes. Applied to the case of an EMR system, the four basic functions of persistent storage in CRUD are:

1. Create: uploading an entry of a medical record;
2. Read: accessing and reading an entry of the medical record;
3. Update: modifying an entry;
4. Delete: destroying an entry.

The research question is centered on accountability on access, so any read event of the medical files should be monitored. There are some Python tools suitable for this purpose.

TABLE 5.4: Comparison of file monitoring libraries

	py-notify	**watchdog**	**fsmonitor**
Popularity (08-08-18)	1 GitHub stars	3062 GitHub stars	55 GitHub stars
Language	C and Python	C and Python	Python
Platform	Linux only	Windows and Linux	Windows and Linux
Documentation	Limited documentation	Small tutorial, some blog posts	Small tutorial
Functionality	Tools for Observer programming pattern	Live filesystem monitoring API and shell utilities	Live filesystem monitoring API

Py-notify seems to be slightly outdated, as the current version is several years old and the documentation page contains dead links. Watchdog is the most popular API in terms of GitHub stars, contains the most extensive documentation and covers the functionality that is needed for the prototype. A downside to these options is that they rely on expensive continuous polling.

5.5.2 Monitoring download of files on webpage

If this option is chosen, every file that has been uploaded should be downloadable by every user in the system. This has two advantages. The first one is that it is user friendly, as one can directly access the information. The second one is that monitoring the access becomes very straightforward: the download of the information is the access event. The third one is that this solution is independent of how and where the files are stored. This is important, because we are not aware of the architecture of existing EMR systems. Of course, this option also has its disadvantage. The APIs listed in the previous section monitor all the CRUD functions, so if a file has been modified, the log will show that. In turn, this solution would not be able to detect this. This does not make the access log less tamper-proof however, because the modified file will create a new event on the access log if it is re-uploaded.

5.6 Final architecture and implementation choices

After careful consideration of all the aforementioned options and their advantages and disadvantages, it is time to design an overview of the system. Building a novel blockchain may take a significant amount of time, but it enables us to tailor the blockchain exactly to the needs of the system in order to answer the research question. In a sense, this project uses the blockchain only for its original basic use: keeping a tamper-proof distributed ledger of events. Developing a novel blockchain also means that a consensus mechanism for this chain has to be chosen. Both HoneyBadger BFT and Proof of Elapsed Time are sensible options: they are suitable for asynchronous permissioned systems and are efficient in energy consumption. Unfortunately, the HoneyBadger API is not ready to use and making an own implementation would be very big task that does not contribute that much to answering the research question. Proof of Elapsed Time is easier to implement, if one disregards the SGX chips that provide attestations of the execution of the correct code. Because the *MediTrail* system is a prototype, it is a pragmatic solution to implement a PoET consensus mechanism without attestations. One of the functions that the system should fulfill is that a user should be able to indicate who has to sign the entry, and that the system indicates when all the required signers have actually signed the entry. Threshold ECDSA is a very elegant solution for this. However, it is inefficient for this specific purpose in terms of key administration. Because there is a very popular and well-documented ECDSA library, the choice falls on using ECDSA with python-ecdsa. *MediTrail* should provide each user with a list of the uploaded files and the possibility to download them. Giving accurate information about when a file was viewed and by whom is a core functionality of the system. Monitoring these events using the download button is a slightly less thorough way than via file monitoring on the operating system, but it is more user-friendly and applicable to many storage solutions.

Chapter 6

MediTrail prototype general overview

The *MediTrail* prototype has been developed. In this chapter, a general overview is presented of the system and the algorithmics behind it.

6.1 Overview

The MediTrail prototype consists of a blockchain and a website to which a user can upload files. The nodes in the blockchain are the patients and their health care providers. Each patient has their own blockchain, so other users of *MediTrail* do not have access to other patient's data.

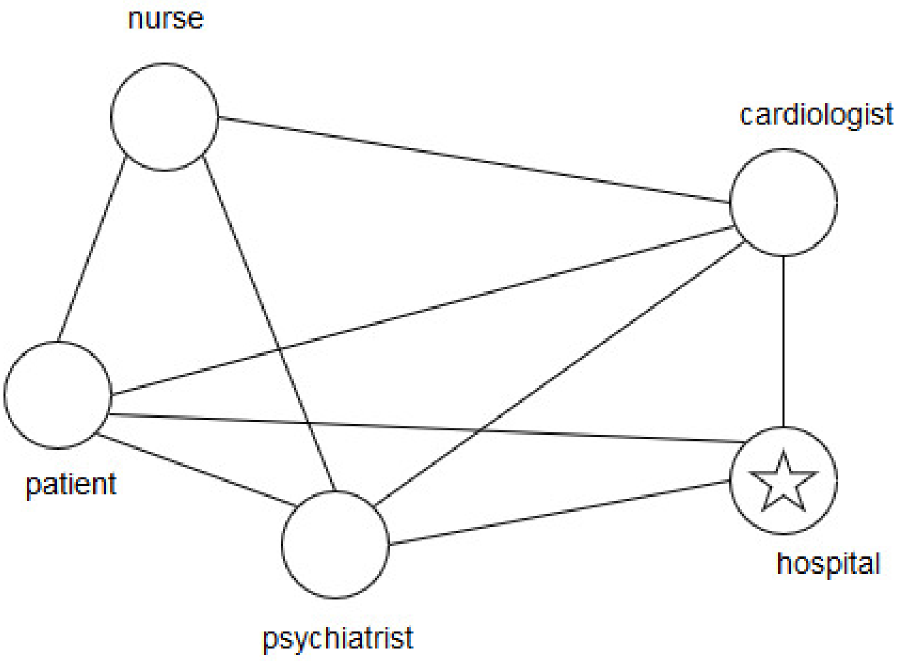


FIGURE 6.1: Example of nodes participating in blockchain

In the figure above, the hospital node is a special node, indicated by a star. This is because the node also provides the webpage which patients and health care providers can use to upload, view and sign files.

6.2 The web page

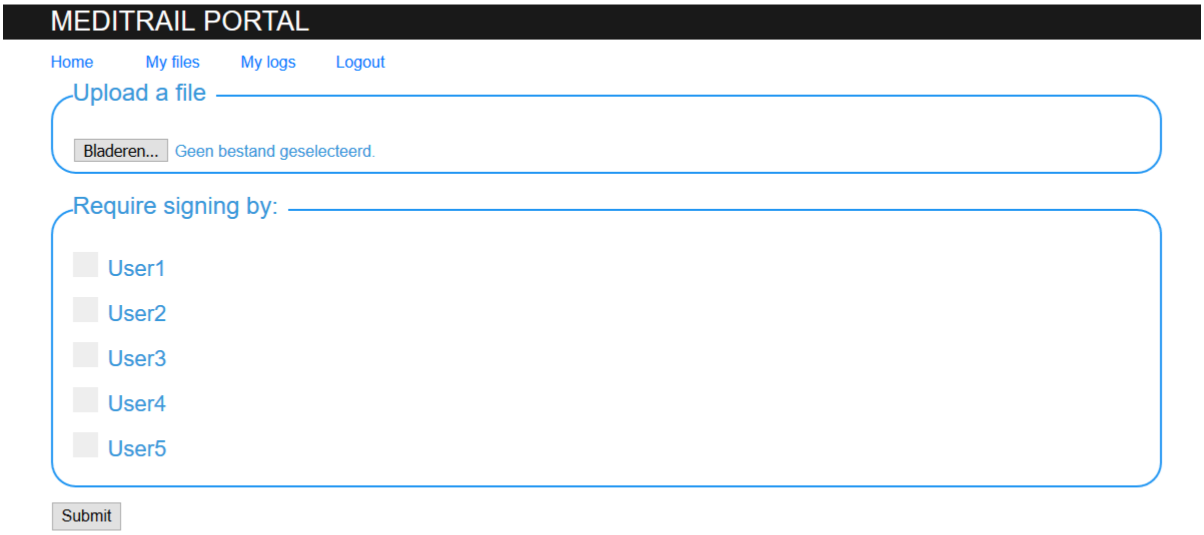
The web framework Flask is used to construct a simple website and receive HTTP requests. To create a minimum level of aesthetic appeal, the html files are enhanced by Bootstrap, a very popular html, css and javascript library.

6.2.1 Identification and authentication

None of the webpages are accessible without authentication. The login page requires a username and a password. After a valid login, the current user is remembered and can access any page and functionality. Every other page contains a *logout* link which logs the user out and redirects them to the login page.

6.2.2 Home page

On the *home* page, there is a button to select a file from the user's computer to upload it to the system. Under this button there are checkboxes that can be checked if the user wants to require other users to sign the file. There is no limit to the number of users that the uploading user can select. When the file has been successfully uploaded, the user is redirected to a page containing a confirmation of the upload.



MEDITRAIL PORTAL

Upload a file

Geen bestand geselecteerd.

Require signing by:

□ User1
□ User2
□ User3
□ User4
□ User5

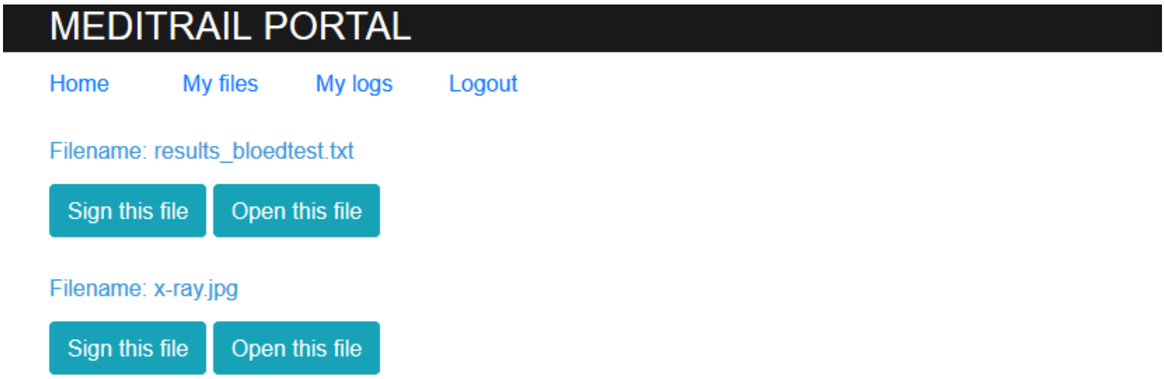
FIGURE 6.2: Screenshot of homepage

6.2.3 My files page

All the files that have been uploaded by any of the users are shown in a list on the *My files* page. Under the filename there are two links: one for opening the file and one for signing it. When a user clicks on *Open this file*, the file is opened in a new tab. When a user clicks on *Sign this file*, the file is signed and the user is redirected to a page containing a confirmation of the signature.

6.2.4 My logs page

A visual representation of the blockchain at the core of this system is found on the *My logs* page. The blocks are listed in chronological order (oldest first). In figure 6.4, a screenshot of a *My logs* page is shown.


MEDITRAIL PORTAL

Filename: results_bloedtest.txt

Filename: x-ray.jpg

FIGURE 6.3: Screenshot of the *my files* page

6.3 Blockchain and algorithms

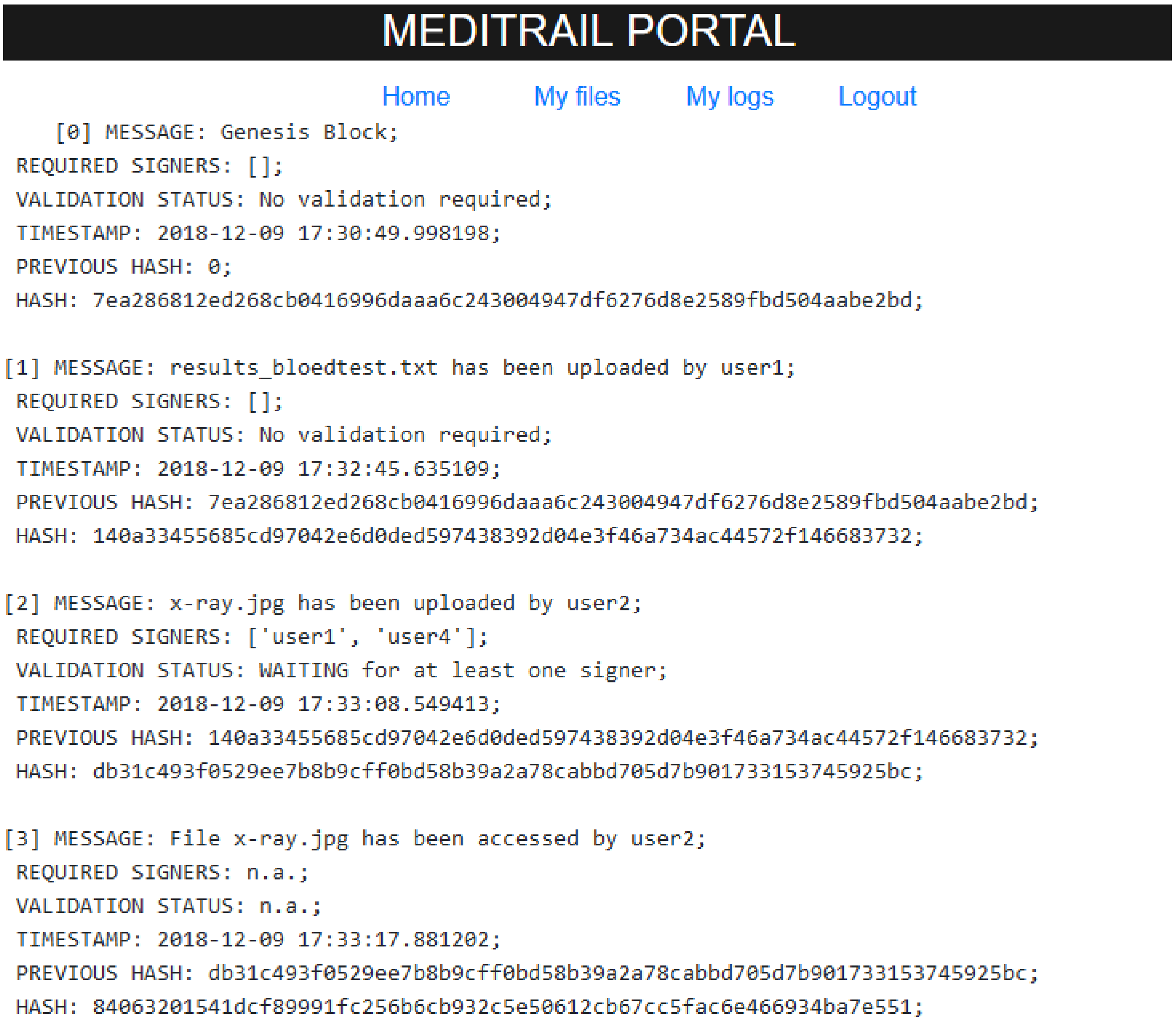
In this section, the implementation details of the main features of the prototype are discussed.

6.3.1 The blockchain

A novel blockchain was constructed for this system, initially based on the SnakeCoin (Nash 2017). SnakeCoin is known as "the world's tiniest blockchain." This system was subsequently expanded with every needed functionality. There are three types of blocks in the blockchain: UploadBlocks, SigningBlocks and ReadBlocks. All of these inherit from the superclass Block. A Block contains the following information:

1. Description: a short description of the event
2. Required signers: users that are asked to sign the file named in this block
3. Validation status: indicates whether the named file is waiting for validation
4. Timestamp
5. SHA256 hash of previous block
6. SHA256 hash over this block's index, timestamp, description, signers, validation status and previous hash

The *description* is a short description of the event, giving the most basic information. The *required signers* field gives the list of users that have been asked to sign the file described in the *message*. The *validation status* indicates whether the named file is still waiting for signatures from the users in the *required signers* field.



```
   [0] MESSAGE: Genesis Block;
 REQUIRED SIGNERS: [];
 VALIDATION STATUS: No validation required;
 TIMESTAMP: 2018-12-09 17:30:49.998198;
 PREVIOUS HASH: 0;
 HASH: 7ea286812ed268cb0416996daaa6c243004947df6276d8e2589fbd504aabe2bd;

[1] MESSAGE: results_bloedtest.txt has been uploaded by user1;
 REQUIRED SIGNERS: [];
 VALIDATION STATUS: No validation required;
 TIMESTAMP: 2018-12-09 17:32:45.635109;
 PREVIOUS HASH: 7ea286812ed268cb0416996daaa6c243004947df6276d8e2589fbd504aabe2bd;
 HASH: 140a33455685cd97042e6d0ded597438392d04e3f46a734ac44572f146683732;

[2] MESSAGE: x-ray.jpg has been uploaded by user2;
 REQUIRED SIGNERS: ['user1', 'user4'];
 VALIDATION STATUS: WAITING for at least one signer;
 TIMESTAMP: 2018-12-09 17:33:08.549413;
 PREVIOUS HASH: 140a33455685cd97042e6d0ded597438392d04e3f46a734ac44572f146683732;
 HASH: db31c493f0529ee7b8b9cff0bd58b39a2a78cabbd705d7b9017331537459250c;

[3] MESSAGE: File x-ray.jpg has been accessed by user2;
 REQUIRED SIGNERS: n.a.;
 VALIDATION STATUS: n.a.;
 TIMESTAMP: 2018-12-09 17:33:17.881202;
 PREVIOUS HASH: db31c493f0529ee7b8b9cff0bd58b39a2a78cabbd705d7b901733153745925bc;
 HASH: 84063201541dcf89991fc256b6cb932c5e50612cb67cc5fac6e466934ba7e551;
```

FIGURE 6.4: Screenshot of the my logs page

6.3.2 File upload

When a file has been uploaded to the system, it is stored in the local filestorage. An UploadBlock is created and added to the blockchain. In addition to the information in a standard Block, an UploadBlock contains a field *uploaded_file* which is the name of the uploaded file.

6.3.3 File signing

To place a signature on a file, the *find_event_and_sign* method is called with *postedhash*, the hash value of the UploadBlock of the file, the name of the user and their private key as parameters. The *find_most_recent_sign_block_for_event* method is called. This starts looking from the most recent SigningBlock to the oldest, until it has found the most recent SigningBlock for the event in which the file was uploaded that the user wants to sign. When this block is found, or it is established that there is no such block, a signature is placed with the user's private key and a SigningBlock is added with the updated required_signers array.

```
Data: User $u$, private key of the user $sk_u$, postedhash;
Result: File is signed, sign event is added to blockchain.
initialization;
latest_signing_block = find_latest_signing_block(postedhash);
if lastest_signing_block is null then
    sign file with $sk_u$;
    required_signers ← original_upload_block.required_signers.copy();
    if $u$ is a required signer then
        remove $u$ from required_signers;
    end
    add SigningBlock to blockchain;
end
if lastest_signing_block is not null then
    sign file with $sk_u$;
    required_signers ← latest_signing_block.required_signers.copy();
    if $u$ is a required signer then
        remove $u$ from required_signers;
    end
    add SigningBlock to blockchain;
end
```

Algorithm 1: find_event_and_sign

```
Data: postedhash;
Result: Returns latest SigningBlock for file with postedhash.
initialization;
latest_signing_block ← null;
foreach block in blockchain do
    if block is instance of SigningBlock then
        if block.hash equals postedhash then
            latest_signing_block ← block;
        end
    end
end
return latest_signing_block;
```

Algorithm 2: find_latest_signing_block

The worst-case time complexity of the method is $O(n)$ with n being the number of blocks in the blockchain. There is one foreach loop which inspects all blocks from the most to the least recent, costing linear time. The other operations are executed in constant time.

6.3.4 File access

When a user accesses a file, a new ReadBlock must be added to the blockchain containing the information about this event. Because file names may not be unique, the hash of the file that is being accessed is posted. This hash is used to find the file name.

```
Data: Postedhash;
Result: Read event is added to blockchain;
initialization;
filename;
foreach block in blockchain do
    if block is instance of SigningBlock then
        if block.hash equals postedhash then
            filename ← block.uploaded_file.filename;
            break;
        end
    end
end
add ReadBlock;
```

Algorithm 3: find_event_and_add_read_block

The worst-case time complexity of the method is $O(n)$ with n being the number of blocks in the blockchain. There is one foreach loop which inspects all blocks from the oldest to the most recent, costing linear time. The other operations are executed in constant time.

Chapter 7

Validation and performance

In this chapter the validity of the system is assessed against requirements in previous chapters and the performance of the system is measured.

7.1 Solving Barbie's problem

This master thesis has attempted to solve a particular problem in a new way with the application of blockchain technology. In particular, the *MediTrail* portal should be the first system that provides patients with power and knowledge over their data in the form of an auditable access log for their medical data. The goal of this chapter is to determine the extent to which the *MediTrail* prototype actually solves Ms. De Jong's problem - and that of many other patients.

7.1.1 Fulfillment of accountability and validation requirements

The requirements to the system to satisfy the research goals as described in chapter 2 are accountability on access and validation of entries.

Accountability on access

Requirement: Every access to an entry in the EMR system is recorded. The log contains information on the name of the user who accessed the file, the name of the file itself, and the timestamp of the event.

MediTrail prototype: Whenever a user accesses an entry via the webpage pertaining to the system, a block logging this event is added to the blockchain. This block contains the name of the user who uploaded the file, the name of the file and a timestamp, in addition to other fields.

Validation of entries

Requirement: A user should be able to sign an entry with a secure digital signature. The digital signatures should be verifiable by anyone in the system.

MediTrail prototype: All users can sign an entry via the "My files" page. This action sends add a block to the chain describing this event. For a user, it may not be clear how this signature works and thus may not understand its value. There is no method presented to the user to verify a signature.

7.1.2 Fulfillment of CIA triad requirements

To adhere to the standards of the CIA triad, the system should satisfy confidentiality, integrity and availability requirements.

Confidentiality

Requirement: Information stored in the EMR system itself as well as the event log should only be accessible to the users it is intended for.

MediTrail prototype: The security level of the identification and authorization method is very low. Credentials like usernames and passwords are hardcoded. It falls out of the scope of this project to provide proper security for this, because the system is intended as an extension for existing EMR systems which presumably already have a secure login system in place.

Integrity

Requirement: Information stored in the EMR system cannot be changed by and adversary without being noticed.

MediTrail prototype: The goal of integrity is very closely related to that of accountability in the previous section. A blockchain is responsible for storing the access and event data. When any of the information in the block is changed, the hash value is changed and thus the chain of hashes is broken. This is evidence that the log has been subject to tampering.

Availability

Requirement: Information stored in the EMR system is available for the users whenever they need or want to access it.

MediTrail prototype: The system provides the users with a website on which they can perform the actions. If this website is offline, users cannot access the files, sign files, or look at the audit log. In theory, the code could be modify to provide a GUI-less API as well, in order to let nodes send information to each other independently for the website. Alternatively, multiple health care providers could set up a web server for the application.

7.1.3 Fulfillment of user experience requirements

The requirements for the user experience focus on easy navigation, clear information and the evidence of the blockchain being untampered.

Easy navigation

Requirement: The user should be able to navigate between the functionalities of the system without effort.

MediTrail prototype: The prototype provides the user with a very simple website which has only four tabs. Any information is therefore easy to find with one click. Uploading a file is straightforward.

Clear information

Requirement: The information displayed to the user should be clear and easily understandable.

MediTrail prototype: A user can find information on two pages: the *My files* page and the *My logs* page. On the *My files* page, the user can download and sign files using the buttons. There is not much room for confusion here. The *My logs* page presents a visual representation of the blockchain to the user. For people unfamiliar wih blockchain technology, this may be slightly difficult to interpret. On the other hand, the messages and timestamps in each block should be clear for everyone.

Verifiably untampered

Requirement: The user should be able to easily verify that the access log has not been tampered with.

MediTrail prototype: Each block on the *My logs* page contains a hash value and the hash value of the previous block. By checking the sequence of these hash values, users can confirm that the chain has not been tampered with.

7.2 Resistance against attacks

If there is a malicious actor in the system, they may try to attack the network in order to gain control of the blockchain and possibly replace the blockchain by a fraudulent one. Sybil attacks were briefly discussed in Chapter 3. The threat of Sybil attacks is mainly present in permissionless blockchains, where anyone can join and make an unlimited amount of nodes. We consider a blockchain compromised if the attacker generates nodes, the nodes propose blocks that get added to the blockchain and the chain containing the malicious blocks becomes the longest chain. In Intel's Sawtooth Lake implementation of PoET, a block verification method is used to reject suspicious blocks. This is based on statistical tests. Assuming that each node has the same winning probability in a round, the number of winning times per node follows a binomial distribution. When the number of nodes is large enough, it approximates a normal distribution. A one-sided z-test is used to assess whether a node is generating blocks too fast (winning the lottery too often). A z-score can be calculated as $z = \frac{x_{win} - mp}{\sqrt{mp(1-p)}}$, where x_{win} is the number of times the node has won the lottery, m is the total number of blocks in the system and p is the probability that a node wins a round. If the z-score is larger than a predefined parameter **zmax**, the block is rejected. When using the Intel SGX chips with Sawtooth Lake, an attacker needs to control $\Theta = \frac{\log \log n}{\log n}$ malicious nodes, where n is the total number of nodes in the system, in order to simulate the fastest honest node (Chen et al. 2017). In order to achieve a constant threshold, the researchers propose two potential approaches. One is to pick a more concentrated probability distribution. Another is to reject blocks that have been proposed by a certain fraction of nodes, even if some honest nodes may be included. The consensus mechanism implemented in *MediTrail* follows the PoET philosophy but is not equipped with a trusted execution environment or statistical tests. In this system, a user needs an account to the portal with login credentials in order to create a node. By keeping a good administration of the accounts, no participant in the system should be able to create more than one node.

MediTrail does not protect against some simple actions that a user of the system can take after accessing the medical information. For example, the user can take a picture of the screen or download the file and diffuse it illegitimately. A measure that could be taken against this behaviour is adding a watermark to every file, containing information about the user and time and date of the download. This can aid in tracking down the responsible person in case a file is leaked.

7.3 Correctness of blockchain

In order to use the *MediTrail* portal to create an auditable access trail, it is necessary that the blocks in the chain are correct. During use tests, it turned out that there were sometimes errors in the hashes of the blocks when using the PoET consensus mechanism. When adding a a large amount of files at once, some blocks did not contain a new hash value, but used the hash value of the previous block instead. Thus, the *previous hash* and the *hash* field contained the same value. As this affects the core functionality of the chain, it was very important to mitigate this issue. After extensive debugging effort it turned out that adding a 10 ms waiting time between the creation of two blocks removed the problem.

7.4 Speed of *MediTrail* features

The performance of the system, particularly the speed of the transactions, should not be a deterrent to using the *MediTrail* portal. In this section, we take a look on how the prototype performs under different configurations. For the experiments with the PoET consensus, the sleeping time for each node is a random amount of time between 500 and 1000 ms.

7.4.1 Data representation

For every experiment, the average time needed to perform the function with different parameters is shown in a table. To represent the results of the file upload experiments and the file read experiments, box plots are used. A box plot is a graphical representation of groups of numbers, where the box itself represents the interquartile range (midspread). The line in the middle of the box is the median of the data set. A cross indicates the average of the data set. The whiskers represent all the data points still within 1.5 times the IQR of the lower or higher quartile. Any outlier is shown as a dot.

7.4.2 Uploading files on the *MediTrail* portal

The speed of uploading a file and adding the upload event to the blockchain is compared for the system with PoET consensus and no consensus. The results are obtained by executing the method 100 times.

TABLE 7.1: Comparison average time required for adding a file upload event to the chain

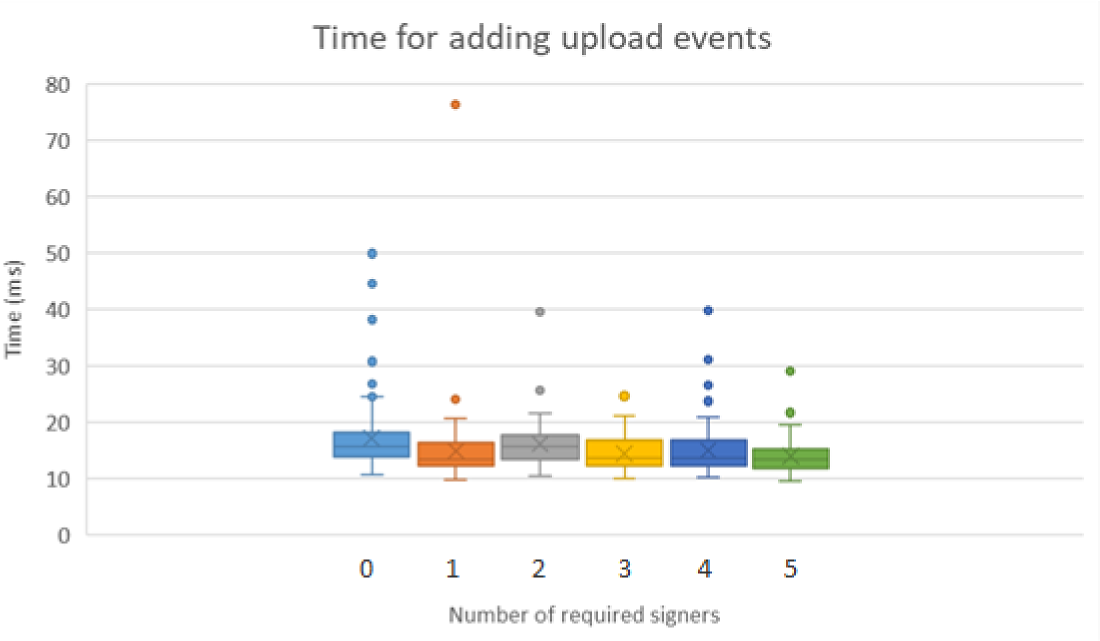


FIGURE 7.1: Time needed for adding upload blocks (N=100), no consensus

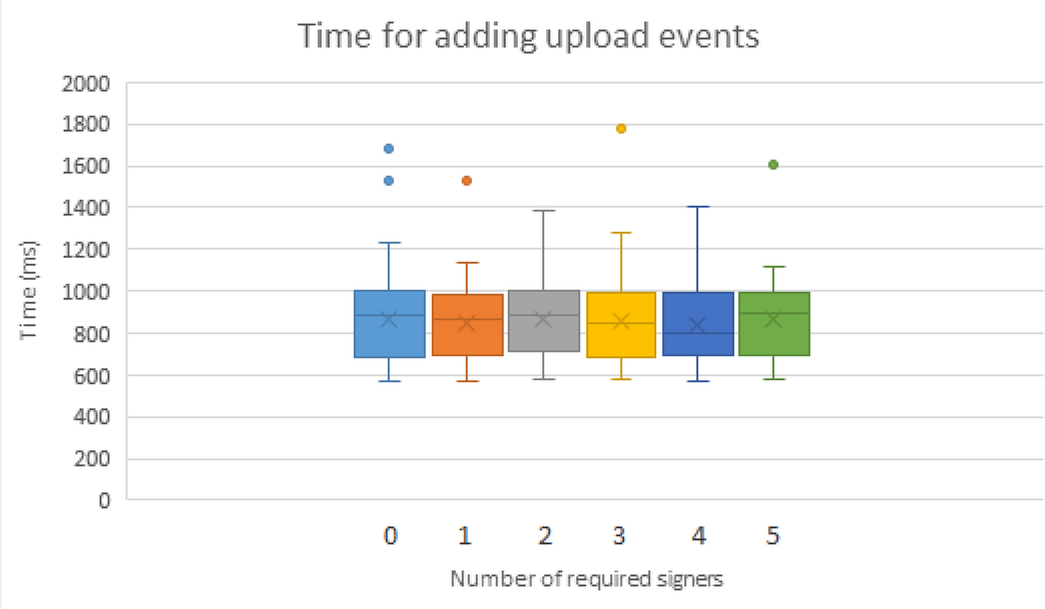


FIGURE 7.2: Time needed for adding upload blocks (N=100), PoET consensus

Number of required signers	PoET	No cons.
0	863 ms	17 ms
1	845 ms	15 ms
2	869 ms	16 ms
3	886 ms	15 ms
4	843 ms	15 ms
5	864 ms	14 ms

Adding an UploadBlock to the blockchain with a limited number of required signers and no consensus algorithm costs around 15 ms. This time is barely noticeable for a user and will not be an impediment to the user-friendliness of the system. The results for the experiment when using PoET consensus with the chosen parameters are higher. Adding an upload event to the blockchain using the PoET consensus algorithm costs on average between 800 and 900 ms, with the most significant outlier taking just under 1,8 seconds. If a user checks the *My Logs* page immediately after uploading a file might they be annoyed or concerned at not seeing the entry yet.

7.4.3 Signing files in the *My Files* list

Next, the speed of signing a file and adding the signing event to the blockchain is compared for the system with PoET consensus and no consensus. The best case is when the file that should be signed was the latest file to be signed as well. The longer ago a file has been signed, the more blocks the system has to search, so the longer the functionality takes. Results are obtained by executing the method 20 times. This small number of repetitions is due to the need for manual signing of the files: the production of each data point requires the researcher to upload a file, sign the file, add more files and sign the file again without automation. Unfortunately, this number of data points is too low to be represented by a box plot, because the variance is meaningless. This is why these experimental results are accompanied by a scatter plot.

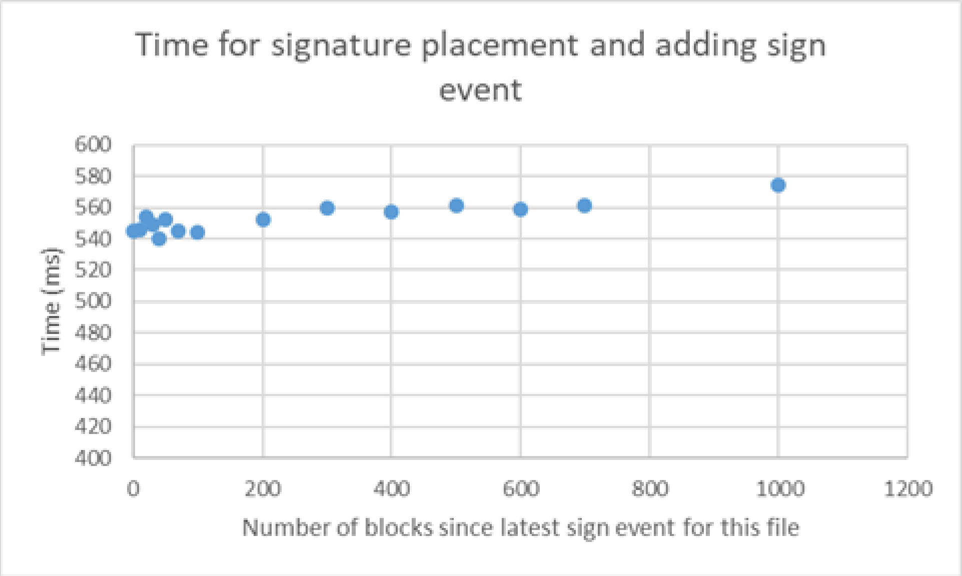


FIGURE 7.3: Time needed for placing a signature and adding a signing block (N=20), no consensus

TABLE 7.2: Comparison speed of signatures plus adding signature event

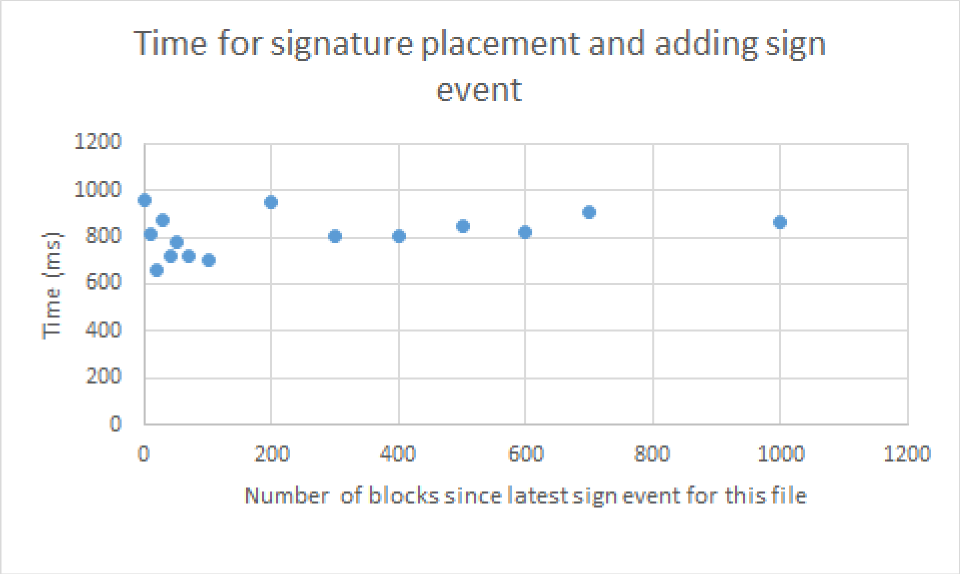


FIGURE 7.4: Time needed for placing a signature and adding a signing block (N=20), PoET consensus

Distance to latest signing block	PoET consensus	No consensus
0 blocks	958 ms	545 ms
10 blocks	809 ms	546 ms
20 blocks	661 ms	554 ms
30 blocks	871 ms	549 ms
40 blocks	721 ms	540 ms
50 blocks	775 ms	552 ms
70 blocks	718 ms	545 ms
100 blocks	705 ms	544 ms
200 blocks	950 ms	552 ms
300 blocks	804 ms	560 ms
400 blocks	801 ms	557 ms
500 blocks	847 ms	561 ms
600 blocks	818 ms	559 ms
700 blocks	910 ms	561 ms
1000 blocks	864 ms	574 ms

With a combined signing and block-adding speed for the signing functionality of around 550 ms, the results fit into the expected range according to the speed of signing with python-ecdsa as mentioned in table 5.3. Interestingly, the addition of the consensus mechanism does not add as much overhead as expected. The average time needed for the placement of a signature and adding the event to the blockchain using PoET consensus stays under 1000 ms for every round of experiments.

7.4.4 Downloading files in the *My files* list

The speed adding the read event to the blockchain after downloading the file is compared for the system with PoET consensus and no consensus. The speed of the download itself is not measured, as it partly depends on the bandwidth of the internet connection. The results are obtained by executing the method 100 times.

TABLE 7.3: Comparison speed of adding download event

Index of upload event	**PoET consensus**	**No consensus**
1	826 ms	12 ms
10	871 ms	12 ms
20	1112 ms	12 ms
30	847 ms	12 ms
40	893 ms	12 ms
50	844 ms	12 ms
70	857 ms	12 ms
100	857 ms	12 ms
150	895 ms	12 ms
200	887 ms	13 ms
300	901 ms	12 ms
400	930 ms	12 ms
500	944 ms	12 ms
600	994 ms	13 ms
700	986 ms	13 ms
1000	1077 ms	14 ms

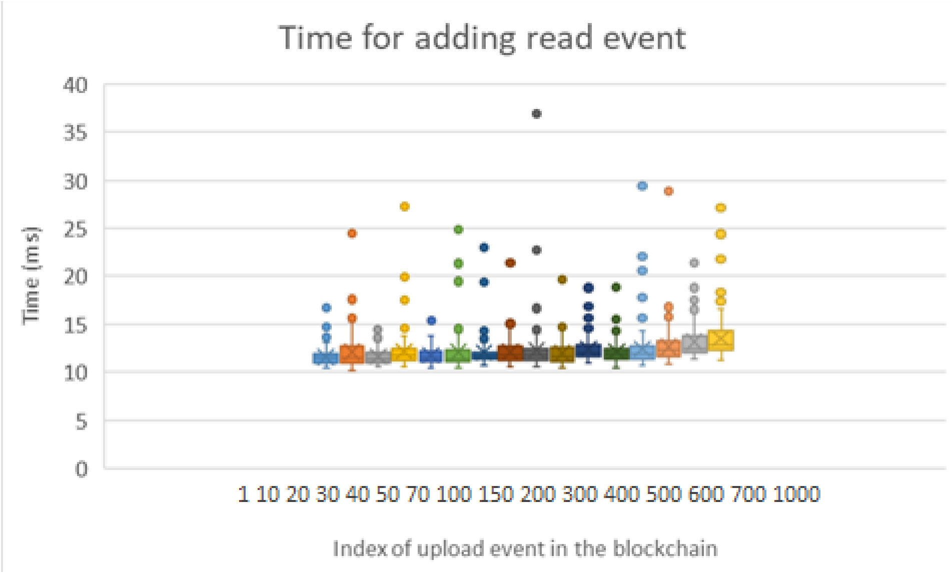


FIGURE 7.5: Time needed to add a read block (N=100), no consensus

Adding a read event to the chain happens quite fast, comparable to the results of assing an upload event. These results are even lower: around 12 ms for most rounds but slightly increasing as the files that are being accessed have a higher index (are more recent). In hindsight, constructing the code for adding a ReadBlock to the blockchain in such a way that the search for the file that is being accessed starts at the genesis block was a wrong choice. It should start at the most recent block. It makes sense that, as time passes by and new entries are added regularly, users will want to access recent files instead of older files. Although the search is conducted in linear time, the difference in performance may become significant if a blockchain has grown very long.

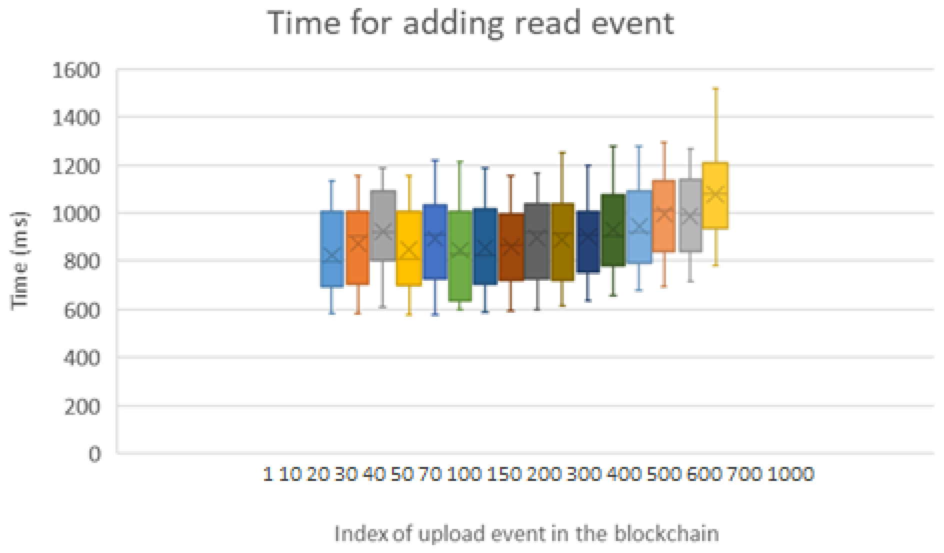


FIGURE 7.6: Time needed to add a read block (N=100), PoET consensus

7.5 General discussion of performance

In general, the performance of the different functions of the *MediTrail* prototype is overwhelmingly dependent on the consensus algorithm. When no consensus algorithm is used, the system works quite fast. Using the PoET consensus algorithm with the chosen parameters add substantial overhead. It seems likely that if a proper implementation would be made with the appropriate supporting hardware for PoET consensus, the sleeping time for the nodes could decrease significantly and thus the performance would improve. Unfortunately the Sawtooth documentation does not provide advice for choosing the right parameters. However, the system would in practice be used as a plug-in to EMR systems, and not as a system to automatically add events to a chain. In a hospital situation, each block added to the blockchain is the result of a manual action. Even with the current PoET parameters, the time needed to add an event to the chain is barely noticeable for a user. This is why the performance of the systems could be classified as sufficient for fulfilling the user experience requirements.

7.6 Reflection on research methodology

The performance tests on the *MediTrail* prototype are quite limited in terms of the diversity in configurations. This is mainly due to the fact that the system was developed for demo purposes and the code is not suitable for large automated tests without having to make major changes in the code structure. If such a project would be done again in the future, it would be wise to design the architecture of the system in a more flexible and modular way so the performance tests are facilitated. Development time is a scarce resource in these kinds of projects. If choosing an existing blockchain like Ethereum instead of building a novel blockchain would have resulted in the opportunity to create a system that is suitable for more thorough experiments, it might have been an advantageous choice. It would also have been a good idea to research the most appropriate sleeping time frame and probability distribution, to find the right balance between performance and security.

Chapter 8

Conclusions and future work

The *MediTrail* prototype is the first system for EMRs that provides users with a blockchain-based, tamper-proof auditable access trail. Conceptually, it delivers in fulfilling the requirements that provide accountability on access and validation of entries. Implementation-wise, there are major improvements to make.

8.1 Conclusions

There is a considerable societal need for patients to have knowledge and power over their medical data. The *MediTrail* portal is a prototype that has been developed with a twofold purpose: to provide patients with a tamper-proof access log of their data and to enable patients as well as health care providers to validate an entry in an EMR system with a digital signature. *MediTrail* succeeds in presenting a tamper-proof access log to patients. Thanks to the novel blockchain that was specifically created for this purpose, blocks contain only the most relevant information for users. As long as the master node (hosted at the hospital) is online, a patient can check their access log at any time. Users can place a digital signature to indicate the validity of a certain entry. Unfortunately, users cannot verify each other's signatures. The prototype underwent some performance tests to measure the speed of different functionalities. Without using a consensus algorithm, the system performs pretty well. It is fast enough for users not to notice a waiting time with normal use. After introducing the Proof of Elapsed Time consensus algorithm, the system performs worse. The most time-costly transactions that were tested can take up to 1500 ms, averaging around 1100 ms. In a real world situation, this overhead can be reduced by using the right hardware. Although this prototype is still in a low maturity level, it does succeed in presenting an interesting step in using blockchain technology for medical data.

8.2 Future work

A weakness against which the system does not properly protect, is the possibility that a user downloads a file from the system and spreads this via other ways such as email or by uploading it to a service which makes the file publicly available. Watermarking the files can aid in determining the source of the file when an investigation is opened. The question whether a consensus algorithm should be used for the *MediTrail* blockchain should be further studied. For this master thesis, a system with and without consensus algorithm was compared. The comparison is not entirely fair, because of limitations in terms of availability of useful APIs and hardware. If another suitable consensus algorithm such as HoneyBadger BFT would have been available in the form of a usable python library, or the researcher would have disposed of the appropriate Intel chips for PoET consensus, a more interesting comparison in performance could have been made. Additionally, the options of creating trust with digital signatures or with a protocol such as TrustChain should be explored to implement in this system. For the goal of exploring the practical benefits and drawbacks of a system like *MediTrail*, a test network using fake medical data could be set up, simulating normal activity and interaction with the system. A legal concern is the virtually eternal storage of logging data on a blockchain. It should be investigated what the GDPR implications are of the difficulties of destroying the data.

Bibliography

Adams, A. and M. A. Sasse (1999). “Users are not the enemy”. In: *Communications of the ACM* 42.12, pp. 40–46.

Anderson, Ross J (1996). “A security policy model for clinical information systems”. In: *Security and privacy, 1996. proceedings., 1996 ieee symposium on*. IEEE, pp. 30–43.

Anderson et al. (2009). “Database State: A Report Commissioned by the Joseph Rowntree Reform Trust Ltd”. In:

Bashir, I. (2017). *Mastering Blockchain*. Packt Publishing Ltd.

Burrows, Mike (2006). “The Chubby lock service for loosely-coupled distributed systems”. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, pp. 335–350.

Calder, A. (2016). *EU GDPR A Pocket Guide*. IT Governance Ltd.

Carvel, J. (2017). “Concern over NHS’s IT systems after 50 view celebrity’s details”. In: *The Guardian*.

Castro, M., B. Liskov, et al. (1999). “Practical Byzantine fault tolerance”. In: *OSDI*. Vol. 99, pp. 173–186.

Chen, L. et al. (2017). “On security analysis of proof-of-elapsed-time (poet)”. In: *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, pp. 282–297.

Chipsoft (2018). *Update PAAZ screenshot*. https://assets.chipsoft.com/PublishingImages/Solutions/Oplossingen/HiX%20update%20PAAZ%20screenshot.png. Accessed: 01-10-2018.

De Telegraaf (2018). “Barbie met spoed naar ziekenhuis gebracht”. In: *De Telegraaf*.

Desmedt, Y. and Y. Frankel (1991). “Shared generation of authenticators and signatures”. In: *Annual International Cryptology Conference*. Springer, pp. 457–469.

Douceur, John R (2002). “The sybil attack”. In: *International workshop on peer-to-peer systems*. Springer, pp. 251–260.

eHealth Estonia (2018). *E-health records*. https://e-estonia.com/solutions/healthcare/e-health-record/. Accessed: 02-10-2018.

Ekblaw, A. et al. (2016). “A Case Study for Blockchain in Healthcare:“MedRec” prototype for electronic health records and medical research data”. In: *Proceedings of IEEE Open & Big Data Conference*. Vol. 13, p. 13.

European Society of Radiology (2017). “The new EU General Data Protection Regulation: what the radiologist should know”. In: *Insights into imaging* 8.3, pp. 295–299.

Fairley, P. (2017). “The Ridiculous Amount of Energy It Takes to Run Bitcoin”. In: *IEEE Spectrum*.

Felix, I. et al. (2018). *Praktijkproef blockchain kraamzorg met Mijn Zorg Log*.

Gennaro, Rosario, Steven Goldfeder, and Arvind Narayanan (2016). “Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security”. In: *International Conference on Applied Cryptography and Network Security*. Springer, pp. 156–174.

Gennaro, R. et al. (1996). "Robust threshold DSS signatures". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, pp. 354–371.

Gillum, Richard F (2013). "From papyrus to the electronic tablet: a brief history of the clinical medical record with lessons for the digital age". In: *The American journal of medicine* 126.10, pp. 853–857.

Green, Marc and Thomas Eisenbarth (2015). "Strength in Numbers: Threshold ECDSA to Protect Keys in the Cloud". In: *IACR Cryptology ePrint Archive* 2015, p. 1169.

HagaZiekenhuis (2016). "HagaZiekenhuis stapt succesvol over naar EPD HiX". In:

Hassey, Alan, David Gerrett, and Ali Wilson (2001). "A survey of validity and utility of electronic patient records in a general practice". In: *Bmj* 322.7299, pp. 1401–1405.

Icke, V. (2018). "Barbie wees ons op ontwerpfout in Elektronisch Patientendossier". In: *NRC Handelsblad*.

Intel Corporation (2017). *Sawtooth introduction*. https://sawtooth.hyperledger.org/docs/core/nightly/0-8/introduction.html. Accessed: 05-11-2018.

Kiayias, A . et al. (2017). "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Annual International Cryptology Conference*. Springer, pp. 357–388.

Kostkova, P . et al. (2016). "Who owns the data? Open data for healthcare". In: *Frontiers in public health* 4, p. 7.

Kuo, T., H. Kim, and L. Ohno-Machado (2017). "Blockchain distributed ledger technologies for biomedical and health care applications". In: *Journal of the American Medical Informatics Association* 24.6, pp. 1211–1220.

Kusiak, L. (2018). "Baas over eigen zorgdata". In: *Zorgvisie ICT* 19.4, pp. 12–14.

Lamport, L., R. Shostak, and M. Pease (1982). "The Byzantine generals problem". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3, pp. 382–401.

Li, N., T Li, and S Venkatasubramanian (2007). "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, pp. 106–115.

Lubbe, J.C.A. van der, M.J. de Boer, and Z. Erkin (2014). "A Signature Scheme for a Dynamic Coalition Defence Environment Without Trusted Third Parties". In: *International Conference on Cryptography and Information Security in the Balkans*. Springer, pp. 237–249.

Mankins, John C (1995). "Technology readiness levels". In: *White Paper, April* 6.

Maymounkov, Petar and David Mazieres (2002). "Kademlia: A peer-to-peer information system based on the xor metric". In: *International Workshop on Peer-to-Peer Systems*. Springer, pp. 53–65.

McCall, Becky (2018). *What does the GDPR mean for the medical community?*

Miller, Andrew et al. (2016). "The honey badger of BFT protocols". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, pp. 31–42.

Nakamoto, S. (2008). "Bitcoin: A peer-to-peer electronic cash system". In:

Nash, G. (2017). *Let's build the tiniest Blockchain*. https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b. Accessed: 29-10-2018.

Neal, Richard D, Philip L Heywood, and Stephen Morley (1996). "Real world data—retrieval and validation of consultation data from four general practices". In: *Family Practice* 13.5, pp. 455–461.

Otte, P. et al. (2017). "TrustChain: A Sybil-resistant scalable blockchain". In: *Future Generation Computer Systems*.

Paar, Christof and Jan Pelzl (2009). *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media.

Ponemon Institute (2016). *Sixth Annual Study on Privacy and Security of Healthcare Data*.

Presser, L. et al. (2015). “Care. data and access to UK health records: patient privacy and public trust”. In: *Technology Science* 2015081103.

Priisalu, Jaan and Rain Ottis (2017). “Personal control of privacy and data: Estonian experience”. In: *Health and technology* 7.4, pp. 441–451.

Ramachandran, D. (2017). *Epic software screenshot*. http://caduceusblog.com/wp-content/uploads/2017/08/Epic-WideScrn.jpg. Accessed: 01-10-2018.

Ross, R. (2016). *Lights and shadows of healthcare digitalization: Estonian experience since 2007*. http://www.marebalticum.org/brehca/images/stories/wis2016/wis2016keynoteross.pdf. Accessed: 02-10-2018.

Schnorr, Claus-Peter (1991). “Efficient signature generation by smart cards”. In: *Journal of cryptology* 4.3, pp. 161–174.

Shamir, Adi (1984). “Identity-based cryptosystems and signature schemes”. In: *Workshop on the theory and application of cryptographic techniques*. Springer, pp. 47–53.

Solat, S. (2017). “RDV: Register, Deposit, Vote: a full decentralized consensus algorithm for blockchain based networks”. In: *arXiv preprint arXiv:1707.05091*.

Spagnuelo, Dayana and Gabriele Lenzini (2016). “Patient-centred transparency requirements for medical data sharing systems”. In: *New Advances in Information Systems and Technologies*. Springer, pp. 1073–1083.

Stange, Kurt C et al. (1998). “How valid are medical records and patient questionnaires for physician profiling and health services research?: A comparison with direct observation of patient visits”. In: *Medical care*, pp. 851–867.

Szydlo, M. (2004). “Merkle tree traversal in log space and time”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, pp. 541–554.

Tobin, A. and D. Reed (2016). “The Inevitable Rise of Self-Sovereign Identity”. In: *The Sovrin Foundation*.

Vukolić, M. (2015). “The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication”. In: *International Workshop on Open Problems in Network Security*. Springer, pp. 112–125.

Wood, Gavin (2014). “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151, pp. 1–32.

World Economic Forum (2012). *Rethinking personal data: A new lens for strengthening trust*.

Yue, X. et al. (2016). “Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control”. In: *Journal of medical systems* 40.10, p. 218.

Zyskind, G., O. Nathan, and A. Pentland (2015). “Enigma: Decentralized computation platform with guaranteed privacy”. In: *arXiv preprint arXiv:1506.03471*.

Zyskind, G., O. Nathan, et al. (2015). “Decentralizing privacy: Using blockchain to protect personal data”. In: *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, pp. 180–184.