Bernardo Brás Gonçalves

Degree in biomedical engineering sciences

Secure storage and sharing of health data in a Blockchain environment

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Biomedical Engineering

Adviser: Pedro Vieira, Assistant Professor, NOVA University of Lisbon

Co-adviser: MSc Luís Curvelo, Marketing and Innovation Director, Compta

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

September, 2018

Secure storage and sharing of health data in a Blockchain environment

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

A escolha do tema da tese é sempre uma importante e difícil decisão. Para mim, não foi diferente. No entanto, a minha vontade de explorar novas áreas e abordagens levou-me a um tema completamente diferente e novo para mim, numa área que estava longe de ser a minha especialidade. E foi isto que me levou a reunir com o Luís Curvelo para perceber mais acerca do objetivo e tema da tese. Após essa reunião a minha decisão estava praticamente tomada. O Luís conseguiu demonstrar-me a importância desta tese e que a Compta era o sítio ideal, com as pessoas e meios certos, para fazer a minha tese. Por isso, quero agradecer-lhe por me ter dado a confiança para aceitar este desafio, inicialmente, e durante o trabalho para me manter focado, com a inesquecível frase: "Não se preocupe, ocupe-se". Adicionalmente, quero agradecer também ao Nuno Miquelina, ao João Soares e ao professor Pedro Vieira. O último responsável pela ligação entre a FCT e a Compta, e que, com a sua experiência em orientar teses de MIEB, conseguiu sempre ajudar-me e dar-me boas sugestões de modo a manter o foco desta dissertação na engenharia biomédica. Ao Nuno agradeço as ajudas e esclarecimentos importantíssimos para ultrapassar algumas dificuldades técnicas adjacentes ao trabalho desenvolvido. Por fim, agradeço ao João por-me ter introduzido e explicado a aplicação da Compta, MyXimi. Obviamente, não posso deixar de agradecer a todos os colaboradores da Compta com quem contactei e que me receberam muito bem.

Além disso, agradeço também à FCT e a todas as pessoas que dela fazem parte, por estes quase 5 anos sendo como a minha segunda casa. Em especial quero agradecer aos meus professores pela partilha de conhecimento técnico e não só. Dentro dos professores, quero destacar a coordenadora (durante a maioria do meu percurso em MIEB) do curso, professora Carla Quintão, por tudo aquilo que fez em prole do curso de engenharia biomédica e pelos seus alunos.

Durante toda a minha vida tive um grande apoio familiar que foi muito importante para tudo aquilo que já obtive tantp pessoalmente como academicamente. No entanto queria deixar uma especial palavra de agradecimento aos meus avós paternos pelo enorme apoio, especialmente durante estes 5 anos.

Não posso deixar de agradecer a todos os meus amigos e amigas, por todos os momentos em que não me deixaram trabalhar e por aqueles em que me mandaram trabalhar, na verdade, por tudo mesmo. Por fim, quero deixar uma palavra de agradecimento à minha namorada que me ouviu imensas vezes a refilar e a questionar-me sobre coisas que ela nem

fazia ideia do que eram.

Abstract

These days, the amount of digital health data is increasing. Legacy systems with privacy and security problems are used to store and share them. The blockchain technology appears as a disruptive solution to improve those systems. The main objective of this thesis work is to create a model based on the blockchain technology to securely store and share simple health assets.

The days when blockchain was only related with cryptocurrencies, like bitcoin, are long past. Today, blockchain is considered an important technology to all applications that need immutable and traceable data, a cryptographic and distributed ledger and secure transactions. MedRec and Medical chain are examples of solutions, based on the blockchain technology, to manage health records that have been developed.

The projected model describes a possible integration between MyXimi and a blockchain network. MyXimi users would be the terminals of the network. They will insert new health assets and get assets from the network. The network would consist of several distributed machines administrated by Compta. Those machines would store a public ledger of the health data transactions and all the sensitive data encrypted, always. As an application feature, management terminals would have access to anonymous health data for stats.

To test the main features of the projected model it was created a demonstration system: XBlock. Within this system were used several methods to increase the privacy and security of the health data, such as, data encryption and data masking. XBlock was presented to a board of business and IT specialists, in Compta.

In conclusion, XBlock proves itself as a true value to enhance the levels of security and privacy of health data when storing it and sharing it. As future work, is suggested the implementation of the propose model integrated with an health app. During this process, it's important to improve the blockchain network, in terms of its performance and scalability.

Keywords: Blockchain, Encryption, Privacy, Health data, Caregiver, Safe Sharing

Resumo

Atualmente, os dados digitais de saúde estão a aumentar. Os sistemas utilizados para os armazenar e partilhar têm problemas de segurança e privacidade. A tecnologia blockchain surge como uma solução disruptiva para melhorar estes sistemas. O objetivo deste projeto é a criação dum modelo baseado em blokchain que possibilite o armazenamento e partilha segura de dados de saúde simples.

A tecnologia blockchain teve origem no aparecimento das criptomoedas, em concreto, a bitcoin. Atualmente, é considerada uma tecnologia a ter em conta para todas as aplicações que necessitem de dados imutáveis e rastreáveis, dum registo criptográfico e distribuído e de transações seguras. A MedRed e a MedicalChain são dois exemplos de sistemas, baseados na tecnologia blockchain, criadas para gestão de registos de saúde.

O modelo projetado descreve a integração duma rede blockchain com a MyXimi. Os utilizadores da MyXimi serão os terminais dessa rede. Eles poderão aceder a ativos de saúde presentes nela e inserir novos. A rede consistirá em várias máquinas distribuídas, administradas pela Compta. Elas irão conter o registo público das transações de ativos de saúde e toda a informação sensível encriptada. Como condição da aplicação, os terminais de gestão da rede vão poder aceder a dados de saúde anónimos para a extração de conclusões estatísticas.

De modo a testar as principais características do modelo proposto foi criado um sistema de demonstração: XBlock. Neste sistema foram usados métodos de encriptação de dados e de mascaramento para aumentar a segurança e privacidade dos dados de saúde. O XBlock foi apresentado perante um quadro de especialistas nas áreas de negócio e TI, na Compta.

Em conclusão, Xblock apresenta-se como uma mais valia para o aumento dos níveis de segurança e privacidade de dados de saúde aquando do seu armazenamento e partilha. Como trabalho futuro sugere-se a implementação do modelo projetado integrado com uma aplicação de saúde. Nesse processo será importante melhorar a escalabilidade e a performance da rede blockchain.

Palavras-chave: Blockchain, encriptação, privacidade, dados de saúde, cuidador, partilha segura

Contents

List of Figures

List of Tables

Acronyms

AE	Asymmetric Encryption.
AES	Advance Encryption Standard.
API	Application programming interface.
BHD	Big Healthcare Data.
DDBMS	Distributed database management systems.
EC	European Commission.
P2P	Peer-to-peer.
RSA	Rivest-Shamir-Adleman.
SE	Symmetric Encryption.

Chapter 1

Introduction

The proliferation of health applications, smartphones and the digitization of the health records are hugely increasing the amount of health data available. This data is stored and shared in legacy systems, that have major problems of security, privacy and interoperability. Therefore, a solution is needed and, that's where the blockchain technology rises, as a new and disruptive technology to create more secure, more private and interoperable health systems. The main objective of this thesis project is to create a model, based on the blockchain technology, to share and store simple health data in a more secure way, maintaining full privacy and giving the owner full control of his data. Additionally, it's also necessary to access that health data anonymously, to extract important stats. The starting point of the work was the health application: *MyXimi*. The proposed model is to be associated, in the future, with this application, in order to improve its security and privacy. During this project was also made an additional effort to comprehend, not only all the technical specs of the blockchain technology, but also the impact of this emergent technology in the today's healthcare and businesses world.

The issue of cybersecurity and privacy of the so called Big Healthcare Data is real. In 2016, according to a report released by CynergisTek, the increase of hacking attacks to health systems was very relevant and, therefore, the number of breaches of sensitive data is rising. Thus, when talking about an application that collects and stores healthcare data, it's very important to have methods to ensure the security of the data and the privacy of the data owner. Therefore, in this thesis project, the blockchain technology and other encryption and anonymization algorithms will be used to ensure total privacy and security. These problems, in addition with the lack of interoperability between health systems of different health entities, like hospitals and clinics, are major obstacles to the existence of an integrate global network of healthcare data.

Biomedical engineering has many forms and areas, however its main goal is well defined,

which is integrating engineering sciences with biomedical sciences and clinical partice to improve human healthcare [1]. According to the Imperial College of London, biomedical engineering includes: "the development of new devices, algorithms, processes and systems that advance biology and medicine and improve medical practice and health care delivery"[1]. The propose of this work fits perfectly in this definition. Information technology is put to work for healthcare improvements, specifically in order to solve the problem of lack of security and privacy of health data.

In short, this thesis has three starting questions:

- Is the blockchain technology a valid solution to create a secure system of storage and sharing of the health sensitive data?
- What are the features of blockchain technology that will be important to use in the propose solution?
- Is it possible to access anonymous health data inside a blockchain network?

1.1 *MyXimi* application

MyXimi is a hybrid mobile application, developed by Compta, with the following propose: "fighting loneliness through gamification" [2]. The application was designed for people aged between 50-69 years. The app has two different main fields: social and well-being. Briefly, the social field consists in arranging activities with friends (challenge them, making them happier) and chat with them. In the well-being field, the application has access to six biometric parameters: blood glucose, blood pressure, heart rate, oxygen saturation, temperature and weight. The biometric data is presented in list or graphic form, to show the temporal evolution. This data can have two different sources: data imputed manually, or data automatically synchronized with biometric sensors, using Bluetooth. There is, also, a screen where the medicines prescribed to the user and his medical appointments are displayed. *MyXimi's* users can decide who can access their biometric data, by giving them an access key, randomly generated by the system. Only caregivers, with access to the caregiver application, can enter the key and access the information, they can, also, prescribe medicines and make medical appointments, automatically synced with *MyXimi*. The keys are shared via text message with the caregivers. A statistical analysis of the health data collected and inserted in *MyXimi* can be important to establish correlations related with the impact of mental and physical activity in an active and healthier ageing.

1.2 Global tasks

In order to achieve the major objective of this thesis, it's intended to accomplish 5 global tasks:

1. Comprehend the main technical valences of the blockchain technology and its impact in the current healthcare and businesses world;
2. Create a model of a blockchain network where all the healthcare data can be privately and securely stored and shared;
3. Develop the code to automatically manage the permissions and accesses to the healthcare data;
4. Write code to enable the extraction of anonymized data from the network;
5. Test the developed system.

1.3 Overview

This dissertation is organized in 7 chapters. In the current chapter, the objectives and proposal of the thesis were described. It was also given a brief explanation about the background information, which served as a base to the model and work developed in this thesis project.

- **Chapter 2** - *Background*: this chapter is divided in two different issues: the blockchain technology and big healthcare data. In the first section, the main versions of blockchain are explained, in order to comprehend the evolution of the technology and understand its main features. In addition, a brief section is dedicated to the impact of this technology in the current businesses. In the second section, the importance of privacy and security of the healthcare data is discussed, as well as some technical considerations about data encryption.
- **Chapter 3** - *Review of the literature*: based on the read papers, the main problems of the currently used databases and healthcare systems are described, and confronted with the valences of the blockchain technology. After that, some problems and challenges of this technology in healthcare are expressed, and also, some implementations and solutions, already developed by other authors.
- **Chapter 4** - *Ximi blockchain network model*: the model of a real implementation of a blockchain network in association with *MyXimi* application is detailed.
- **Chapter 5** - *Demonstration system*: the system, developed to demonstrate the valences of the model described in the previous chapter, is explained. All the main functions and algorithms are detailed.
- **Chapter 6** - *Simulation*: a simulation of the code developed is shown. The objective is to show the functionality of the system and its main features.
- **Chapter 7** - *Conclusions*: in this final chapter are compiled the thesis conclusions, limitations of the developed work and future directions.

Chapter 2

Background

In the current chapter are addressed the main concepts related with this thesis, essential to the comprehension of the developed work. Here, are described the technical specs, the main blockchain platforms and the impact of the blockchain technology. Additionally, it's given an overview about big healthcare data privacy and security.

2.1 The Blockchain Technology

The first contact of the scientific community with the blockchain technology occurred in 2008, when a white paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System"[3] was published. In that paper, Satoshi Nakamoto (a pseudonym for an unidentified person or persons), proposed a new cryptocurrency, the *bitcoin*. The main objective of *bitcoin* was solving the peer-to-peer (P2P) double-spending problem. The technology underlying the *bitcoin* was the first generation of the blockchain technology. Around 2014 the focus of the scientific investigation passed from bitcoin to the blockchain itself. This change resulted in different versions of the blockchain technology with many applications outside the financial world, developed since then [4]. Most importantly for this thesis, it resulted in blockchain applications in healthcare, like MedRec in 2016, and MedicalChain in 2018. Both will be discussed in chapter 3.

2.1.1 Bitcoin: the first generation of blockchain

In order to comprehend the most recent applications of the blockchain technology, more importantly those outside the financial world, it is necessary to first understand the main generations of blockchain implementations, beginning with *bitcoin* (*Blockchain 1.0*). As it was said before, the *bitcoin* was created as a response to the problem of double-spending. Double-spending is giving the same digital coin to more than one person. To solve this,

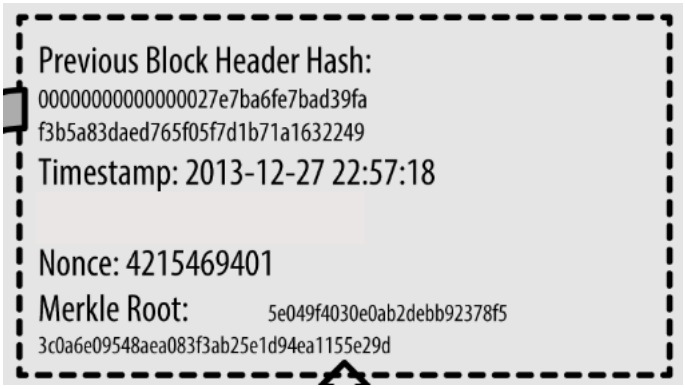

Figure 2.1: **Metadata contained in a block.** Each block in the blockchain contain: the hash from the previous block of the chain; the timestamp, time when the block was created; the nonce, essential to the proof of work algorithm and the Merkle tree root for the transactions included in the block. Adapted from [6].

Nakamoto, created a P2P network where every transaction is saved into blocks and every node of the network can see all the information in those blocks, that is a distributed P2P network. In addition, the existence of a distributed timestamp mechanism is also very important [5]. Each block in *bitcoin* blockchain has four pieces of metadata (figure 2.1):

- reference to the previous block: hash value of the previous block (hash-chain);
- the timestamp of the block;
- the Merkle tree root for the transactions included in the block;
- the nonce, arbitrary random number that can only be used once.

According to Andreas M. Antonopoulos, in *bitcoin* protocol, "merkle trees are used to summarize all the transactions in a block, producing an overall digital fingerprint of the entire set of transactions". Those Merkle trees are binary and contain cryptographic hashes of each transaction. The root is the first transaction that lead to the others included in the block [6].

To make a secure and trustworthy transaction the identity of both parties has to be confirmed. *Bitcoin* blockchain does that by using the public and private keys cryptographic protocol. "Public Keys are cryptographic padlocks in the form of unique complex strings of numbers"[7]. Each node has his public key, that every member of the network can see, however, it can only be unlocked by him, using his private key. The private key, just like its name suggests, it's private and a user only has access to his key. When a transaction is made, the public keys of both the parties are associated with it. Thus, the recipient and the sender identities are determined. To confirm that both parties are, indeed, fit for transaction, the sender will unlock his public key with his private key, proving is identity [7]. This process was adapted from the asymmetric encryption of data, that will be discussed in detail in section 2.2.1.

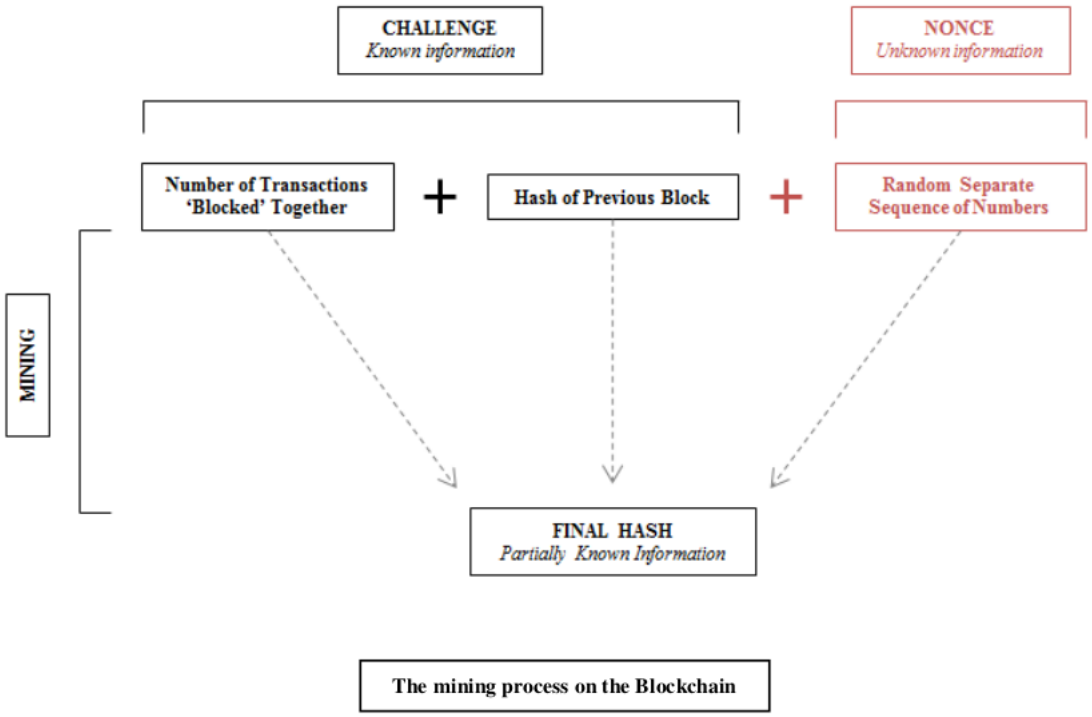

Figure 2.2: **The mining process in a blockchain.** There are three inputs of the predefined hash function, to create the final safest hash: the hash of the content of the block; the hash of the previous block and a random component, the nonce. In order to find the nonce that, combined with the other components results in a final hash, with given requirements, is necessary to implement a trial-and-error mechanism: trying a huge number of combinations until the right nonce is find. From [7]

To create a new block in bitcoin, it's necessary to solve a consensus algorithm. There are several consensus algorithms like: proof of stake; proof of burn, proof of elapsed time and proof of work. Bitcoin, uses proof of work [5]. The process of solving the algorithm, called mining, should be difficult, to discourage attempts of creating invalid blocks, but easy to check if is valid. The objective of mining is the creation of the safest hash for the next block to be created. A hash is mathematical cryptographic output, of fixed size, (see figure 2.1) from the process of compressing a certain input of any type. The function that transform any input in a hash is the hash function. In a blockchain the input is a list of transactions (all to be recorded in the block) and the hash function is defined by the blockchain platform in use, like the hash requirements are. In the figure 2.2, is shown the components to create the final safest hash: the hash of the content of the block (each transaction has its hashes), the hash of the previous block and a random component, the nonce. To solve the proof of work algorithm it's necessary to find the right nonce that, when combined with the other hashes (from the previous block and from all the transactions in that block), results in a certain final hash, the safest one, with certain requirements. An example of requirement is having a set number of zeros in the prefix [7]. When a node solves the proof of work algorithm, the final nonce is automatically broadcasted to all the

members of the network. They try the nonce for themselves and if the final hash is the same for every node, then all the transactions are verified, and the block can be sealed. As an incentive, the faster node to solve the algorithm receives some amount of *bitcoin*.

If someone change a transaction (could be malicious actor), all the subsequent blocks must be recomputed, which requires a prohibitive amount of computational power. Therefore, a blockchain is immutable and that makes it much more secure and transparent.

All these mechanisms are accomplished without a central intermediary, originating a decentralized and distributed network. This feature prevents the single point of failure - if there was a central authority, and if it was compromised or attacked, by any reason, all the network was also compromised [5]. Also, the nonexistence of a central intermediary reduces the costs of the network [8].

2.1.2 Ethereum, Hyperledger and other platforms

After the creation of *bitcoin* many other crypto-currencies have been created and, the most valuable of them was created by Vitalik Buterin, in 2014, and it's called Ether. Ether has the second largest market cap (numbers from April 2017), the largest is *bitcoin*. However, Ethereum, Ether's platform, is also an open blockchain platform with a built-in programming language (solidity) that allows the creation of "smart contracts". Within that platform it is possible to build and deploy decentralized applications (called Daaps), using "smart contracts", that run on Ethereum's blockchain network. "Smart contracts" are programs that execute predefined actions when certain conditions within the system are met, like an expiration date or strike price. Those contracts are in the public ledger, so they are visible to every node in the network. They allow the establishment of contracts between "trustless" parties, if all requisites to the contract are coded into the blockchain. Ethereum blockchain is a public network and uses the proof of work consensus algorithm [9].

Another important blockchain platform is the Hyperledger Fabric. This platform is part of the Hyperledger project, initiated by the Linux Foundation in 2015. Fabric is a "platform for distributed ledger solutions underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility, and scalability". In this platform there is no single blockchain standard - the community can develop different version of blockchain technologies, for example users can choose the consensus algorithm. Besides that, Fabric has its own version of smart contracts: chaincode, that can be developed using one of two programming languages: Go or Node. The chaincode make the connection between the blockchain ledger and external applications. The shared ledger of Fabric has two components: the world state and the transaction log. The first one is the database of the ledger, it describes the state of the ledger at a given point in time. The second is a record of all transactions that contributed to the current value of the world state. Another novelty of this platform is the possibility of creating channels. The channels offer the ability to allow a group of participants to create a separate ledger of transactions, that no one else can

access, it's like a "sub-private" blockchain [10].

Blockchain is also seen as a new form a distributed database or ledger to make transactions of arbitrary metadata. MultiChain and BigchainDB are examples of distributed ledgers with the capacity to store metadata of different sizes. This type of blockchain combined with "smart contracts" is known as blockchain 2.0 [5].

The blockchain 3.0 denote nonfinancial applications of the blockchain technology. This version keeps all the basic features of a blockchain, that is, a distributed P2P ledger, transparent, immutable, secure and with the capacity for "smart contracts", although, it doesn't have a cryptocurrency associated [5].

Platforms like Ethereum or *bitcoin* are "permissionless" or public blockchains, however, there are permissioned or private blockchains, like Hyperledger Fabric, that can be much more interesting outside the financial world. As the name says in a public blockchain everyone can be part of the network, in private blockchains the members need to be accepted by the community. In a public blockchain network it's necessary to validate all transactions, because the users of the network could be malicious. On the contrary, in a private blockchain it may not be necessary to validate transactions, so the mining process can be neglected.

In conclusion, "a blockchain is a P2P distributed (and decentralized) ledger forged by consensus, combined with a system for "smart contracts" and other assistive technologies" [11]. The next chapter will clarify the benefits of using blockchain technology in health systems and describe some applications that already exist on blockchain for health.

2.1.3 Blockchain: a growing technology

Blockchain is a recent technology, however, due to its disruptive nature and enormous potential, the number of companies studying it and developing new solutions based on it is growing fast. In the figure 2.3 is shown the sectors currently using blockchain as part of their commercial solutions. The shown graph is based on a study made by the Cambridge Centre for Alternative Finance, in 2017, using a list of 132 blockchain use cases, in Europe. The figure shows the existence of emergent sectors, like healthcare and entertainment.

In 2017 was released a facts sheet by the European Commission. The document explored the main benefits of using blockchain technologies and how could Europe help the industry adopting them. In that year, €83 million have been allocated, by the EU, in blockchain related projects. It's expected that in the next years (2018 to 2020) that amount can rise to €340 million. This shows the huge interest in this emergent technology. Over the year of 2017 many blockchain solutions were tested (through proof of concept) and projects in support to EU policies were piloted. In that document was also stated the importance of creating a EU Blockchain Observatory and Forum to map relevant initiatives, share experiences and organize debates [12].

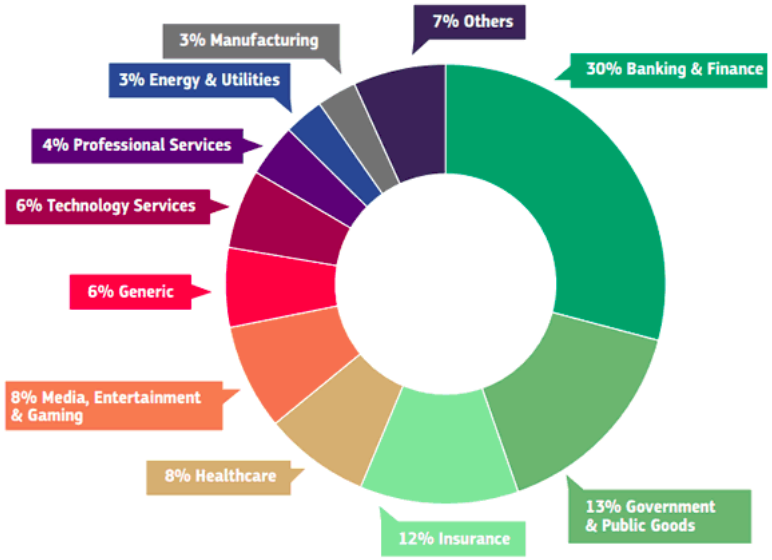

Figure 2.3: **Sectors currently using blockchain.** As expected the banking and finance sectors have the largest percentage of blockchain use cases, followed by the governement and insurance. From [12].

In terms of global impact of the blockchain, the International Data Centre predicted, in October 2017, that "by 2021, at least 25% of the Global 2000 (the world's biggest public companies) will use blockchain services as a foundation for digital trust at scale". Another relevant fact was revealed by a blockchain enterprise survey made by Juniper research, in 2017, to almost 400 large companies. The results concluded that "nearly 6 in 10 (57%) large companies are either actively considering, or are in the process of, deploying blockchain technology"[13].

Specifically, in healthcare, the blockchain impact is also relevant as shown in the IBM Institute for Business Value blockchain study that surveyed 200 healthcare executives, in 16 countries, in 2016. From that study can be concluded that about 16 % of the healthcare executives expect to have a commercial blockhchain solution at scale in 2017. Additionally, 9 in 10 are planning to heavily invest in blockchain pilots over 2018 [14].

2.2 Big Healthcare Data: security and privacy

Nowadays the amount of digital health data is growing fast, generating the so called Big Healthcare Data (BHD). This fact is closely related with the proliferation of software applications and mobiles devices and, of course, the digitization of medical records and patient data. As any other type of big data, BHD can be very useful and important. It can "improve patient outcomes, predict outbreaks of epidemics, gain valuable insights, avoid preventable diseases, reduce the cost of healthcare delivery and improve the quality of life in general" [15]. Despite all the inherent advantages of using BHD, there are issues related with the security of the health data and the privacy of the patient that are raised, and they need to be dealt with. In fact, according to a report released by CynergisTek: in 2016, hacking attacks on healthcare providers were increased by 320 % and 81 % of health data breached came from hacking attacks specifically. The usage of legacy medical hardware and software, the negligence of the cybersecurity problem and the interconnected health data nature are some factors that potentiate those attacks. The stolen healthcare data can be used for blackmail or to create medical frauds, and unlike account numbers or credit cards, medical records and patient data never change. Thus, BHD is valuable, not only because its richness but also because of its longevity and, therefore, it must be secure [16]. As was mentioned in the previous chapter *MyXimi* applications collect health data, thus it must be secure and private, just like any other health system, but off course, on a smaller scale.

About the issue of share and access of healthcare data, the European Commission, has three main priorities [17], as is written in the referred document, originally published in 2014 and updated in 2018. They are:

- Secure access and exchange of health data.
- Health data pooled for research and personalized medicine.
- Digital tools and data for citizen empowerment and person-centred healthcare.

In this document is stated the ambition of having the health data available to its owners and caregivers all across the European union. Only possible with total security, privacy and interoperability between different systems. Another ambition is using a larger amount of health data for research as well as give to the patients total control of their data [17].

To enter in a more formal and technical discussion about privacy and security in healthcare, it's necessary to first define these concepts. Privacy is "defined as having the ability to protect sensitive information about personally identifiable healthcare information"[15]. On the other hand, security is "defined as the protection against unauthorized access, with some including explicit mention of integrity and availability"[15]. Wherefore, privacy focus on the utilization and management of the individual's personal data and, security on protecting data from malicious attacks.

To improve the security of BHD there are available several methods [15]:

- **Authentication.** Authentication is important to ensure the identity of a certain user, to secure the access to networks and protecting the identities of the users.
- **Encryption.** Encrypting data prevents from unauthorized access of sensitive data.
- **Data masking.** To mask some data is to replace the sensitive data with an unidentifiable value. There is no way back, that is, after the mask, the original value cannot be retrieved.
- **Access Control.** Even after authentication the movements of users in data systems need to be controlled and secured, based on their privileges and rights determined by the owners of the data.
- **Monitoring and auditing.** It's important to gather and investigate all the network events to catch possible intrusions.

In this thesis project the focus will be specially the data encryption, however, all the other methods are applied.

In favour of improving the privacy of BHC, data protection laws have an essential role, and, on the side of the system providers, anonymization algorithms are important, as well. There are laws and regulations in every country for data protection. In truth, this year, it's been happening a reform of EU data protection rules, on sequence of the revelations about Facebook/Cambridge Analytica. Thus, with this new set of rules to all the companies, the European Commission (EC) order that companies: improve the language in which they explain their privacy policies; wait for the affirmative consent of the user before use his data; clearly inform the user about any transfer of his data; well-define the purpose to the usage of the users' data and inform about any decision making algorithms related with the users' data. Additionally, users had won more rights: the right to be inform of data breaches; to move their data; access all the data that a business as on him and the "right to be forgotten" [18].

2.2.1 Data Encryption

The main objective of data encryption is to protect sensitive data from unauthorized access. In order to achieve that, the data is translated into *ciphertext*, unreadable encrypted data. The only way to access the *plaintext*, that is, the sensitive data in its initial form, is to decrypt the *ciphertext* with a decryption key. The process of creating the ciphertext consists in the application of an encryption key and an encryption algorithm [19]. There are two major types of data encryption: asymmetric encryption (AE) and symmetric encryption (SE).

These two types of encryption differ, mostly, in the number of cryptographic keys and in their usage. In AE, there are a pair of keys mathematically linked, one is a public key and the other is a private key, because of it, this type of encryption is also called public-key encryption. If the plaintext is encrypted using the public key, then the private key will be the decryption key, or vice versa. In SE, there is only one key, a secret key to encrypt and decrypt the plaintext. Despite been much faster than AE, in SE it's needed to exchange the secret key with the recipient before he can decrypt the ciphertext, which can be a problem. Thus, with AE the keys are more protected, therefore, it's more difficult to discover the decryption key. The encryption strength is directly proportional to the key size. Despite being more secure, longer keys need more computing resources to be created, so, it's necessary to make a commitment between the size of the key and the computational power to create it. The most common way to break encryption is by using brute force attacks. In those attacks, several random keys are tried, until the right one is reached. Brute force attacks are less successful with larger keys, logically. Other types of attacks explore weaknesses in the cipher (cryptanalysis) or in the implementation of the cipher (Side-channel attacks). Those attacks are more likely to be successful if there is a flaw in the cipher or in the implementation of the cipher [19].

Currently, the most used algorithm of SE is AES, Advanced Encryption Standard, also known as Rijndael. This algorithm was chose by NIST, in 2000, to be the US's new encryption standard. It is used in archive and compression tools (RAR, WinZip); disk/partition encryption (BitLocker, FileVault); Virtual Private Networks (ExpressVPN, NordVPN) and other mainstream applications, like WhatsApp or Facebook Messenger. AES encrypts data on a per-block basis so, it belongs to the family of block ciphers. These ciphers produce a block of ciphertext for each block of plaintext. Thus, if the block is 256 bits long, then, for every 256 bits of plaintext, 256 bits of cipher text are produced. The ciphertext is produced iteratively and number of iterations depend on the length of the key, for example, a 256-bit key has 14 rounds [20]. In each round there are four steps of algebraic operations, such as substitutions byte-by-byte and permutations, between the bytes-matrix of plaintext and bytes-matrix of the key. The AES keys can be 128, 192 or 256 bits long. With a 128 bit there are $2^{128} = 3.4 \times 10^{38}$ possible keys. So, a PC that tries 2^{55} keys per second, will take 149.000 billion years to break AES cipher. Using the longer keys will increase the number to impossible values, considering the computational power currently available [21].

In terms of asymmetric encryption, the most known algorithm is the RSA (Rivest-Shamir-Adleman) algorithm. This algorithm was first described in 1977, but was only released to public domain in 2000, by RSA Security. It is used in many network protocols, like SSH and SSL/TLS and in email encryption algorithms, like OpenPGP and S/MIME [22]. The RSA cipher is based on the fact that is quite easy to multiplicate two large numbers, but factoring large numbers is very difficult [23]. The creation of the pair of keys is the most complex process of this algorithm, and it is what makes him very different of

the symmetric algorithms. These are the mathematical steps to create the public key [22] [23]:

1. 2 large prime numbers are generated, p and q;
2. modulus n is calculated by multiplying p and q, $n = p \times q$.
3. the Euler's totient function of n is calculated ($\phi(n) = \phi(pq) = (p-1)(q-1)$) and a relatively prime to it is chosen, called public exponent, e, normally 65537;
4. n and e are the components of the public key, PubKey(n,e).

n is the link between the two keys, public and private, and its length, in bits, is the key length. To create the private key these are the steps [23]:

1. calculate the modular inverse of e modulo $\phi(n)$, $d \times e \equiv 1 \ (mod\, \phi(n))$.
2. in the prior equation, d is the private key, PrivKey(n,d)

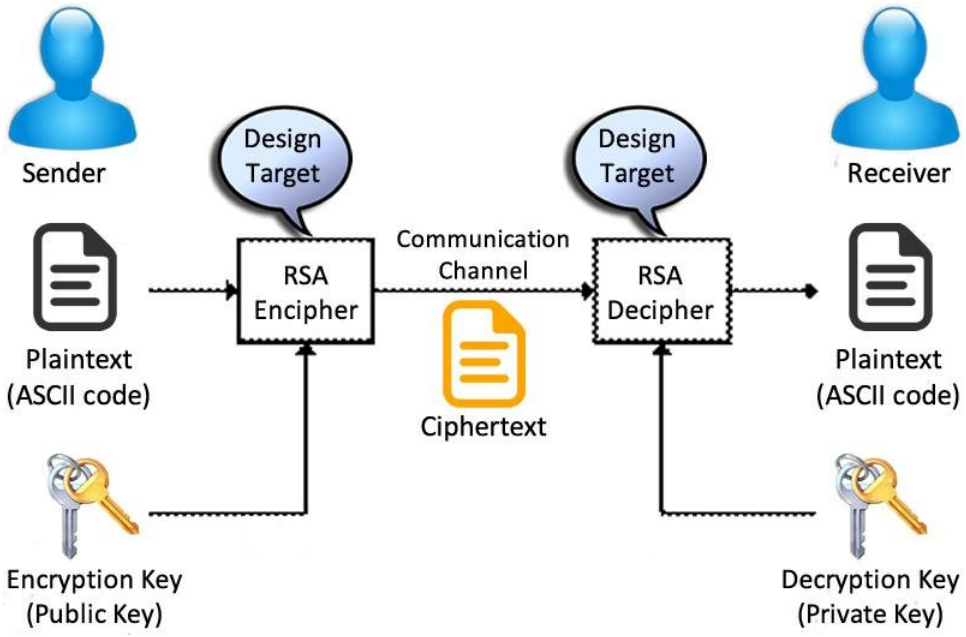

Figure 2.4: **Overview of the encryption/decryption process of RSA algorithm.** The sender encrypts the message with the receiver's public key. The receiver decrypts the cyphertext with his private key. Adapted from [24]

.

After the creation of a pair of keys, the message can be encrypted using one of the keys, normally a public key, and decrypted using the other. In figure 2.4 is represented a general view of the encryption and decryption process. The encrypted message needs to be smaller than the key. Encryption and decryption consists in these steps [23]:

1. message is converted to bytes, m;

2. the sender calculates $c \equiv m^e \ (mod\, n)$, where c is the ciphertext;

3. the receiver computes $c^d \equiv m \ (mod\, n)$, therefore retrieving m;

4. at last, the receiver translates m to plaintext.

Just like in symmetric encryption, the strength of RSA is related with the key size (n). The newer keys are 4096-bit and are the most secure [23].

Another important feature of RSA algorithm is the capacity of digitally sign a message. With this feature it's possible to confirm the origin of the message. To accomplish that, the sender of the encrypted message signs the message with his private key. The receiver, to confirm the origin of the message, verifies the digital signature with the public key of the sender. The first step of the creation of a signature is the creation of a hash of the message [25]. To create the hash is necessary to apply, to the message, a hash function, the most used is SHA-256. When applying this function, the message is reduced to an unique value, 256 bits long. Figure 2.5 gives an overview of process of signing and verifying a message.

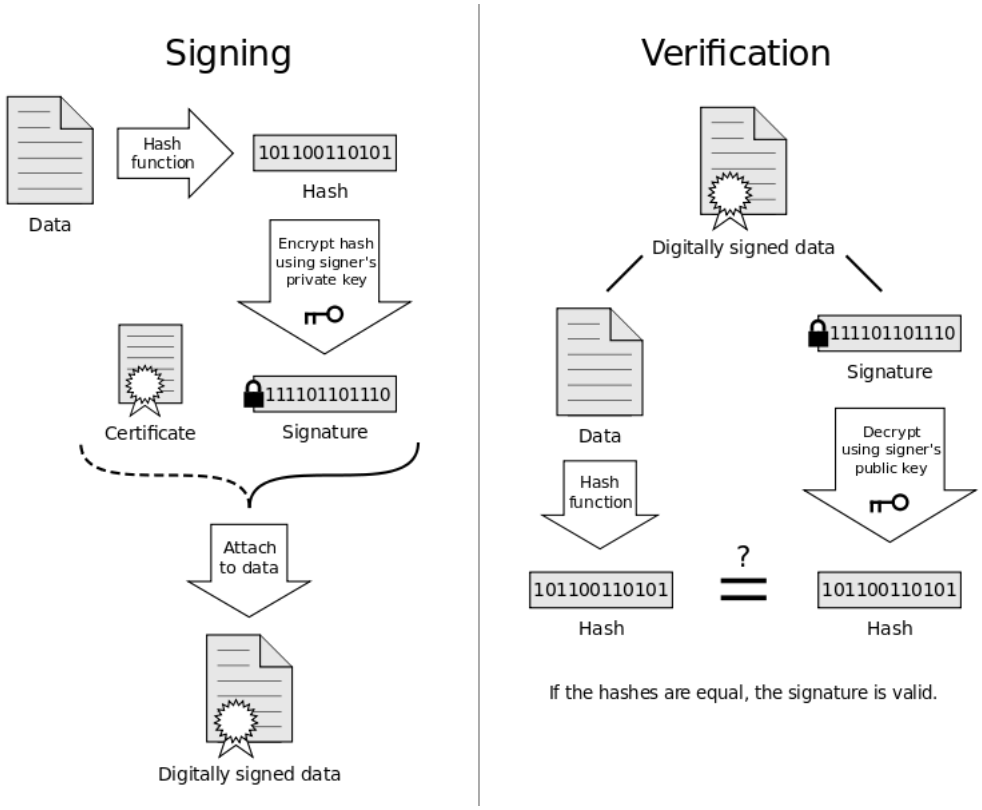

Figure 2.5: **RSA algorithm for signing and verifying messages.** Data is signed with the private key of its owner and verified with the public key of the owner. From [26].

To conclude, the keys of RSA have to be larger than AES keys, in order to be secure, so AES keys are faster to create and use less computational resources. AES key need to be distributed, which can be a problem, with RSA there is no such problem. RSA encryption has a greater message size limitation. With RSA it's possible to direct the message to a

certain recipient. Therefore, these two algorithms can complement each other, if they are both used at same time, for example, using asymmetric encryption to distribute symmetric keys.

Chapter 3

Review of the literature

The first sections of this chapter are based on revision articles about blockchain technology applications as a database and as an alternative to health systems. In the last section are analysed some blockchain implementations in healthcare.

3.1 Traditional databases vs blockchain

Currently, traditional distributed database management systems (DDBMS) are used to store biometric data. Those databases are centralized, although, the data is distributed by many computers. DDBMS synchronize all the data periodically, as if it were all stored in the same computer, so they are logically centralized [27]. Within those systems there are two major types: Structured Query Language (SQL)-based systems, like Oracle, and NoSQl-based systems, like Apache Cassandra, CouchDB and PouchDB, the last ones are used by MyXimi [5]. Given the objective of this thesis it's important to realize what are the advantages of using the blockchain technology as a distributed ledger, instead of the traditional databases. Those advantages can be grouped in five central points [5]:

(a) decentralized management;

(b) immutable audit trail;

(c) data provenance;

(d) robustness and availability;

(e) security and privacy.

As it was said before, blockchain is a P2P, decentralized database management system (a): every node has a copy of the data and runs independently while following the protocols.

Decentralized management makes blockchain technology suitable to integrate applications in which healthcare stakeholders wish to collaborate without a central management, which could cause some friction and increase the costs. On the contrary, DDBMS are centralized, which makes them susceptible to the problem of single point of failure [5].

Changing the data, already inserted into a blockchain, is nearly impossible, due to the cryptographic links between its blocks (b). Thus, it's said that blockchain only supports read and create options. So, blockchain has the capacity to hold critical information, without the risk of it being deleted or changed. In contrast with traditional databases, which support create, read, update, and delete functions, managed by a central authority [5].

In a blockchain database the ownership of the digital assets can only be changed by the owner (c), following the cryptographic protocols. However, on DDBMS, the ownership can be changed by the central authority. In addition, in a blockchain the origin of the data can be traceable: it's possible to confirm the sources or the records. For this reason, blockchain is applicable for managing critical digital assets [5].

Both DDBMS and blockchain database management systems are distributed, however, it takes much cost to the DDBMS to achieve the high level of data redundancy blockchain does. Therefore, blockchain is a good way to store records that necessarily need to always be available and preserved (d), such as electronic health records of patients [5].

Finally, blockchain uses cryptographic algorithms that improve the security and privacy of the data (e): encryption algorithms, hashing algorithms, for example (see 2.2). Additionally, blockchain users are identified by a generated hash, rather than, for example, an IP address. Other algorithms can be used to ensure the ownership of the digital assets, by generating and verifying the authenticity of the public and private keys as digital signatures [5].

3.2 Healthcare traditional systems vs blockchain

In the recent years, the awareness of the blockchain key advantages mentioned above has been growing. This fact results in an increased number of studies, related with blockchain distributed ledger technology applied to health and biomedical care. In many of those studies were developed new applications that can be categorized based on their objectives as: enhance insurance claim process; improved medical record management; accelerated clinical/biomedical research and advance biomedical/health care data ledger [5]. Those applications try to solve the existing problems in the traditional healthcare and health information exchange systems.

Traditional health systems are centralized: there is an intermediary who controls and stores all the data [8]. Blockchain is forged by consensus, so every node helps to decide what information is traded and to whom and, of course, every node has a copy of the data stored in the blockchain. In addition, decentralized systems reduce the costs of transactions.

Healthcare records are considered critical information and imply high privacy and confidentiality. However, sometimes, it's important to share them with someone. With the current healthcare systems this can be difficult, given their lack of interoperability. Most of the systems have not compatible data types, which forbid the exchange of data [8]. Blockchain systems are "based on open source software, commodity hardware, and open API's (application programming interfaces)" [28]. An open source blockchain for healthcare enhances interoperability between systems and, it's more efficient handling large volume of data and more users. Open API's facilitate the exchange and integration of data between the blockchain applications and other systems. The blockchain distributed ledger enables near real time updates in all nodes [8], simplifying the data exchange in the same network.

Blockchain "smart contracts" have a major role in assuring the privacy and confidentiality of the blockchain network. In the current health systems there are inconsistent rules and permissions that constrain the process of a certain health stakeholder accessing the data of some patient [8]. With "smart contracts" it is possible to create a set of rules to control the access to the patient's data [8]. By doing that, the patient, owner of the medical data, can choose with whom he wants to share is medical information. Besides that, "smart contracts" are one of the major features of blockchain technologies, that critically contribute to the enhancement of decentralize management, because it allows rules and permissions to be written in code, therefore fulfilled automatically without a single central entity to control it.

3.3 Blockchain challenges

As any other technology still in expansion, blockchain, and especially, blockchain in healthcare, has some challenges to overcome. The first one is related to transparency and confidentiality [5]. Transparency can be one of the greatest benefits of blockchain, because it enhances trust, yet can be dangerous in healthcare, if not safeguarded with a good implementation of "smart contracts" to manage permissions. In a blockchain network everyone can see the metadata inherent to a transaction of information, and in some cases this is critical private data. Another issue is related with the "pseudonymity" of the users, in other words, despite the blockchain users be identified only by hash's, they can be reidentified through analysis and investigation of the public metadata contained on the transaction with certain user's hash, thus blockchain can provide only pseudonymity and not total anonymity [5].

The second challenge and biggest one is related to speed and scalability [5]. As explained in the previous section, to ensure the creation of a new block, i.e. validate all the transactions inside it and write them into a block, it's necessary to solve the consensus algorithm, which can take some time. This speed constrain might affect the capacity of the blockchain of being scalable. As an example, Visa has a theoretical maximum speed of transactions of 4000 per second, on the other hand, Bitcoin only has 7 per second [5].

In Apache Cassandra database, a traditional DDBMS, the throughput increases with the increase in the number of nodes of the network. However, in the bitcoin blockchain that doesn't happen, the throughput doesn't change with the increase of the number of nodes [29]. The speed of the transaction isn't only important to scalable blockchain applications, it's also paramount for the construction of real time distributed ledgers.

The last challenge is the threat of a *51% attack* [5]. This challenge is common to all types of blockchain applications, specially public blockchains. It refers to the non zero probability of existing fewer honest nodes than malicious ones. In these cases, the malicious nodes have more computational power than the honest ones, thus, when two blocks, one malicious and the other honest, are competing to the next spot in the chain, the malicious will win. Hence, the malicious nodes take control of the network, by consensus [5].

3.4 Blockchain solutions and implementations

After realizing some challenges and issues of a potential healthcare blockchain system, it's necessary to walk through some proposed solutions and implementations, that intend to overcome the previously described points.

There are two major approaches to improve the scalability of the system (with the same objective of reducing the quantity of data stored in the blockchain): use the blockchain only as an index of health data [28] or use it only to store ongoing verified transactions [30]. Rather than storing every information, every medical record, in the blockchain, which would have data storage implications and data throughput limitations, the authors, recommend using the blockchain as an index of healthcare data, a list of all the user's health records and data [28]. To do that, each transaction inside a block would contained the user's hash; an encrypted link to the health record; a timestamp and, to improve the data access efficiency, the type of data of the medical record and other metadata that would facilitate possible queries. The encrypted link work as a pointer to a data repository, where is all the medical data, called Data Lake [28].

Data lakes are scalable data repositories holding data in its natural format. They use a flat architecture, and each data element is assigned with an unique identifier, tagged with metadata [31], encrypted and digitally signed [28]. These repositories support interactive queries, text mining and analytics, and machine learning [28]. However, they are centralized which make them susceptible to the single point of failure problem. In [28] was proposed a general mechanism for recording medical/health data into a blockchain and data lake, that can be visualized in figure 3.1. With this mechanism it's possible to take advantage of two different ways of acquiring health data: by wearable sensors and mobile applications, and by medical records made by health professionals. To store the data securely, all the data must have a digital signature to verify its information. After that the data would be encrypted and sent to the data lake for storage. When the data is stored, a pointer in the blockchain is registered along with the user's hash and, then, is sent a notification to the user/patient with the information that new medical data was added to his blockchain

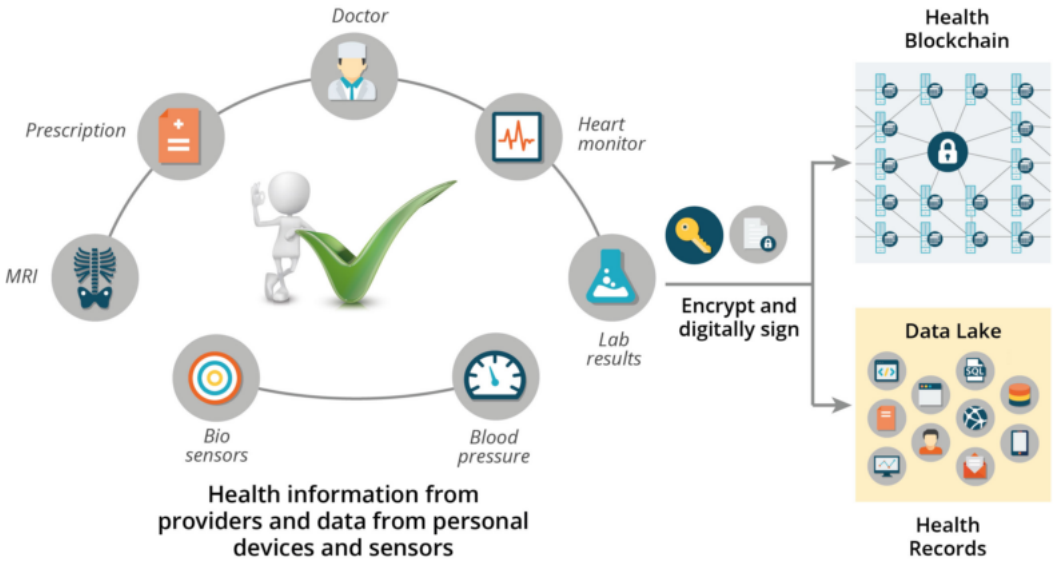

Figure 3.1: **Proposed mechanism of storing health data inside the blockchain and data lake.** Health data with different origins - medical records, like magnetic resonance images, prescriptions, etc.; personal devices, like blood pressure, heart rate, etc. - are encrypted and digitally sign. Then, they are stored in the data lake, in their native form, and in the blockchain as a pointer to the data lake, with some metadata. Therefore, blockchain becomes a list of all health records of the patient and data lake stores all the raw data from the healt records. Adapted from [28].

[28]. Keeping in mind this mechanism, the authors, proposed that the users could use their personal mobile devices to give permission to a caregiver to access their health data. With that permission the caregiver could decrypt and authenticate the digital signature, accessing to the data lake through the blockchain. Additionally, the user would be able to see the list of the users that access his blockchain [28].

MedRec [32], is prototype system to manage the electronic health records, using blockchain technology, developed by the MIT media lab. The objective of this project was to create a patient-centred management system, as a response to four critical problems: fragmented, slow access to medical data; system interoperability; patient autonomy; improved data quality and quantity for medical research. A block, in MedRec, represents data ownership and viewership permissions shared by all the members of a private blockchain network. The MedRec blockchain utilizes "smart contracts", using an Ethereum blockchain, to automate and track changes in permissions or creation of new records. This is done by establishing a patient-caregiver relationship, which associates a medical record from the patient with viewing permissions and pointers to the raw data for the caregiver, to use in external databases. By logging these relationships, the caregivers can add a new record from a specific patient. Patients can share their data with caregivers, generating a notification and a verification, in both cases, to accept or reject the changes made [32].

Like [28], all the raw health data is in an external database. However, each MedRec user has his database with the data that he has the permission to view. A syncing algorithm manage the exchanges of data off-chain, between the caregivers' database and the patient's database [32]. The MedRec blockchain is private, i.e. only pre-approved nodes can access the network [32]. This way the malicious nodes are kept out of the network, at least most of them, thus, the probability of a *51 % attack* is minimal and the extraction of frequency-based insights from the blockchain records is blocked. The implementations of MedRec are reliable solutions against the problems of transparency/confidentiality, scalability and the *51 % attack*.

Another proposal by MedRec's authors was the usage of the health data inside data lakes, similar repositories, or even in chain, as mining rewards to medical research entities, in order to invite them to mine the blockchain network [32]. This process would be like in the Bitcoin network [3], the nodes that mine the chain, by solving the proof of work algorithm, receive their reward: a certain amount of cryptocurrency. According with the creators of MedRec, healthcare stakeholders and medical researchers should mine the blockchain. As a reward, the health data owners should release the access to their data, in anonymized form. This process brings a revolutionary way of gather and access data for research proposes, and with that data it's possible to create better insights about the medical treatment and healthcare outcomes [32].

Lastly, it's important to describe an ongoing project, called Medical Chain. This promising project started in 2016 and the first prototype was released in mid-2017. Later that year a partnership with the Linux Foundation was established. Early in this year (2018) a beta platform was launched. The main objective of this project is the creation of a single true version of user's health data. Additionally, enabling the users total control of their data, allowing them to give conditional access to their data to different healthcare agents such as doctors, pharmacists or hospitals, is also an important objective. To make those interactions fully secure, transparent and auditable it was used the blockchain technology, specifically, two well-known blockchain platforms: Hyperledger Fabric and Ethereum. Thus, Medical Chain platform is a private and permission based blockchain using a dual structure. Hyperledger Fabric is used to manage the permissions and access to the health data. Ethereum is used to create a token and enable other applications to run. Besides the distributed network of secure health records, the Medical chain is also developing a doctor-to-patient telemedicine application (remote medical consultations) and a health data marketplace. The marketplace will be an opportunity for the users to make money ("Medtokens" - Medical Chain tokens) granting access to their medical data to third parties for example, to medical research companies.

All the health records are encrypted using a symmetric key. Afterwards, they are stored off chain, in local data stores. The figure 3.2 shows with detail the connection between the data store and the ledger (blockchain). The management of symmetric keys is done using asymmetric encryption directed to specific entities [33]. Thus, when a certain entity

has permission to access a health record from patient, that record is decrypted with the patient's private key. Then the symmetric key, which encrypts the health data, is encrypted with the public key of the authorized entity. Hence, that symmetric key can be decrypted using the private key of the entity and, therefore the symmetric key is ready to decrypt the health record.

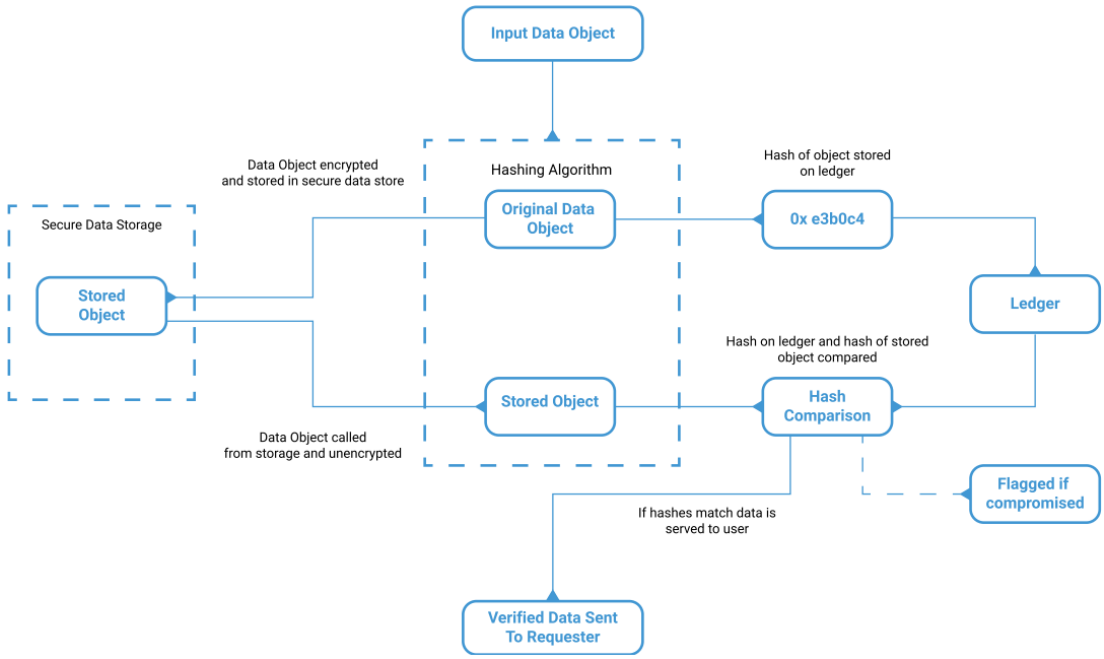

Figure 3.2: **Encryption cryptography from Medical Chain.** The health record is stored encrypted in a secure data store. The hash of the object is registered in the blockchain ledger. The hashed from the ledger and from the data store, after decryption, are compared in order to verify the data. From [33].

At this stage all the main features of the blockchain technology had been explored, as well as some real blockchain implementations in healthcare. In the upcoming chapters will be presented this thesis project work. First, it will be explained a model, that intends to describe the interaction between a blockchain network and the MyXimi app. With this interaction would be possible to elevate the level of security and privacy of the app. The presented model was inspired on the solutions described in this chapter. It intends to utilize the most important features of the blockchain technology, and suppress the features that aren't crucial, to assure a secure and private environment to share and store health data. After, the code developed to demonstrate the model and its simulation will be detailed.

Chapter 4

Ximi Blockchain network model

This chapter has the objective of describing the proposed model of interaction between MyXimi application and a blockchain network. This model would be the basis to a proper implementation of a blockchain network, to enhance the security and privacy of MyXimi.

4.1 Overview of the model

The proposed model has three main elements: MyXimi terminals, Ximi network and management terminals.

Ximi network is the centre of the system, in which are all the health assets (encrypted) of all MyXimi's users. The other elements interact with the network, to get access to health assets and to insert new health assets. Figure 4.1 illustrates the interaction between all the elements of the system.

As referred in chapter 1 the main objective of this thesis is to propose a model, based on the blockchain technology, to ensure the security of the health assets and the privacy of its owner. In favour of this goal, Ximi network is a blockchain network. It is a P2P network, decentralized, with cryptographically linked immutable blocks and encrypted data. All the health data inside the network is encrypted. That health data is set of several health assets, or health measures. The constitution of the health assets will be discussed in chapter 5.

Ximi network has distributed blocks that all the nodes of the network can see. In a first approach, all the nodes of the network would be machines from Compta, geographically distributed to avoid a single point of failure. In future, would be necessary to invite health stakeholders or government nodes to expand the network. In any case, the network would be a private network. Although, Ximi network consists in a certain number of physical machines, it is seen like a cloud by the users, for them there is no distinction between different nodes of the network.

Only MyXimi users and caregivers can use the network, to insert encrypted health assets or to access fully decrypted health assets, respectively. Besides that, the network providers, in form of management nodes, can access total anonymous health data to make health stats. All MyXimi's terminals have a pair of RSA public/private keys, that will be an important part of the data encryption process. All the public keys of the users are stored in a repository.

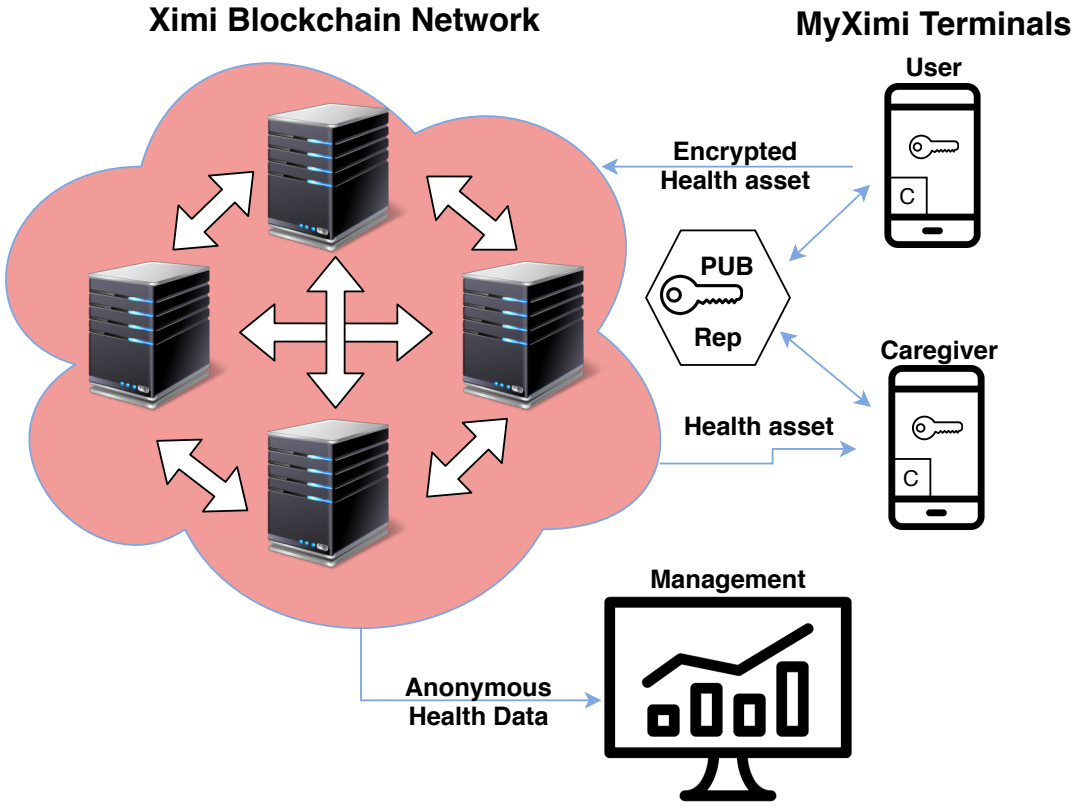

Figure 4.1: **General view of the association MyXimi/Blockchain network.** The blockchain network consists in multiple distributed nodes that store all the health data, encrypted. Ximi users interact with the network to access health assets and insert them. There are management apps to access anonymous data, only available to the network owners. The public key repository, stores all the public keys of all MyXimi users.

4.2 New health asset

The health assets in Ximi network are physiological measures, acquired using proper sensors linked with MyXimi application. Firstly, those measures are stored in a local cache, and then uploaded to the network. All the health data that is uploaded to the network needs to be encrypted, first.

The data is encrypted in two steps. First, it is symmetrically encrypted and then, the symmetric key generated is asymmetrically encrypted, so that only a specific user (caregiver) can use the symmetric key to access the health data decrypted. To perform the

asymmetric encryption, it's used the public key of the caregiver. This public key is stored in a repository. Additionally, the health data is also digitally signed with the owner private key. Therefore, once the health asset/measure is encrypted and signed it is ready to be uploaded to the blockchain network. Figure 4.2 represents these steps of inserting a new health measure in Ximi network.

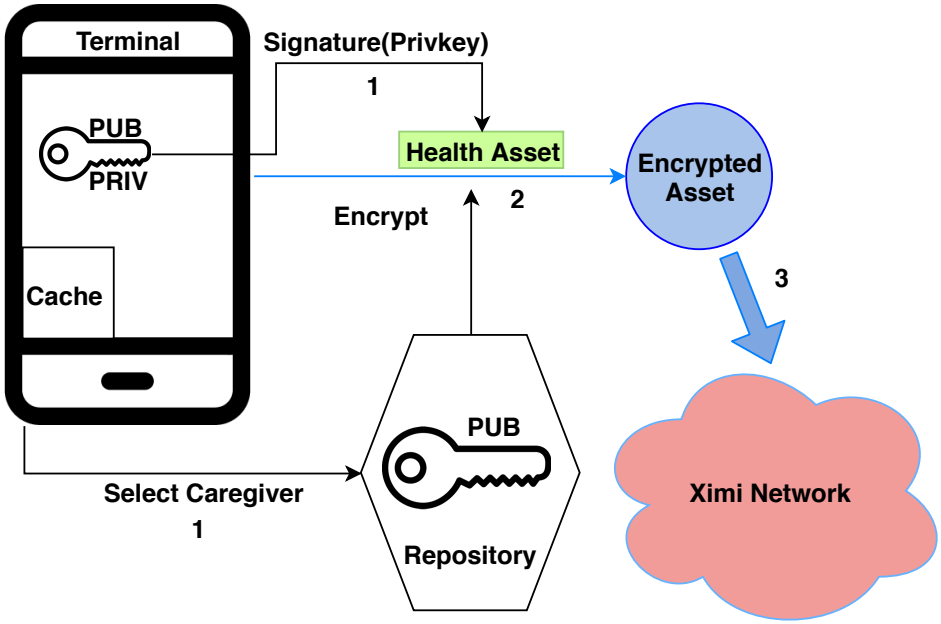

Figure 4.2: **The process of inserting a new health asset in Ximi network. 1 -** The user selects the caregivers to share the data and signs the data with is private key; **2 -** The data is encrypted with the caregiver's public key; **3 -** The encrypted asset is inserted into the blockchain network

Once inserted in the Ximi network the health asset has to different components: a public component and private component. The public component is totally anonymous and is constituted only by hashs and ids. The objective of this component is to mantain a immutable ledger, for every node to see, of all the health assets in the network. The private component is encrypted data, to be, decrypted by its owners and his caregivers. Due to space limits in smartphones, the full state of the blockchain network, that is, all the health assets and its components, cannot be stored in every smartphone terminal. However, a limited amount of health data is maintained on the smartphones (user can choose the data to be deleted and configure the local cache size). That data can be accessed when the device is offline. To access every health asset the user must be online, this way, the network works as an expanded database to health data inserted by the users. The "online" version of the blockchain gives an overview of the entire network, showing the blocks with the public data and the private data that the user has permission to see.

4.3 Accessing a health asset

Currently, if a caregiver wants to access a health measure of his related user, inserted via MyXimi, the user must give the key, by text message, for example, to the caregiver and, then, he uses it to access the data. This process of sharing the key isn't the best neither the most secure. With the introduction of a blockchain network this process becomes more secure and easier at same time. The permissions of access are automatically managed. When a user inserts a certain health asset, he selects the caregivers to share it. By doing that, he asymmetrically encrypts the symmetric key of the asset with the public key of the caregiver. So, when a caregiver selects the user that he wants to access the data, the first step is verifying if the caregiver is in truth a caregiver of the user, for that, every asset has the list of caregivers associated. If the caregiver is in the list, then he can access the asset. To access it, his private key will decrypt the symmetric key of the asset, and that key will decrypt the health asset. The asset is then stored in the local cache of the caregiver. In no time, a decrypted asset is stored in the network. Simultaneously, the digital signature of the asset is verified using the public key of the owner of the asset. Figure 4.3 shows these processes of accessing a health asset stored in Ximi network.

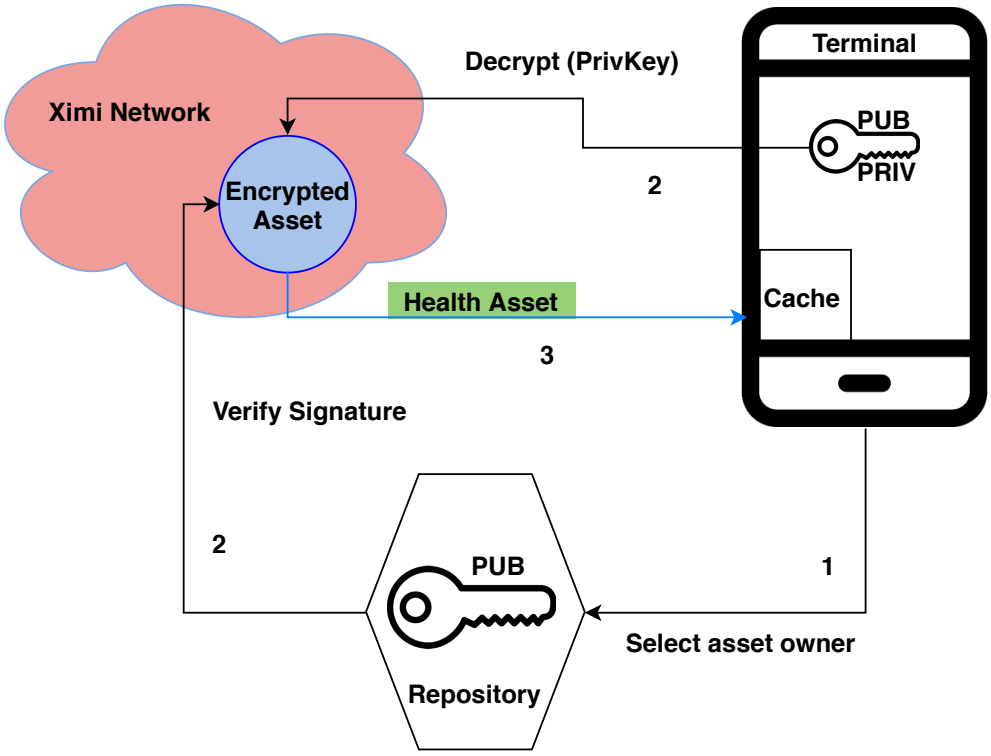

Figure 4.3: **The process of accessing a health asset. 1 -** the caregiver selects the owner of the asset; **2 -** The signature is verified with the public key of the asset owner and the data is decrypted using the caregiver's private key; **3 -** Caregiver gets access to the health asset and stores it

This model presupposes that the caregiver knows the identity of the related user. The caregiver will have to insert some basic personal data of his related user, to get the user public key and therefore have full access to his health data. All the personal information inside the health assets, like the owner info and the caregivers info, is converted to cryptographic hashes, so it can be available to be compared with the inputs. For example, the public part of a health asset has the hash of the owner of the asset. That hash was calculated from a specific predetermined personal data of the owner. When a caregiver wishes to access health data of that user he has to insert that specific personal data in order to create the same hash, so it can be used to search for the assets of that user. That same personal data is used to get the public keys and to search for caregivers.

In the next chapter more technical details will be given about the algorithms used, the composition of blocks and health assets and hashes.

4.4 Security considerations

In any information system there are many ways of happening data breaches. In this proposed system, the elements that can be attack are the MyXimi terminals, the network and its nodes, the repository of keys and the management terminals. The security of the MyXimi terminals (smartphones) can't be assured by the network and app providers, so it will not be discussed here.

There are no infallible networks or systems, so there is always the possibility of a data breaches, however, it can be assured that the data breached doesn't have any value and doesn't compromise the privacy of the users. Therefore, all the health data distributed across all the nodes of the ximi network is encrypted, so it is completely illegible. The keys that decrypt that data are also encrypted. The algorithms (AES and RSA) used to encrypt the data and the keys are proved to be secure, at least, considering the current computational power (in the future, quantum computers will be able to break RSA and AES algorithms very fast). The unencrypted data is only hashes (illegible), indexes, timestamps and numeric ids. Thus, any leaked data from the blockchain network won't compromise the privacy of the user and doesn't represent any value for malicious actions. In the repository of keys, the same happens, the only available data are hashes and public keys, that can't decrypted nothing and can't be related to any specific user. The management terminals only have anonymous health data, so the privacy of the users is assured.

The system providers need to keep track of the Ximi users and be sure to only grant access to the app to well-intentioned users, therefore preventing malicious users to enter the network. In any case, the network is private and controlled by the management nodes, so even if there are malicous terminals in the network, they could never take control of the network.

Chapter 5

Demonstration system

In the present chapter will be described and explained the code developed to simulate the network model presented in chapter 4. First, will be given a general view of the code. Next, its the main functions and features will be explained in detail. From now on, the demonstration system developed will be referred as XBlock.

5.1 General view

The first step in the creation of XBlock was deploying a blockchain environment. In order to overcome possible limitations and the lack of easy customization imposed by complex and well known blockchain platforms, like Ethereum and Fabric (see chapter 2), it was used, as skeleton, a very basic open source blockchain environment. This environment was created by Daniel van Flymen and it is dedicated to cryptocurrencies. The code was developed in python 3 and, released as open source code on github in late 2017 [34]. There were needed several new functions and, modifications in the already existing functions, to transform that environment in a proper blockchain system to store and share health data. The communication system and main structure of the chain remain from Flymen version of blockchain. The blockchain system is based on communication via HTTP requests and uses the micro web framework Flask [35] to deploy the network. The interactions with the blockchain are done using Postman [36], that provides a simple way to use HTTP requests and visualize JSON (JavaScript Object Notation) objects. Thus, XBlock is like a web API. The XBlock code is in two python files: one with the main code and, the other (key file) with the code to generate public and private keys for the users.

The main components of XBlock are:

- **The chain.** Python list where all the blocks are appended.

- **The blocks.** Python dictionaries with relevant information related with the previous block, the creation of the block and the health assets inside it.

- **The health assets.** Python dictionaries with information that characterize the health measures acquired by MyXimi.

- **The nodes/users.** Addresses (URL) of where the code is running (ex. http://127.0.0.1:5002).

The different nodes of the network are simulated running the main code in several ports on the same machine (localhost). Another way of doing this is using different machines. In order to start interacting with XBlock it's necessary to run the main code. Once the main file is called, the key generator file runs, as well, and asks for the user to insert his name and email. After those inputs a pair of keys are created to that specific user, running on that URL. The keys are created in a file .pem, one for each key, those files are stored in the computer running the code. After that, he can start interacting with XBlock, via HTTP requests, using Postman, or another HTTP client. In this initial step it's defined the type of user that is entering the network. If the user inserts is name and email, then, he is a **normal user**, that can insert health data, share it and access data shared with him. On the other hand if the inserts a password then he enters the network as a **master node**, so he can only perform administrative actions. Normal users are the caregivers and users described in chapter 4. The master nodes are the management nodes. In table 5.1 are described the main specifications of each user.

For sake of simplicity, in XBlock there is no distinction between nodes and terminals. In XBlock users are the nodes of the network, that store the full version of the blockchain ledger. As explain in the previous chapter this is not possible when talking about smartphone terminals, however, this demonstration system was developed for computers. Each XBlock normal user has his local and temporary "blockchain", where he puts his data before uploading to the network. When the data is uploaded to the network, the local "blockchains" are replaced by the new, immutable, definitive and common to all nodes, XBlock blockchain. Besides inserting their data in XBlock, normal users can also access health data, shared with them, and consult the historic ledger of the accesses to their health data. The master nodes have the responsibility to upload the data from normal users, creating the "single version of truth" blockchain and broadcast it to all the nodes of XBlock. Additionally, master nodes can delete and add users and, access anonymous data, for stats.

The main file of XBlock is divided in two segments. The first one is the class *Blockchain*, where are all the functions that create the logic and functionalities of XBlock. After that, Flask is initialized and an object of the class blockchain is instantiated, entering, that way, in the second segment of the code. In that segment are all the methods to interact with the blockchain, via HTTP requests. As said before, each node runs the code by itself, and the interactions with the network and, therefore, between nodes are only by HTTP requests.

Table 5.1: **Nodes in XBlock.** A brief description of the nodes specifications in the network.

Nodes	Initial input	Actions	Pub/Priv keys pair
Master Nodes	Password	Register, delete users; upload data from the users; access anonymous data	No
Normal users	Name; email	Load, share and access health measures; access historic ledger	Yes

5.2 Main code

Before entering in a detailed description of each of the core functions of XBlock, it's necessary describe the main variables inside the constructor of the class blockchain, which defines XBlock's initial state. The following list shows a brief description of each main variable:

1. ***chain***: list where are appended the blocks.
2. ***nodes***: a set containing the URL's of all the nodes registered in XBlock.
3. ***pubdata_network*** and ***privdata_network***: lists with the public and private (encrypted) data in XBlock network.
4. ***local_pubdata*** and ***local_privdata***: lists with the local public and private (encrypted) data, before upload to XBlock.
5. ***my_ids***: list with all the *ids* of the health assets inserted by the node.
6. ***my_private***: list of tuples with 3 elements: asset id, symmetric key, token of the asset. This variable is kept in local storage, to allow offline decryption of the health assets of the node.
7. ***private_master***: list of tuples with 2 elements: symmetric key, token. The token and key in this variable refer to non-sensitive data, only for stats.
8. ***views_historic***: list of dictionaries with the *id* of the health asset accessed, hash of the node and the date of access. It provides a historic of accesses of that node to the health assets.
9. ***node_info***: info inputted by the user, comes from the key generator file.
10. ***my_pubk_str***: Public Key *.pem* file converted into a string, also comes from the key generator file. It's useful to facilitate the sharing of public keys in XBlock.
11. ***my_hash***: hash (SHA-256) of that node personal info .

In the initialization of the blockchain is also created the genesis block, by a custom function: *genesis_block*. This block is special because it's the first one, it doesn't contain any health assets and the hash of the previous block (indispensable parameter of a block) is considered to be 1. Additionally, in the XBlock initialization, the public and private keys of the node need to be loaded. In order to "activate" the keys, they are loaded from the .pem file and converted into the python type: rsa.key.Public(Private)Key. The pair of "activate" keys are stored in two variables: *my_pubk* and *my_privk*. The function *load_keys* is responsible for activating the keys.

To facilitate the understanding of the developed coded, table 5.2 gives a brief description of the main functions inside the code and the section where they are explained in more detail.

Table 5.2: **Main functions of the developed code.** A brief description of the main functions and the section where they are explained.

Function	Description	Section
hash	creates a hash of a certain input	5.2.1
load_keys	loads the rsa keys from the .pem file	5.2
create_excel	creates a excel file with health data	5.2.4
generate_id	generates a unique id	5.2.3
resolve_chain	logic of the consensus algorithm	5.2.2
genesis_block	creates the genesis block	5.2
new_block	defines a new block	5.2.1
hash_to_pubk	gives the active public key of a certain user	5.2.3.1
new_asset	create a new health asset to be uploaded into the network	5.2.3
encrypt_key	asymmetrical encryption of the symmetric key	5.2.3.1
encrypt_data	symmetrical encryption of the health data	5.2.3.1
get_my_private	gets the raw private data of the node	5.2.4
full_unlock_private_data	decrypts the private data	5.2.4
global_anonymous_data	returns the anonynous health data in the network	5.2.4
create_views_historic	creates an entry to the historic of views	5.2.5
organize_historic	compiles the historic of views	5.2.5

All the functions in table 5.2 are from the class Blockchain. For each one of them there is, at least, another function that creates an URL that triggers the main function.

5.2.1 Defining a block

The function *new_block* defines and creates a new block, to be appended into the chain. This function can receive, as a parameter: the hash of the owner of the block. This parameter can assume two forms, either it is, directly, *my_hash* or it is the hash of the owner of the health assets in the block. When the block is created by a normal node, in the process of inserting his health assets, it is appended to the local and temporary blockchain and, the owner hash is simply *my_hash*. If the block is created by a master node, in the process of creating the single version of blockchain, the hash appears as a parameter of the function and, it takes the value of the hash of the owner of the health assets inside that block. Thus, this function is triggered by normal users and master nodes. As said before (view section 5.1), a block is a dictionary, with five pairs key/value. Table 5.3 presents and briefly describes all the information inside each block.

Table 5.3: **Content of a block from XBlock.** A brief description of the items inside of a XBlock block.

Name	Type	Description
Block index	integer	Block position in the chain
Owner	string	Hash (SHA-256) of the owner of the health assets inside the block
Assets list	list	All the public information of the health assets
Creation data	float	Unix timestamp
Hash previous block	string	Hash (SHA-256) of content of the previous block

All the hashes are created in a proper function (*hash*), created in the main code. It returns a hash of the type SHA 256, when inserted any hashable item. Blocks are the public information that any user can see in XBlock. The creation of a block is finalized when the block is added to the chain, which is a ordered list with all the blocks.

5.2.2 Generating a chain of blocks

The function *resolve_chain* creates "the single version of truth" of XBlock. This function is only triggered by Master nodes. The algorithm of this function consists in three steps:

- Upload all the data, public and private, from the nodes to XBlock;
- Create a unique chain with those uploaded health assets;
- Broadcasting the chain to all the nodes.

This function requires interaction between nodes. In the first step, the master node requires all the lists with public and private (encrypted) data from all the other nodes. To accomplish that, a GET-Request is sent to get the local data of each node in the network, inside the lists: *local_pubdata* and *local_privdata*. This process gets the data in association with its owner node hash. Once the data is uploaded, it needs to be created a blockchain

to insert the data. There are three different situations that define how the new blockchain is created:

- First time ever, there is no local data or data in the network;
- There is local data to be uploaded to XBlock;
- There is no new data to be uploaded.

If there is no data in XBlock, that is, it's the first time ever that health assets are uploaded to XBlock, then, a new blockchain is created. The functions: *new_ block* and *genesis_ block* are triggered and, they create a totally new blockchain with all the uploaded health assets. Since these assets were obtained in association with their owner node, a new block is created every time the owner changes. This way, each block only has health data from one normal user, which facilitates the visualization of data. On the other hand, if there is already uploaded data then, the existing blockchain remains, and new blocks, with the new health data, are appended to the existing chain. Finally, if there is no local data in any of the nodes, and the function resolve chain is triggered then, the existing chain remains and there is no need to change or broadcast it.

To broadcast the new chain, a POST-request is sent to all nodes. This request updates all the chains, all the public data and all the private data of all nodes. This process of update empties the variables *local_ pubdata* and *local_ privdata*, that are waiting for the user to insert new data to upload. Simultaneously, the variables *pubdata_ network* and *privdata_ network* are filled with the public and private encrypted data from all nodes. Once these steps are done there is only one single version of blockchain in XBlock.

5.2.3 Creating a health asset

With the function *new_ asset*, the health assets are created and locally stored. A new health asset has two parts: public data and private data. The first one is visible in the blocks to any user of XBlock. The second is only available to specific users. The creation of a new health asset begins with the owner input. The items inserted by the user are specified in table 5.4, along with a brief description of each. When these items are inserted by the user a POST request is generated to register the new asset in the node. After the upload of the data, made by the master node, this asset is distributed across the XBlock network.

Table 5.4: **User input inserting a new health asset.** Type and description of all the items inputted by the user to specify his new health asset.

Input	Type	Description
Measure type	integer	ID with a physiological match
Measure value	integer	Value in predetermined units
Acquisition date	string	Normal date
Caregivers list	list	Name; email of the caregivers

A health asset is a health measure acquired with MyXimi application. The measure is configured by his type, value and date. Currently the application supports six different types of health data. In table 5.5 are enumerated those types of data, as well as the predetermined units and the ID's. Another important parameter to characterize a health asset are the caregivers. The user inserts the personal data (for example, Name and email) of the users, that he gives permissions access that health asset. Those users are his caregivers. If the user doesn't want to share the asset with anyone he just has to let that field empty. When this happen, the data is inserted in the network however only the owner can access it.

Table 5.5: **Correspondence between the ID and the physiological term.** Predetermined unit of each physiological parameter is shown, along with its name and matching ID.

ID	Unit	Physiological term
1	mg/dL	Blood glucose
2	bpm	Heart rate
3	^{o}C	Temperature
4	mmHg	Blood pressure
5	O_2%	Oxygen saturation
6	Kg	Weight

The public data of a health asset is described in table 5.6. The hash owner is the variable *my_hash*. The hash caregiver is calculated using the function *hash()* with the personal data inserted by the user, as a parameter . If there is only one caregiver then, hash caregiver is only one hash. If there is more than one caregiver, the hash caregiver becomes a list of hashes. The Asset ID has the following form: *XXXX_00* and it is created by a custom function: *generate_id*. The X's are the first four elements of the hash of the owner of the health asset (*my_hash*). The 00's are the number of asset, that is incremented in each creation. Thus, the created ID is unique and easy to distinguish.

Table 5.6: **Public information of a health asset.** Type and description of all the items that defined the public information that will be inserted in blocks.

Name	Type	Description
Hash owner	string	Hash of the personal information of the asset owner
Hash caregiver	string	Hash(es) of the personal information of the caregiver(s)
Asset ID	string	Unique ID of the health asset
Date	string	Acquisition date inputted by the user

The private data is described in table 5.7. The asset ID and the hash caregiver are repeated because they are useful to search assets faster and easier. The most sensitive data of the private data is the owner personal identity and info, that needs to remain private, thus only the caregivers, chose by the user can have access to that. It's also a supposition of XBlock that the Caregivers already have the personal information of the

user needed to create the hash owner. In the function: *new_asset* are called two other functions responsible for encrypting the private data.

Table 5.7: **Private information of a health asset.** Type and description of all the items that defined the private information that will be encrypted.

Name	Type	Description
Owner	string	Personal information of the asset owner
Hash Caregiver	string	Hash(es) of the personal information of the caregiver(s)
Asset ID	string	Unique ID of the health asset
Date	string	Acquisition date inputted by the user
Type	integer	ID with a physiological match
Value	integer	Value of the health measure in predetermined units

5.2.3.1 Health data encryption

To improve the security of the health data, which is the major objective of XBlock and this thesis project, the data is *double* encrypted. First, the data is symmetrically encrypted generating a symmetric key. Then, this symmetric key is asymmetrically encrypted. Figure 5.1 represents a flowchart with a summary of the process of double encryption.

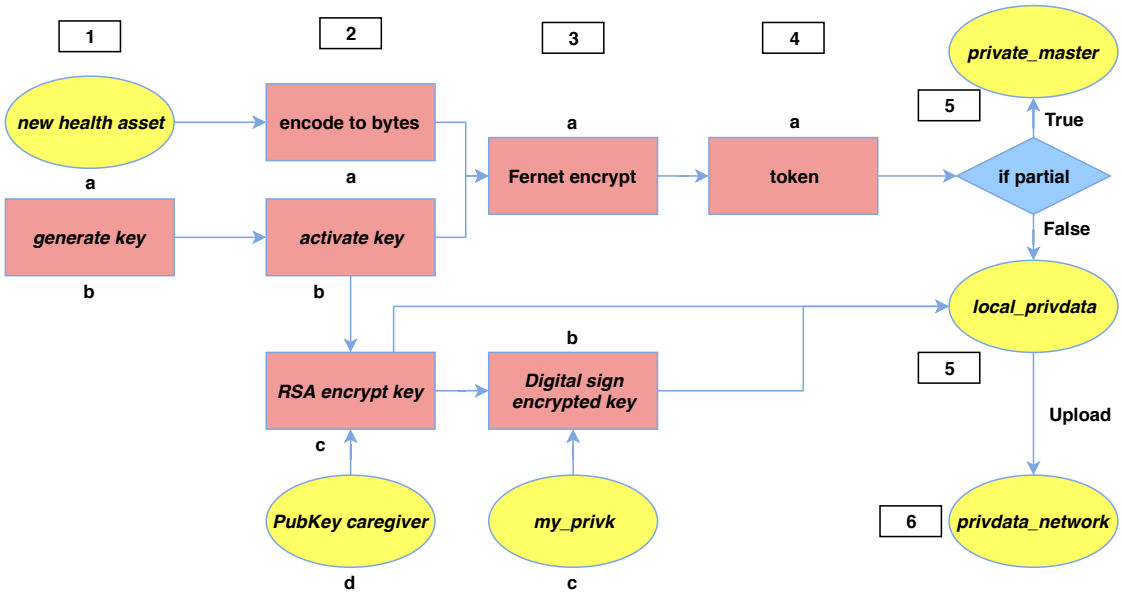

Figure 5.1: **Flowchart of the process of data encryption.** The numbers represent different time moments. The letters represent parallel or near parallel processes. For each health asset are generated two different keys (1.b.). One for the complete data and the other for the anonymous data. For the sake of simplicity, it weren't represented two different branches for each key. However, in moment 5 it's shown the different destinations of each type of data, associated with its specific key. The anonymous data isn't asymmetrically encrypted.

Once the dictionaries with the public and private data of a health asset are created (Fig. 5.1- 1.a), it's time to encrypt the private data and make it available to specific users

only. The public data associated with that health asset is uploaded, so the network can keep a total ledger of the health measures created by the users, even if they aren't shared with any other user. The two functions responsible for encrypting data are *encrypt_data* and *encrypt_key*. Every time *encrypt_key* is triggered the other function is automatically triggered also, but the contrary doesn't happen. Thus, when there isn't a caregiver only *encrypt_data* is called.

The function *encrypt_data* receives as a parameter the private data directly from the *new_asset* function if there is no caregiver to share the data. Inside this function are generated two different python tuples of encrypted data. One contains the anonymous private data and the other all the private data. In the first one, the personal information of the owner is replaced by his hash. So, the result is a tuple with the symmetric key and the token of the anonymous data. The second generated tuple has three elements: the asset ID, the symmetric key and a token for all the data. The tuple with the partial data encrypted is appended to the *private_master* list (Fig. 5.1 - 5.a). On the other hand, the tuple with the total data encrypted is appended to the *my_private* list. For each health measure encrypted there is a new symmetric key and token. To symmetrically encrypt the data was used the python module: cryptography.fernet. The encryption process consists in the following steps:

1. Create a symmetric key: Fernet.generate_key() - fig. 5.1: **1.b**.
2. Activate the generated key: Fernet(key) - fig. 5.1: **2.b**.
3. Prepare the private data to encryption, convert dictionary to bytes - fig. 5.1: **2.a**.
4. Encrypt data: Fernet(key).encrypt(data), generates the token - fig. 5.1: **3.a -> 4.a**.

Each encryption, of the anonymous data and total data, goes by these steps. Fernet module uses AES 128 bits algorithm. The generated key is the base64url encoding of two fields: a signing key (128 bits) and a encryption key (128 bits), resulting a key of 256 bits. The token created is the base64url encoding of five fields [37]:

- version - version of the format used by the token (8 bits);
- timestamp - seconds since January 1 1970 (64 bits);
- IV - initialization vector used in AES encryption and decryption (128 bits);
- Ciphertext - original inputted message encrypted (multiple of 128 bits, AES block size);
- HMAC - SHA256 hash of the three previous items (256 bits).

Besides appending tuples of partial and total data to their specific list, this function returns the key and token of the total data that is used in the function *encrypt_key*.

The function *encrypt_key* has the objective of direct the private encrypted data to a specific user, using asymmetric encryption. If there is a caregiver defined by the user then, this function is triggered in the *new_asset* function. This function requires as parameters the private data and list of tuples: hash, public key. As said before, when initializing XBlock each normal user gets a pair of RSA keys stored in a *.pem file* in his computer, that are activated after (my_pubk and my_privk). These keys are 1024 bits long and they are created using RSA python module. Every user has to have access to all the other user's public keys. For that to happen they need to be distributed in the network. To get the public keys of the other nodes a GET-request was created, that returns the variable *my_pub_str* of each node and the node hash to identify the owner of the key. From this string it's possible to create an active RSA public key. The string of the key is converted to *.pem* file and then it's loaded into a proper RSA key. So, in the first time that is necessary to share an asset with a user this process of getting his public key has to be done. After that, the public key file of that node remains stored in the user's computer. The function *hash_to_pubk* sends the request and compiles the information in a list (for more than one caregiver) of tuples (hash, public key). Thus, when *encrypt_key* is triggered it calls *encrypt_data* to symmetrically encrypt the private data and then uses the token and symmetric key generated by that function. This function generates a tuple consisting of the private data token and a list with of the symmetric keys encrypted with the public keys of specific users. Each item of this list is dictionary that contains:

- a digital signature: the user signs the key of his health asset with his private key - rsa.sign(key, *my_privk*, "SHA-256") - flowchart: **3.b**;
- the symmetric key encrypted: the key is encrypted with the public key of the caregiver - rsa.encrypt(key, pubkey of the caregiver) - flowchart: **2.c**;
- the hash of the caregiver;
- the hash of the owner.

The number of elements of this list is equal to the number of caregivers defined by the user, when inserting his health asset. The final tuple generated by this function will be added to the list *local_privdata* (Fig. 5.1 - 5.b). After the upload of this data only the caregiver selected will be able to decrypt the key and have total access to the health asset. Additionally, it's important to have a symmetric key encrypted with the public key of its owner. Thus, if the user loses access to his local data or if he needs to delete it, he can access the network to get it again. In this case, the user will behave like a caregiver of himself. The health data symmetric key can never be in the network without being asymmetrically encrypted.

5.2.4 Access private health data

The permission of access to private data can be given in three different situations, that trigger different functions in the main code:

1. A normal user wants to access his local private data
   - *get_my_private()*.

2. A normal user wants to access the private data shared with him
   - *full_unlock_private_data()*.

3. A master node wants to access anonymous private data for stats
   - *global_anonymous_data()*.

In the first case, it's only necessary to decrypt the private data that is locally stored in the list *my_private*, that receives data directly from the function *encrypt_data*. This variable can never be shared. Its only objective is to grant access, locally, of the health data to their owners. As previously mentioned, each element of that list is a tuple (id, key, token). Thus, to access the data the function uses the key to decrypt the asset that is only symmetrically encrypted with a fernet key. The method used is: *Fernet(key).decrypt(token)*, which returns the private data coded. After decoding the data, the full private data is obtained. The user can choose if he wants to see all his data (requires only a GET-request) or some specific assets (POST-request), inserting the ids (asset id).

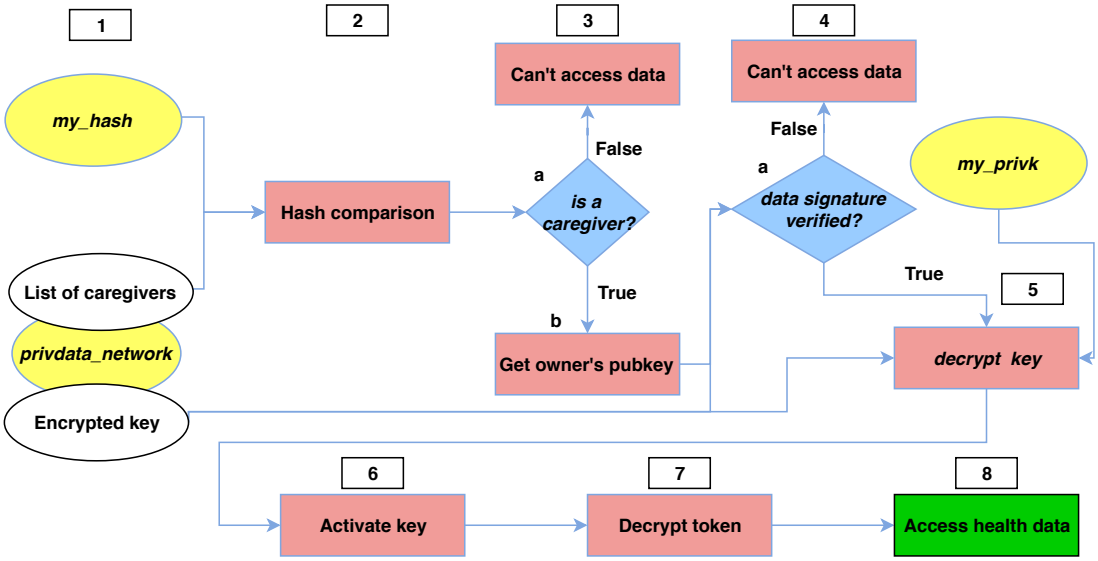

Figure 5.2: **Flowchart of the process of data decryption.** The numbers represent different time moments. The letters represent parallel or near parallel processes.

The second case is described in the figure 5.2 by a flowchart. It consists of a double decryption of the private data already uploaded into XBLock. The private data to decrypt is inside *privdata_network* (fig. 5.2 - 1). The variable *privdata_network* contains encrypted

data from all the users and other important elements: signature, key, caregiver hash and owner hash. When a normal user requests the access of all the health data shared with him the following steps take place:

1. Confirmation that the user belongs to the list of caregivers in the health assets. This is done by hash comparison between hash of the node asking for the data and the hashes in the list of caregivers - fig. 5.2: **1**,**2**.

2. Get the public key of the health asset owner - flowchart: **3.b**.

3. Using the public keys to confirm the origin of the data. This is done verifying the digital signature - *rsa.verify(encrypted key, signature, public key)*, returns True if it is successfully verified - fig. 5.2: **4.a**.

4. Decrypting the symmetric key of the health data. It is used the method: *rsa.decrypt(encrypted key, my_privk)* - fig. 5.2: **5.a**.

5. Decrypting the health asset using the decrypted key. To accomplish this step it is used the same method used to decrypt the local assets - fig. 5.2: **6**, **7**, **8**.

If any of these steps fails, the access to the health asset is denied. The user can choose if he just wants to see health data from a specific user or from all users that he can. To specify the user, he inserts the personal data required to create the owner hash, a POST-request is generated, and the assets are searched by hash comparison between the input and the hash owner. In the other option, all the assets that have the caregiver (user that is asking to access the data) hash in its caregiver list are returned.

In the third case, a symmetric decryption is all that is needed to access the data. To get the anonymous private data of all the XBlock users, the master nodes need to request the variable *private_master* from all nodes and proceed like in the first case.

As an additional feature of XBlock all these functions create an excel file with the data accessed. When all the data asked is collected it is placed inside a data frame. This data frame is, then, converted in an excel file. The objective of the creation of this file is to improve the data visualization and posterior analysis.

5.2.5 Historic of views

The last main feature of XBlock is creation of a historic of views. This historic compile the accesses to health assets in the variable *views_historic*. The function responsible for creating this variable, called *create_views_historic()*, is triggered every time a health asset is accessed and only requires the id of the accessed asset. Each this function is triggered it creates a entry to the historic, that contains the asset id, the hash of user that accessed the data and the date of access. This way the views are individually appended to the list

views_historic and each normal user has his own list of views. When a user wants to have access to historic of views to his health data, the lists *views_historic* of all users are merged and the necessary health assets are selected. The function that prepares the historic for visualization is called *organize_historic()*. The selection of the assets is again made by hash comparison. The user must specify the identity of the caregiver, whose historic he wants to see, that way hash comparison can be made. If he wants to see the historic of more than one caregiver, he just has to insert all their identities. In association with this function is a POST-request.

Chapter 6

Simulation

The objective of this chapter is to demonstrate the features of XBlock, described in the previous chapter 5, and testing its the security and functionality. Most of the presented images are screenshots of Postman, the program used to interact with XBlock.

To show XBlock working and to test all of its methods, previously described, it was simulated a network with four virtual users. Each user is a terminal window running XBlock the main code, each in a different port, as figure 6.1 shows. In the port 5000 is the master node, that only inserts the password (that is very basic in the example). In the other ports (5001, 5002, 5003) are the normal users, each one inputs his personal data, in the example, name and email.

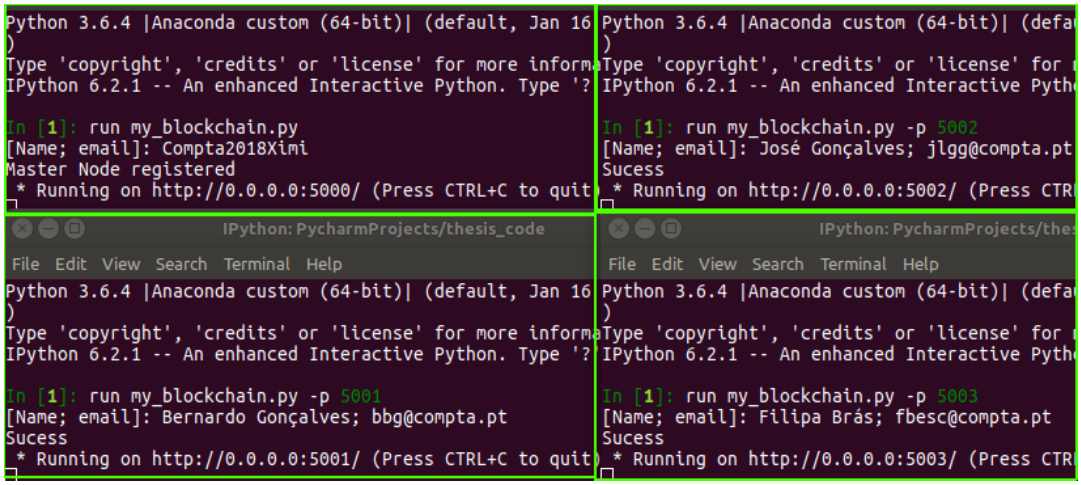

Figure 6.1: **Terminal windows simulating 4 users.** Each window has a user created, with a name and email or password (master node)

After this first interaction, XBlock is deployed, ready to receive health assets and to work with them. After this point all the interaction is made through a HTTP client

(like, Postman). Figure (6.2) shows the environment used to interact with XBlock and a representation of the user input to create a health asset. As said in chapter 5 the user inserts the physiological type of the measure, the value of the measure, the caregivers to share the asset and the date of acquisition. Each interaction with XBlock is made by an HTTP request, so each main function has one or more requests associated. In the presented environment the requests are organized by user and function.

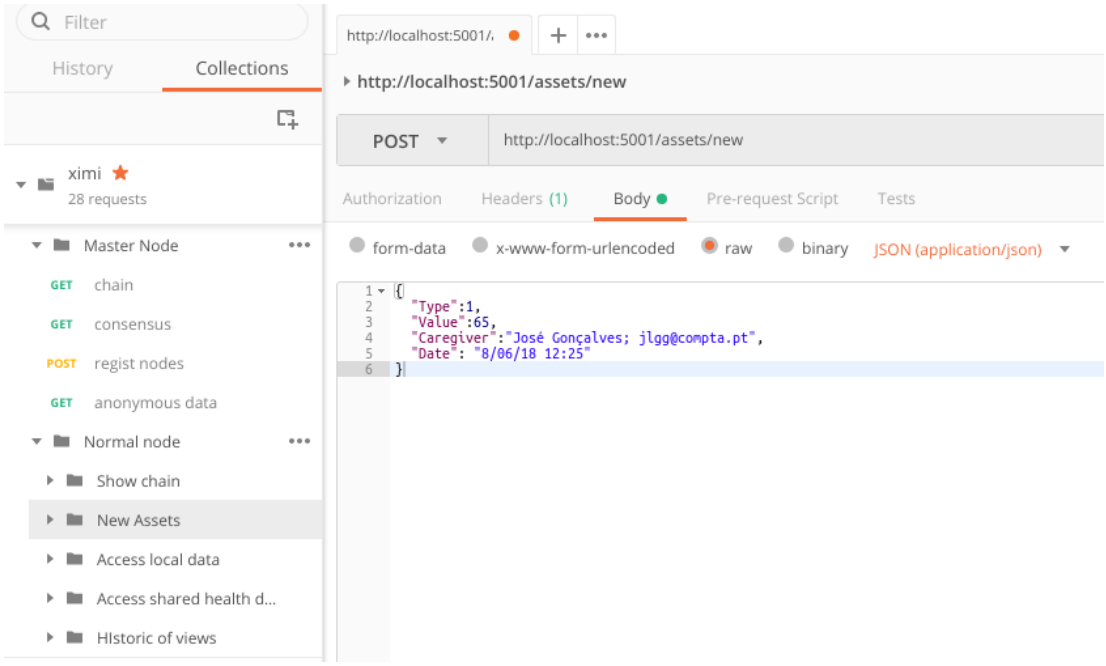

Figure 6.2: **Postman environment and health asset input.** At the left side of the figure is a collection of requests divided in two groups. In the first group are the requests associated with master node's functions. The other group is related with the normal nodes main functions: see chain, create new assets, access local data, access shared data, access historic of views.

It was created a thread that executes several HTTP requests to speed up the simulation process. In this thread, using cURL language, the master node registers all the nodes of the network, and each of them inserts their health assets. Table 6.1 shows the distribution of assets per user. Each asset was created with inputs like the one showed in figure 6.2, except for the date acquisition. It was used the date of the introduction of the assets in the system as the acquisition date. Thus, this thread creates the base scenario of the simulation and tests.

Table 6.1: **Assets per user.** Number of assets of each user and which of them are shared.

Port	user	1st 4 - Hash	Nr Assets	Shared Assets
5001	bbg	44a0	12	8
5002	jlgg	c290	8	8
5003	fbesc	bfe6	16	14

At this stage there are three normal users, each one with their personal health assets

stored only in their local storage. So, to create a blockchain distributed network, all the health data, from all the users, must be uploaded into the network and securely stored in it and distributed to all users. The master node is the one that have the responsibility of ordering that upload and, with that, creating the single version of blockchain, that all the users can equally see. The uploaded data was two components:

- **Public data**: registered in the immutable blocks that every user can see;
- **Private data**: encrypted data, hidden from unauthorized users.

In the figure 6.3 is shown the first block of the user *bbg*. In the block there is the list of all the health assets inserted by that user, for each asset is shown only the public data. That is, the ID, the hashes of the caregivers and the acquisition date.


```
    {
        "Asset ID": "44a0_9",
        "Date": "2018-06-14 14:26:05",
        "Hash caregiver": [
            "c290b53af21fc790c753ddaf2d69e0b859a187015112d43865416c45769ce5ba",
            "bfe6187ab441dbfbfd61d9d1b922ac0c724c117b03231d1537b6ee6121679c80"
        ]
    }
    {
        "Asset ID": "44a0_10",
        "Date": "2018-06-14 14:26:05",
        "Hash caregiver": "c290b53af21fc790c753ddaf2d69e0b859a187015112d43865416c45769ce5ba"
    }
    {
        "Asset ID": "44a0_11",
        "Date": "2018-06-14 14:26:05",
        "Hash caregiver": "This Asset isn't shared"
    },
    {
        "Asset ID": "44a0_12",
        "Date": "2018-06-14 14:26:05",
        "Hash caregiver": "bfe6187ab441dbfbfd61d9d1b922ac0c724c117b03231d1537b6ee6121679c80"
    }
],
"Block index": 2,
"Creation Date": 1528982770.0248914,
"Hash previous block": "cac0bee301c5b23c9ebac14d1b7dda4532301b9cec1636c7cdfd213d26b615df",
"Owner": "44a001203101a83945f5e2b2aa8389b12ee8cf74e803e32606237034ea4c930d"
```


Figure 6.3: **A block in XBlock.** It can be seen all the components from a block, described in chapter 5: index, owner, date, hash and assets list. Due to space limits the assets list is not totally shown. The block owner is *bbg*.

Looking at the table 6.1 and at the figure 6.3 it can be seen that the owner of the block is the user identified by the username: *bbg*. Additionally, the asset *44a0_9* is shared with two users; *44a0_10* is shared with *jlgg*; *44a0_11* isn't shared with anyone and *44a0_12* is shared with *fbesc*. Figure 6.4 shows the private data from a health asset. As can be seen this data is illegible because it is encrypted. The method used to show this encrypted data is not available to users, because it has no use for them.

After the creation and upload of the health assets from each user it is possible to query the network for them. Thus, the authorized user can have full access to the private health data. In the figure 6.5 is shown the private data from a health asset as an authorized user sees it. The figure is a response of the method to access all the health assets from a specific user, in this case, to relate with the figure 6.3, the user that query the network for the

```
"6741414141414142624b5273706e78554f3541465a33775264734a624b67514e553634487857663771383752453 26b6d303872506354524f75
    6458503671555762537378 6a6279534748636d6a393133617046724d4b3862585577543747724c71596a7149535a4849396a454650516
    2755475384936413559524d64317964527a57645a785a576d313932566f3255412d3462455344354575414162372d484973457068664a
    4432397775353764504142686748647 76d335a78344c345943412d715565304c2d726b3746733743375743677169524b5731656230576
    b70576c4f664179476b504b69786842774f7a555966495f36776558584f634768466d4c6a34436e5a335764706f342d5f737538553757
    7641554538684261393546 4b",
[
    {
        "Encrypted key": "47cda088e95a1c450dae3c1cbd6c7fe2aa70abd699f606b5802a5f2eb849dc3d094814ed9f49c533dd282e5
        "Hash Caregiver": "44a001203101a83945f5e2b2aa8389b12ee8cf74e803e32606237034ea4c930d",
        "Hash owner": "c290b53af21fc790c753ddaf2d69e0b859a187015112d43865416c45769ce5ba",
        "Signature": "20fdde3b7ae95859a5fdfba692182ec96eae7913f1a8e5ec71508207154585477 1b2b2a47d998a2ada0fdfb4b2c
    }
```


Figure 6.4: **A sample of the encrypted data.** The encrypted data is distributed for all the network and it is illegible.


```
"This is the health data shared with you by Bernardo Gonçalves",
[
    {
        "Asset ID": "44a0_2",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Blood Glucose",
        "Value": 100
    },
    {
        "Asset ID": "44a0_4",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Weight",
        "Value": 98
    },
    {
        "Asset ID": "44a0_6",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Blood pressure",
        "Value": 160090
    },
    {
        "Asset ID": "44a0_8",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Blood Glucose",
        "Value": 150
    },
    {
        "Asset ID": "44a0_9",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Temperature",
        "Value": 38
    },
    {
        "Asset ID": "44a0_10",
        "Date": "2018-06-19 15:07:46",
        "Owner": {
            "Email": "bbg@compta.pt",
            "Name": "Bernardo Gonçalves"
        },
        "Type": "Weight",
        "Value": 95
    }
```


Figure 6.5: **Decrypted private data accessed from *jlgg*.** It can be seen the raw data with the right physiological term in the type, and all the information inputted by the user, when creating the asset.

assets was *jlgg* and he asked to access the assets of *bbg*. To do this, the user has to insert the personal information of the user that he wants to see the health data, name and email. If he fails to do that, he cannot access the health data no matter what.

It can be seen in figure 6.5 the assets *44a0_9* and *44a0_10*, that were presented in figure 6.3. These assets were referenced as shared with *jlgg* in their public information, and, for that reason, their related private information appear in this figure. After the accesses of health assets a historic is created, as explained in chapter 5. An example of a historic of views is showed in figure 6.6.

Figure 6.6 represents the historic of views made by *jlgg* to the assets from *bbg*. The same assets presented in figure 6.3 and 6.5 are shown (*44a0_9* and *44a0_10*). The historic suggests that *jlgg* accessed those health assets two times in the same day.

If a user queries the network in order to access his local assets, the output is equal to the one presented in figure 6.5, although the assets, that aren't shared with anyone, also

```
{
    "Asset ID": "44a0_9",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:30:58"
},
{
    "Asset ID": "44a0_10",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:30:58"
},
{
    "Asset ID": "44a0_2",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
},
{
    "Asset ID": "44a0_4",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
},
{
    "Asset ID": "44a0_6",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
},
{
    "Asset ID": "44a0_8",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
},
{
    "Asset ID": "44a0_9",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
},
{
    "Asset ID": "44a0_10",
    "Caregiver": "José Gonçalves; jlgg@compta.pt",
    "Date": "2018-06-20 09:42:13"
}
```


Figure 6.6: **Historic of views from *bbg* of *jlgg* assets.** The id of the health asset, the caregiver that saw the asset and the date of the access are displayed, for each asset.

appear.

At last, if the master node wants to get all the health data in XBlock, without compromising the identity of its owner, he has to execute a proper method, that only he has access. The output of that method is presented in figure 6.7.

Obviously, in figure 6.7 there is only presented a part of the health data accessed by the master node. Once again, the assets *44a0_9* and *44a0_10* are presented, however, the information about the owner and caregiver is not shown, as intended.

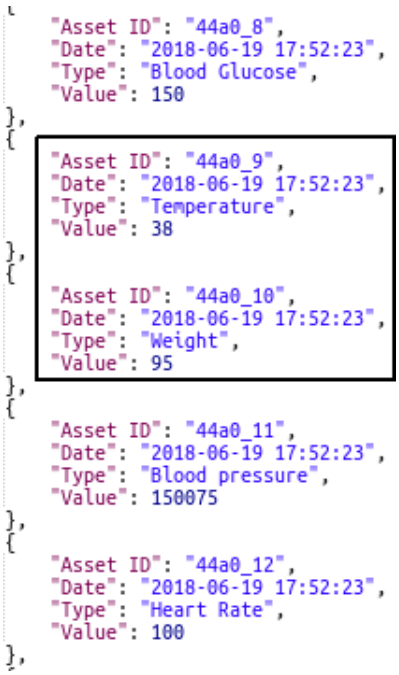

Figure 6.7: **Decrypted anonymous private data accessed from the master node.** All the data from the health asset can be seen, except for the personal information of its owner.

Chapter 7

Conclusions

In an even increasingly cyberworld, data security and user's privacy issues have a greater dimension and importance. The days are gone when the blockchain technology was solely related with money transfers without a bank. Nowadays, this technology appears as a major tool to provide an improvement in terms of security and privacy to the current cybersystems. However, due to the blockchain emergent character, there are several challenges to overcome to create strong and reliable solutions based on blockchain for applications outside the financial world.

In this thesis project the focus was on the application of blockchain in healthcare. All the forms of the blockchain technology were well studied and perceived. After this study it was concluded that is possible to create a system, based on the blockchain technology, to store and share sensitive health data with more security and privacy than the current used systems. Additionally, it was concluded that there were some features of the technology that simply weren't necessary or didn't added value to the solution created in this thesis project. For that reason, the developed model was a hybrid blockchain model. The model uses the blockchain immutability, cryptographic and distribution features and, ignores the traditional mining process and the public nature of blockchain. To demonstrate and test the main concepts and features of the proposed model it was developed a demonstration system, XBlock. That system was presented and, tested in real time, to a board of 28 specialists in areas like cybersecurity, information technology and business intelligence, in a Compta event. The board stated that XBlock was really a true value for improving the security and privacy of health data. Additionally, during the event, was highlighted the importance of the blockchain technology in asset tracking, which took to the creation of a new function in XBlock - historic of views, which was already explained in chapter 5. XBlock shown the encryption methods that secure the information and control the access of its owner or caregiver. Also, methods to anonymize the health data were used, so that

management nodes of the network could access anonymous data. Therefore, during this thesis work the three starting questions were answered: a hybrid blockchain system was created to securely and privately store, share and access anonymous sensitive health data.

XBlock and most of the technologies based in blockchain have a typical problem of low performance in data search. In the developed system to get a specific health asset, it's necessary to walkthrough all the chain, which can take some time with larger chains. For that reason, the code as many loops and, they can easily lower the performance of the system. With chains like the one used in the simulation (chapter 6) there is no such problem. Also, XBlock was developed only for very simple health assets, constituted only by a value and the correspondent physiological greatness. In many applications the health assets can be much more complex.

The limitations above described can lead to future work. In the future XBlock can be optimized, improving its performance and scalability. A smarter indexation of the network can be made using the fact that each block only has health data of one user. By doing that the performance of the network would improve. Additionally, bulk encryption, that is, inserting several encrypted health assets in the network, at same time, can be implemented after the realization of robust security tests to all components the system (blockchain network + health app). Also, the system can be changed in order to accepted more complex health assets.

The blockchain technology will be, for sure, a major technology in the next years with many important applications in most know scientific and business fields. This thesis projects proves that it can also be a true value in healthcare applications. This work, being developed thinking in a small health application, is a small step towards a greater objective: creating an integrated global blockchain network to store health data. Within this network the data could be available to its owner and caregivers in any health entity.

Bibliography

[1] T. Przybycien. "Department of Biomedical Engineering." In: *Biomedical Engineering* (). URL: https://www.imperial.ac.uk/pls/portallive/docs/1/51182.PDF.

[2] Compta. *MyXimi - Fighting Loneliness through Gamification.* 2016. URL: http://www.myximi.com/ (visited on 01/11/2018).

[3] S. Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System." In: *Www.Bitcoin.Org* (2008), p. 9. ISSN: 09254560. DOI: 10.1007/s10838-008-9062-0. arXiv: 43543534534v343453. URL: https://bitcoin.org/bitcoin.pdf.

[4] C. McLain. *A Brief History of Blockchain: An Investor's Perspective.* 2017. URL: https://medium.com/the-mission/a-brief-history-of-blockchain-an-investors-perspective-e9b6605aad68 (visited on 01/09/2018).

[5] T. T. Kuo, H. E. Kim, and L. Ohno-Machado. "Blockchain distributed ledger technologies for biomedical and health care applications." In: *Journal of the American Medical Informatics Association* 24.6 (2017), pp. 1211–1220. ISSN: 1527974X. DOI: 10.1093/jamia/ocx068.

[6] A. M. Antonopoulos. *Mastering bitcoin : unlocking digital cryptocurrencies.* 2014, p. 272. ISBN: 9781449374044.

[7] Observatory of Public Sector Innovation. "Blockchains Unchained: The Implications of Blockchain Technologies for the Public Sector." In: (2017), pp. 1–51.

[8] R. Krawiec, D. Barr, J. Killmeyer, M. Filipova, F. Quarre, A. Nesbitt, K. Fedosova, L. Tsai, and A. Israel. "Blockchain : Opportunities for Health Care." In: *NIST Workshop on Blockchain & Healthcare* August (2016), pp. 1–12.

[9] E. community. *What is Ethereum? — Ethereum Homestead 0.1 documentation.* 2016. URL: http://ethdocs.org/en/latest/introduction/what-is-ethereum.html (visited on 01/10/2018).

[10] The Linux Foundation. *Hyperledger-fabricdocs master documentation.* URL: http://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html (visited on 05/13/2018).

[11] The Linux Foundation. *About – Hyperledger.* 2017. URL: https://hyperledger.org/about (visited on 01/11/2018).

[12] European Commission. *Digital single market - How can Europe benefir from Blockchain technologies?* Tech. rep. 2017, p. 2.

[13] I-Scoop. *Blockchain - distributed ledger technology for digital trust in business.* 2018. URL: https://www.i-scoop.eu/blockchain-distributed-ledger-technology/ (visited on 05/04/2018).

[14] IBM Institute for Business Value. "Healthcare rallies for blockchains." In: (2016). URL: https://ibm.co/2N08SqZ.

[15] K. Abouelmehdi, A. Beni-Hessane, and H. Khaloufi. "Big healthcare data: preserving security and privacy." In: *Journal of Big Data* 5.1 (2018), pp. 1–18. ISSN: 21961115. DOI: 10.1186/s40537-017-0110-7. URL: https://doi.org/10.1186/s40537-017-0110-7.

[16] Calyptic. *10 Biggest Problems in Healthcare Cybersecurity.* 2017. URL: https://www.calyptix.com/hipaa/10-biggest-problems-in-healthcare-cybersecurity/ (visited on 05/29/2018).

[17] European Commission. *Transformation of Health and Care | ehealth | Digital Single Market.* 2014. URL: https://ec.europa.eu/digital-single-market/en/european-policy-ehealth (visited on 09/10/2018).

[18] European Commission. *A new era for data protection in the EU.* Tech. rep. 2018.

[19] N. Lord. *What Is Data Encryption?* 2015. URL: https://digitalguardian.com/blog/what-data-encryption (visited on 05/30/2018).

[20] J. Mason. *What is Advanced Encryption Standard (AES).* 2017. URL: https://thebestvpn.com/advanced-encryption-standard-aes (visited on 05/30/2018).

[21] Martignon. *Rijndael Algorithm (Advanced Encryption Standard) AES.* 2012. URL: https://bit.ly/2x0MoeD.

[22] TechTarget. *What is RSA algorithm (Rivest-Shamir-Adleman)? - Definition from WhatIs.com.* 2014. URL: https://searchsecurity.techtarget.com/definition/RSA (visited on 05/31/2018).

[23] Brilliant.org. *RSA Encryption.* 2018. URL: https://brilliant.org/wiki/rsa-encryption/ (visited on 05/31/2018).

[24] H. Bodur and R. Kara. "Secure SMS Encryption Using RSA Encryption Algorithm on Android Message Application." In: (2015).

[25] B. S. Kaliski, Jr. "{RSA} Digital Signatures." In: *26* 5 (2001), pp. 30,32–33,35–36. ISSN: 1044-789X. URL: http://www.drdobbs.com/rsa-digital-signatures/184404605.

[26] W. commons. *File:Digital Signature diagram.svg.* URL: https://bit.ly/2wRlqGO (visited on 06/21/2018).

[27] TechTarget. *What is DDBMS (distributed database management system)? - Definition from WhatIs.com*. 2005. URL: http://searchsqlserver.techtarget.com/definition/DDBMS (visited on 01/15/2018).

[28] L. A. Linn and M. B. Koo. "Blockchain For Health Data and Its Potential Use in Health IT and Health Care Related Research." In: *U.S. Department of Health and Human Services* (2016), pp. 1–10. URL: https://www.healthit.gov/sites/default/files/11-74-ablockchainforhealthcare.pdf.

[29] T. Mcconaghy, R. Marques, A. Müller, D. De Jonghe, T. Mcconaghy, G. Mcmullen, R. Henderson, S. Bellemare, and A. Granzotto. "BigchainDB: A Scalable Blockchain Database (DRAFT)." In: *BigchainDB* (2016), pp. 1–65. URL: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf.

[30] IBM Global Business Services Public Sector Team. "Blockchain: The Chain of Trust and its Potential to Transform Healthcare – Our Point of View." In: *NIST Workshop on Blockchain & Healthcare* (2016). URL: https://bit.ly/2oBJDLw.

[31] M. Rouse. *What is data lake?* 2015. URL: http://searchaws.techtarget.com/definition/data-lake (visited on 01/19/2018).

[32] A. Ekblaw, A. Azaria, J. D. Halamka, A. Lippman, I. Original, and T. Vieira. "A Case Study for Blockchain in Healthcare: "MedRec "prototype for electronic health records and medical research data MedRec: Using Blockchain for Medical Data Access and Permission Management." In: *IEEE Technology and Society Magazine* (2016), pp. 1–13. DOI: 10.1109/OBD.2016.11. URL: https://bit.ly/2mrsp5O.

[33] A. Albeyatti, M. Tayeb, A. Yin, P. Xiu, W. Alkhalidi, R. Miller, N. Furness, B. Mustafa, K. Koscik, D. Ebbitt, V. Riviere, J. Povey, S. Ankri, T. Robinson, O. Ayaz, and C. Djaja. *Medicalchain (Whitepaper)*. 2018, pp. 1–42.

[34] D. Flymen. *blockchain.py*. 2017. URL: https://github.com/dvf/blockchain/blob/master/blockchain.py.

[35] Flask Team. *Flask 1.0.2 documentation*. 2010. URL: http://flask.pocoo.org/docs/1.0/ (visited on 04/01/2018).

[36] Postman Team. *Postman Docs*. 2012. URL: https://www.getpostman.com/docs/v6/ (visited on 04/01/2018).

[37] *Fernet Spec*. URL: https://github.com/fernet/spec/blob/master/Spec.md (visited on 06/05/2018).