FEDERAL UNIVERSITY OF CEARÁ

CENTER OF SCIENCE

DEPARTMENT OF COMPUTING

POST-GRADUATION PROGRAM IN COMPUTER SCIENCE

ANTONIO EMERSON BARROS TOMAZ

SECURITY AND PRIVACY-PRESERVING OF DATA IN MOBILE HEALTH SYSTEMS: AN APPROACH BASED ON NON-INTERACTIVE ZERO-KNOWLEDGE PROOF AND BLOCKCHAIN

FORTALEZA

2021

ANTONIO EMERSON BARROS TOMAZ

SECURITY AND PRIVACY-PRESERVING OF DATA IN MOBILE HEALTH SYSTEMS: AN APPROACH BASED ON NON-INTERACTIVE ZERO-KNOWLEDGE PROOF AND BLOCKCHAIN

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Computer Science

Advisor: Prof. Dr. José Neuman de Souza

Co-advisor: Prof. Dr. José Cláudio do Nascimento

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

T615s Tomaz, Antonio Emerson Barros.
Security and privacy-preserving of data in mobile health systems : an approach based on non-interactive zero-knowledge proof and blockchain / Antonio Emerson Barros Tomaz. – 2021.
137 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2021.
Orientação: Prof. Dr. José Neuman de Souza.
Coorientação: Prof. Dr. José Cláudio do Nascimento.

1. Privacy-Preserving. 2. mHealth. 3. Blockchain. 4. Resource-Limited Devices. 5. Cryptography. I. Título.

CDD 005

ANTONIO EMERSON BARROS TOMAZ

SECURITY AND PRIVACY-PRESERVING OF DATA IN MOBILE HEALTH SYSTEMS: AN APPROACH BASED ON NON-INTERACTIVE ZERO-KNOWLEDGE PROOF AND BLOCKCHAIN

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Computer Science

Approved on: 07/04/2021

EXAMINATION BOARD

Prof. Dr. José Neuman de Souza (Advisor)
Federal University of Ceará (UFC)

Prof. Dr. José Cláudio do Nascimento (Co-advisor)
Federal University of Ceará (UFC)

Prof. Dr. Emanuel Bezerra Rodrigues
Federal University of Ceará (UFC)

Profa. Dra. Atslands Rego da Rocha
Federal University of Ceará (UFC)

Prof. Dr. Joaquim Celestino Júnior
State University of Ceará (UECE)

Prof. Dr. José Augusto Miranda Nacif
Federal University of Viçosa (UFV)

Dedicated to Fernanda and Lara.

ACKNOWLEDGEMENTS

I would like to formally declare my gratitude to many people in my family, friends, co-workers, and friends I made in the university labs, but this would certainly occupy many pages of this document.

Therefore, I would like to thank, in a particular way, only my mother for the encouragement, my advisor, professor Neuman, and my co-advisor, professor Cláudio, for the significant contribution to the elaboration of this work and the continuous motivation throughout the journey of my Ph.D.

I would also like to express my gratitude to the professors of the examination board for having accepted to review and contribute to this work, and the Federal University of Ceará, especially to the campus of Crateús.

“How many roads must a man walk down before
you call him a man?
(...) The answer is blowing in the wind.”

(Bob Dylan — Blowin’ in the Wind)

ABSTRACT

People of different ages have used miniaturized mobile devices with wireless communication capabilities and integrated with biosensors as wearable accessories to collect health data regularly. This type of medical assistance, which uses mobile devices to monitor patients and offer healthcare services remotely, is known as mHealth. The mHealth devices are typically wearable and have resource-limited, so many mHealth resources are managed through a smartphone. In this scenario, one of the most worrying issues involves communication between the monitoring devices and the smartphone. When the communication uses Bluetooth, it is standard for the device to be paired with the smartphone; but generally, it is not exclusively associated with a specific mHealth application. This feature can allow for a data theft attack. Thus, to address this problem, the present work proposes an authentication scheme based on Non-Interactive Zero-Knowledge Proof (NIZKP) — a cryptographic primitive lightweight enough to run on mHealth devices with resource-limited. In order to preserve patient's privacy throughout the system, this work uses blockchain technology to address the issues of storage, management, and sharing of data. Through smart contracts, the blockchain assumes the role of a decentralized authenticator that guarantees access to data only to legitimate users. As there is no privacy in the standard public blockchain, this work presents a scheme in which the data transmitted, stored, or shared is protected by Attribute-Based Encryption (ABE). Here, the data owner can share the encrypted data, which is associated with an access policy, and he/she himself/herself has the ability to distribute the secret keys to legitimate users to decrypt the data. Privacy-preserving and data security in electronic health record systems, including mHealth systems, are currently among the biggest concerns for patients. Given this scenario of concerns, the proposal presented in this work addresses all these issues holistically, proposing a model for constructing a mHealth system that guarantees the security and privacy of data from end to end, with robust access control and fully managed by the patient. We also provide a real implementation of our proposal using an Arduino Nano to represent the data collection device and the Ethereum blockchain to control access to the data. The results obtained prove that, even with a high level of security, it is possible to implement this scheme on resource-limited devices, requiring low execution time and little memory space.

Keywords: Privacy-Preserving. mHealth. Authentication. Blockchain. Resource-Limited Devices. Zero-Knowledge Proof. Cryptography.

RESUMO

Pessoas de diferentes idades têm usado dispositivos móveis miniaturizados com capacidade de comunicação sem fio e integrados a biossensores como acessórios vestíveis para coletar dados de saúde regularmente. Este tipo de assistência médica, que usa dispositivos móveis para monitorar pacientes e oferecer serviços de saúde remotamente, é conhecido como *mHealth*. Os dispositivos *mHealth* são, normalmente, vestíveis e com recursos limitados; de modo que muitos recursos da *mHealth* são gerenciados por meio de um *smartphone*. Neste cenário, um dos problemas mais preocupantes envolve a comunicação entre os dispositivos de monitoramento e o *smartphone*. Quando a comunicação usa *Bluetooth*, é padrão que o dispositivo seja emparelhado com o *smartphone*; mas, geralmente, ele não está exclusivamente associado a um aplicativo *mHealth* específico. Essa característica pode permitir um ataque de roubo de dados. Assim, para enfrentar esse problema, o presente trabalho propõe um esquema de autenticação baseado em Prova de Conhecimento Nulo Não-Interativa (NIZKP) — uma primitiva criptográfica leve o suficiente para ser executada em dispositivos *mHealth* com recursos limitados. Para preserva a privacidade do paciente em todo o sistema, este trabalho também enfrenta as questões de armazenamento, gerenciamento e compartilhamento de dados usando a tecnologia *blockchain*. Por meio de contratos inteligentes, a *blockchain* assume o papel de um autenticador descentralizado que garante acesso aos dados apenas a usuários legítimos. Como não há privacidade na *blockchain* pública padrão, este trabalho apresenta um esquema no qual os dados transmitidos, armazenados ou compartilhados são protegidos por Criptografia Baseada em Atributo (ABE). Aqui, o próprio paciente tem a capacidade de distribuir as chaves secretas de decifração dos dados para os usuários legítimos, e compartilha os dados cifrados, que são associados a uma política de acesso. Atualmente, a preservação da privacidade e a segurança dos dados em sistemas de registros eletrônicos de saúde, incluindo sistemas *mHealth*, estão entre as maiores preocupações dos pacientes. Diante desse cenário de preocupações, a proposta apresentada neste trabalho enfrenta de forma holística todas essas questões, propondo um modelo para a construção de sistemas *mHealth* que garante a segurança e privacidade dos dados de ponta a ponta, com controle de acesso robusto e totalmente gerenciado pelo paciente. Também fornecemos uma implementação real da nossa proposta usando a um Arduino Nano para representar o dispositivo de coleta de dados e a blockchain Ethereum para controlar o acesso aos dados. Os resultados obtidos comprovam que, mesmo com um alto nível de segurança, é possível implementar esse esquema

em dispositivos com recursos limitados, exigindo baixo tempo de execução e pouco espaço de memória.

Palavras-chave: Privacidade. Segurança. mHealth. Autenticação, Blockchain. Dispositivos com Recursos Limitados. Prova de Conhecimento Zero. Criptografia.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ABE	Attribute-Based Encryption
API	Application Programming Interface
BLE	Bluetooth Low Energy
CIA	Confidentiality, Integrity, and Availability
CP-ABE	Ciphertext-Policy ABE
CRS	Common Reference String
DDoS	Distributed Denial-of-Service
DH	Diffie–Hellman
DLP	Discrete Logarithm Problem
DMB	External Device Mis-Bonding
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EHR	Electronic Health Record
EOA	External Owned Account
EVM	Ethereum Virtual Machines
GDPR	General Data Protection Regulation
HIPAA	American Health Insurance Portability and Accountability Act
IoT	Internet of Things
IPFS	InterPlanetary File System
KGC	Key Generation Center
KP-ABE	Key-Policy ABE
LGPD	*Lei Geral de Proteção de Dados*
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NIZKP	Non-Interactive Zero-Knowledge Proof
P2P	Peer-to-Peer
PHI	Personal Health Record
PKI	Public Key Infrastructure
PoS	Proof of Stake
PoW	Proof of Work

SECG	Standards for Efficient Cryptography Group
SHA	Secure Hash Algorithm
UTXO	Unspent Transaction Output
UUID	Universally Unique Identifier
WBAN	Wireless Body Area Network
WLAN	Wireless Local Area Networks
ZKP	Zero-Knowledge Proof

LIST OF SYMBOLS

$\#E$	Number of points on the elliptic curve
$(G, *)$	A group G that has a certain binary operation $*$
$(\mathbb{Z}, +)$	Group of integers under addition operation $+$
$\mathbb{A}$	Set of descriptive attributes an ABE scheme
A_i	The i-th attribute of an ABE scheme
$\mathtt{AES}$	Advanced Encryption Standard
$\mathtt{D}$	A decryption algorithm
$\mathtt{D_{TM}}$	Deterministic Turing machine
$\mathcal{D}_{adm}$	Administrator device
$\mathtt{Dec_{AES}}$	AES symmetric decryption algorithm
$\mathtt{Dec_{ABE}}$	ABE decryption algorithm
$\mathtt{E}$	An encryption algorithm
E	An elliptic curve
$E(\mathbb{F}_p)$	An elliptic curve over a finite field
$\mathbb{F}_p$	A finite field in which p is a sufficiently large prime number
G	Generating point on an elliptical curve $E(\mathbb{F}_p)$
$\mathcal{H}$	A hash function
ID_d	Monitoring device identifier
ID_u	Data users identifier
$\mathcal{K}_{\mathbb{A}}$	User's secret key generated from a set of attributes $\mathbb{A}$
L	A language over an alphabet Σ
$\mathcal{M}_{\mathcal{K}}$	Master key of an ABE scheme
$\mathbb{N}$	Set of natural numbers
$\mathtt{ND_{TM}}$	Non-deterministic Turing machine
$\mathbb{P}$	Access policy in an ABE scheme
NP	Class of problems solvable in polynomial time by a non-deterministic Turing machine

$\mathbf{P}$	Class of problems solvable in polynomial time by a deterministic Turing machine
$\mathcal{P}$	A prover in the NIZKP system
$\mathcal{P}_{dev}$	A prover monitoring device NIZKP system
$\mathcal{P}'$	A dishonest prover in the NIZKP system
$\mathcal{P}_{\mathcal{K}}$	Public key in an ABE scheme
$\mathbb{Q}$	Set of rational numbers
Q_a	Public key in the ECDH protocol
Q_b	Bob's public key in the ECDH protocol
Q_d	Monitoring device private public
$\mathbb{R}$	Set of real numbers
S	Share secret key in ECDH protocol
S_a	Share secret key of verifier $\mathcal{V}_{app}$
S_d	Share secret key of prover $\mathcal{P}_{dev}$
$\mathcal{S}_{\mathbb{A}}$	Secret key for a set of attributes $\mathbb{A}$ in an ABE scheme
$\mathtt{TM}$	Turing machine
$\mathbb{U}$	Attributes universe in ABE
$\mathcal{V}$	A verifier in the NIZKP system
$\mathcal{V}^*$	A malicious verifier in the NIZKP system
$\mathcal{V}_{app}$	A verifier application in the NIZKP system
$\mathbb{Z}$	Set of integer numbers
$\mathbb{Z}_p$	Finite cyclic group of order p
$\mathtt{add}_{user}$	User blockchain address
$\mathtt{add}_{con}$	Smart contract address
$\mathtt{c}$	A ciphertext
$\mathtt{data}_{\mathtt{sim}}$	The health data set
$\mathtt{data}_{\mathtt{enc}}$	The encrypted health data set
$f(n)$	Function of n, where n to represent the length of the input.

$\mathtt{h}$	A hash value
k_a	Alice's shared secret in the DH protocol
k_a	Bob's shared secret in the DH protocol
$\mathtt{k_p}$	A public key in a cryptosystem
$\mathtt{k_s}$	A private key in a cryptosystem
$\mathtt{m}$	A clear-text message
$\mathtt{m'}$	Digitally signed message
$\mathtt{m_i}$	Each element of a clear-text message
$\mathtt{pac}_{\mathtt{nizkp}}$	Package containing NIZKP and encrypted health data
$\mathtt{s}$	String belonging to language L
Λ	Alice's public key in DH protocol
Υ	Bob's public key in DH protocol
Σ	An alphabet
γ	Data to be encrypted in an ABE scheme
$\gamma_{\mathbb{P}}$	Ciphertext with access policy $\mathbb{P}$ in an ABE scheme
κ	Share secret key in DH protocol
κ_a	Private key in the ECDH protocol
κ_b	Bob's private key in the ECDH protocol
κ_d	Monitoring device private key
π	Response to the challenge in NIZKP system
σ	Challenge in NIZKP system
υ	An integer chosen at random to generate a committed point in NIZKP over ECDPL
∞	A point at infinity in a elliptic curve

CONTENTS

1 INTRODUCTION

The Internet of Things (IoT) age has promoted technological progress in various areas of society. Thanks to the advent of the IoT, assistance for people undergoing medical treatment has improved significantly in recent years. Medical devices that were previously available only in hospitals can now be used by patients as wearable technological accessories. Wearable devices have been massively adopted to monitor individuals and offer health services remotely. This field of telemedicine is called mobile health, or just *mHealth*. Mobile health has improved the quality of care for patients outside the traditional clinical environment. This technology allows individuals to continuously monitor their health, collect physiological data, and share it with healthcare professionals.

The typical architecture of a mHealth system includes miniaturized mobile devices that collect health data using body sensors. These devices communicate via Wireless Body Area Network (WBAN). Ideally, patients should easily manage their data to choose with whom the data is shared. Once chosen, authorized healthcare professionals can recommend treatments based on the data collected without the patient's need to go to the health center. The use of mHealth technology facilitates health monitoring, diagnosis, and treatment of disease — by providing patients with greater convenience. However, mHealth brings several challenges to security and privacy. Mobile health systems require robust security schemes since personal health data is among the most sensitive. To address data security and privacy issues in a mHealth environment, we propose investigating two problems: (1) authentication of the monitoring devices; and (2) storage and access control of the data.

Authentication of the monitoring devices. The advent of mobile devices capable of collecting health data in real-time has boosted the healthcare industry. These devices are small and light enough to be worn on the body as a technological accessory. However, to make this possible, they are built with resource-limited, such as a small amount of RAM, a slow processor, and limited storage. Consequently, the data collection devices (or monitoring devices) typically use the patient's smartphone to pre-process the data before storing and sharing it with healthcare professionals. In this scenario, smartphones are considered essential devices. As they are designed as personal devices, it is usually assumed that the user is actually the owner of such a device. In this way, many resources in mHealth are managed through a smartphone. At the heart of the problem is the communication between the monitoring devices and the smartphone. Although there is great ease of interaction between devices, it is in this communication that there

is a critical vulnerability of the system. The reason is that Bluetooth technology — typically used in this type of communication — only pairs the monitoring device with the smartphone but does not associate the monitoring device with a specific application. Therefore, any application that gets permission to use the communication channel can access any device connected to that channel. Naveed *et al.* (2014) were the first to call attention to this vulnerability, and so motivated by that, they implemented an attack called External Device Mis-Bonding (DMB). The attack allows a malicious application to steal the patient's health data. In a variation of the attack, a fake device can inject incorrect data into the system. Thus, we consider that it is essential to develop a mechanism that associates the monitoring device with the official mHealth application, not only with the smartphone.

Storage and access control of the data. In general, traditional electronic health systems (eHealth) store and share data using local or cloud databases under the healthcare provider's control. This type of centralized architecture requires the patient to trust a third party to manage his/her data. A drawback that stands out in this scenario is that the patient completely loses control of his/her data — causing severe privacy problems. If the healthcare provider is unreliable, it may illegally share patient's data. Even in scenarios where the healthcare provider is reliable, it may be technically unable to maintain data security. Consequently, highly sensitive private data is at risk of being unduly exposed in the face of malicious attacks on the database. The *blockchain* becomes an integral part of this system to eliminate the need for a central entity that typically manages and shares the data. Essentially, a blockchain is a distributed ledger capable of maintaining an immutable log of transactions carried out on a network (ALI *et al.*, 2018). Blockchain was initially proposed by Satoshi Nakamoto[1] as a technology underlying the Bitcoin cryptocurrency (NAKAMOTO, 2008). An essential feature of the approach proposed here is that the control of access to data is patient-centered. The data owner shares his/her data using Attribute-Based Encryption (ABE), which gives the patient the exclusive authority to decide who accesses his/her data and what data each user can read.

The problems mentioned here make the mHealth ecosystem a challenging environment, especially for application developers. Certainly, mHealth systems have the potential to improve the quality of many traditional healthcare services as long as security and privacy issues are adequately addressed.

[1] Satoshi Nakamoto is a pseudonym. Despite significant effort, the identity of Bitcoin's creator has never been conclusively determined (WERBACH, 2018, p.17).

1.1 Primary motivation: pairing vulnerability

Wearable devices that collect patient health data and transfer it to healthcare professionals are the main components of a mHealth ecosystem. Typically, a health monitoring device reports the results to a mHealth application running on the patient's smartphone using a wireless communication channel, which is usually Bluetooth. This scenario requires devices with significantly lower power consumption; thus, the communication should preferably be carried out using Bluetooth Low Energy (BLE) technology. However, when a user pairs his/her smartphone to the mHealth device via BLE, the data communicated between the two devices is accessible to all the user's smartphone applications. Ideally, only the mHealth application should be allowed to communicate with the monitoring device. However, the smartphone operating system, specifically Android, is not able to establish the corresponding access control. Thus, the smartphone's operating system allows any application with permission to the Bluetooth channel to communicate with the monitoring device (NAVEED *et al.*, 2014). Note that operating system ensures that the smartphone is paired with a monitoring device but does not guarantee that the monitoring device is associated only with the specific mHealth application.

Naveed *et al.* (2014) studied this vulnerability and analyzed dozens of healthcare-related applications. They found that none of them is protected by any mechanism that authenticates the monitoring device to the mHealth application. To show the vulnerability of most devices in the market, they implemented an attack and called it DMB. This attack exploits a vulnerability in the Android security model used for communicating with an external device (such as Bluetooth devices). The attacker can steal data from an Android device or help an adversary to deploy a spoofed device that injects fake data into the mHealth system. The DMB attack's success is possible due to the lack of an exclusive association between the monitoring device and its corresponding mHealth application. In the absence of protection at the operating system level, manufacturers of monitoring devices have no choice but to implement security at the application layer to protect data privacy.

In this case, there are at least two possible attack scenarios, as reported by Naveed *et al.* (2014):

1. *Data-Stealing Attack.* A malicious application on the smartphone can steal the patient's data, provided it has Bluetooth permissions. The malicious capture of data can happen when the official mHealth application (the one the user expects to connect to the device) is not connected to the monitoring device. In this

scenario, the malicious application can determine the right time to download, using only its Bluetooth permission and side-channel information.

2. *Data-Injection Attack.* An attacker uses a malicious monitoring device to pair with the patient's smartphone. This attack can happen as follows:
   (i) the malicious application uses its Bluetooth permission to collect information about the legitimate monitoring device;
   (ii) the attacker clones the device using the information collected — such as MAC address, Universally Unique Identifier (UUID), and device name — and places the clone in the vicinity of the legitimate device;
   (iii) the smartphone pairs with the clone, instead of the legitimate device. Once this is done, the cloned device can inject fake data into the official mHealth application.

Naveed *et al.* (2014) further claim that, since the current Android design does not provide a means to link a specific application to an external device, manufacturers need to develop their own authentication mechanism — but this can be very challenging. The main reason is that the devices are generally elementary; they do not have sufficient resources to perform authentication operations, such as executing cryptographic functions.

1.2 Secondary motivation: privacy issues and the crisis of trust

The concept of privacy has been discussed over time and, in general, has proved to be a difficult notion to define. In the modern world, privacy is generally — but mistakenly, related to the digital processing of personal data. However, a broader notion of privacy is given by the philosopher Ferdinand Schoeman and shows that the concept of privacy is not limited to the environment created by technological devices.

> A person has privacy to the extent that others have limited access to information about him, limited access to the intimacies of his life or limited access to his thoughts or his body (SCHOEMAN, 1984, p.3).

Although the definition of privacy is quite broad, we are mainly concerned with privacy in a computing environment. When we bring this discussion to a more technological environment, it is essential to understand that there is a difference between privacy and anonymity, although these concepts are often confused. Privacy refers to the ability to hide the content of something, while anonymity is about hiding the author of it (BRADBURY, 2014). Based on this

definition, there is privacy when, for example, a person's email address, publicly associated with his/her real identity, is known to everyone, but the messages' content is protected. On the other hand, assuming that a person uses an untraceable connection, he/she can remain anonymous while exposing their ideas in public forums using a pseudo-identifier.

In the modern age, where there is massive data manipulation by electronic devices, data privacy raises great concern. This concern becomes even more severe when we talk about personal health data since they are among the most sensitive. However, this is also an age in which traditional institutions — such as government, the media, corporations, and nongovernmental organizations — have undergone a "profound crisis of trust," as is widely discussed by Werbach (2018, p.18) in his book entitled "*The Blockchain and the New Architecture of Trust*."

Filippi *et al.* (2020) reinforce this idea of a crisis of trust when they argue that after the abuse of information technologies for surveillance, dissemination of disinformation, and public coercion (such as Snowden's[2] revelations) have come to light, there was a growing loss of trust in government authorities — as well as big online platforms like Facebook, Google, and Twitter, which may be complicit in such abuses. The authors further claim that these events have triggered a new attitude towards socio-technical systems, whereby the requirement to trust third parties — be they companies or governments — is considered more of an obstacle than a help.

1.2.1 Blockchain in healthcare as a confidence machine

Blockchain is an emerging technology that enables new forms of distributed software architectures, where components can find agreements on their shared states for decentralized and transactional data sharing across an extensive network of untrusted participants and without relying on a central integration point that should be trusted by every component within the system (XU *et al.*, 2016). We will discuss the technical details of the blockchain's structure and working in Chapter 4, but first, in this section, we will discuss the benefits of blockchain in healthcare over traditional cloud computing.

According to Filippi *et al.* (2020), blockchain technology has emerged as a potential solution to the erosion of trust in traditional institutions and online intermediaries more generally, as it allegedly eliminates the need for trust between parties. The underlying premise of blockchain technology and its various applications is that users subject themselves to the authority of

[2] The then technical consultant Edward Snowden decided to reveal that the United States government monitored e-mails and videoconferences of those who used the services of companies such as Google, Skype, and Facebook. Besides, Snowden also revealed that telephone calls from American and foreign citizens were being monitored.

a technological system that they are confident is immutable, rather than to the authority of centralized institutions, which are deemed untrustworthy.

One point that can increase mistrust in traditional centralized organizations — such as healthcare providers or cloud database providers — is that their privacy policy is frequently not clear to users. In the face of that, we chose to leave this traditional model behind and elaborate our proposal on a disruptive platform like blockchain. Our interest to adopt blockchain as part of our proposal, instead of relying on a centralized and supposedly reliable cloud service provider, is supported by two important arguments below, defended by Antonopoulos (2014) — author of the book entitled "*Mastering Bitcoin*," one of the most important about Bitcoin and blockchain:

1. Blockchain technology enables a "shift from trusting people to trusting math".
2. Gradually, decentralized trust will be accepted as a new and effective trust model.

Filippi *et al.* (2020), in their article entitled *"Blockchain as a confidence machine: the problem of trust & challenges of governance,"* extensively discuss the subtle differences between the concepts of "trust" and "confidence" to argue that blockchain technology a "confidence machine." They conclude that trust presupposes an awareness of a specific element of risk — that is, trust is inherently connected with a trustee's ability to breach the trust that someone has conferred him/her. Confidence, in contrast, does not presuppose an acknowledgment of risk but rather an attitude of assurance. As opposed to trust, confidence does not require an individual to put herself into a vulnerable position because it does not operate under a condition of uncertainty. When used to describe a relationship with other people, institutions, or systems, a state of confidence involves a sense of predictability, which significantly contributes to reducing the feeling of risk and uncertainty that would otherwise be felt in entering into such a relationship.

Therefore, by including blockchain in our approach, we agree with Filippi *et al.* (2020) in claiming that blockchain-based systems are intended to produce confidence in a system — even not by eliminating trust altogether, certainly maximizes the degree of confidence in the system. Thus, the blockchain is a way to reduce the need for trust. In other words, we are replacing trust with confidence.

Overall, blockchain technology has a great number of features that can be utilized in healthcare. In the mHealth ecosystem, blockchain technology can produce significant benefits compared to a traditional cloud architecture, such as:

- it does not depend on trusted third parties, preventing service providers from collecting or sharing data without authorization;

- high availability;
- immutability guarantee;
- decentralized storage.

A more comprehensive comparison between cloud storage and computing (centralized) and blockchain (decentralized) can be observed from the Table 1.

Table 1 – Comparison between cloud storage and computing and blockchain.

	Cloud storage and computing	**Blockchain**
1	Single point of failure	Redundant copies of records assures security
2	Centralized control and administration	Consent algorithms using incentives to encourage users to participate in network governance
3	Powerful processors and high capacity storage	Expensive processing and limited storage capacity
4	Expensive infrastructure for the service provider	Processing power and storage is dedicated to the network by the users
5	Maintenance and protection of data is the responsibility of the cloud service providers	Trust-less network in which data and privacy protection is the user's responsibility
6	Far smaller electricity consumption	High electricity consumption especially for blockchain using Proof of Work (see Section 4.5.1)
7	Less latency	Public blockchain using Proof of Work have long transaction confirmation times
8	More scalable	Less scalable
9	Faster response time	Slow response time

Source: Houtan *et al.* (2020).

Regarding the drawback of blockchain in item 3 of Table 1, we included InterPlanetary File System (IPFS) (BENET, 2014) in the proposal to escape the high cost of storage in the blockchain. Regarding item 5, we have developed mechanisms based on robust cryptographic primitives (see Chapter 5) to ensure robustness against failure and data exposure. Regarding item 7, our approach is suitable for situations where health professionals monitor patients or athletes and offer health services and guidance remotely, without a state of emergency. Thus, for our approach, the benefits of using blockchain outweigh the benefits of cloud computing.

1.3 Scope statement

There are various types of mHealth applications, which can be used for different purposes. Wazid *et al.* (2016) point out some examples:

1. Remote monitoring of patient's health by healthcare experts as well as by the patient's relatives;
2. Doctors use the health data transmitted from the patient's monitoring devices to provide remote orientations;
3. Medical prescriptions issued by doctors are accessed and used by nursing staff and pharmacies for dispensing the required medicine;
4. The mHealth systems can also be used to schedule doctor 's appointments and schedule meetings with other health experts that monitor the patient's health.

Another type of application for mHealth systems involves detecting significant variations in the vital signs of chronic patients that may be life-threatening. In these situations, the application sends an alert to the medical staff responsible for monitoring the patient's health in order to obtain urgent help. Note that, in this scenario, the time it takes to collect the data, detecting variations, and sending the alert to doctors is a critical factor. This type of application is outside the scope of this work.

Within the scope of this thesis is the security and privacy-preserving of data on remote monitoring of patient's health by healthcare experts, but in scenarios where time is not a critical factor. Note that our approach is suitable for scenarios where the time between collecting health data and health professionals' analysis should not be a critical factor. The proposal presented in this work does not aim to send alerts to doctors about critical health events, where time is essential to preserve the patient's life. Our approach is not suitable for monitoring critical health situations since the collected data is transmitted to the blockchain — and on the blockchain, the processing time of the transactions carrying health records can exceed the maximum time required to issue an alert to a relief team and save the patient's life. Thus, our approach is suitable for situations where health professionals monitor patients or athletes and offer health services and guidance remotely, without a state of urgency or emergency.

1.4 Research contributions

This thesis makes the following two contributions.

1. The main contribution is to prevent attacks from malicious external devices that corrupt communication between mHealth devices and their official application. Our solution's construction uses an authentication mechanism, based on

Non-Interactive Zero-Knowledge Proof (NIZKP), lightweight enough to run on devices with limited computational resources.

2. Additionally, to address the issues of storage, management, and sharing of data, we propose a blockchain-based approach. However, there is no data privacy in the standard blockchain environment on transactions and storage. Thus, we present a scheme in which the security and privacy-preserving during the data transmission, storage, and sharing is based on ABE. The patient specifies an access policy, and he/she distributes the decryption keys to legitimate users of the system. In this way, we eliminate the risk that healthcare providers collect or share data without authorization. Therefore, we have a system that guarantees privacy and is entirely managed by the patient.

Our proposal offers security and privacy-preserving of health data from end-to-end — that is, from the collection by monitoring devices to data sharing controlled by the smart contract on the blockchain.

1.5 Organization of this thesis

The rest of this work is organized as follows. Chapter 2 provides an overview of mHealth systems focusing mainly on privacy and security issues. Chapter 3 reviews the definitions of algebraic structures and cryptographic primitives used in the construction of our proposal. Chapter 4 presents concepts, characteristics, and techniques regarding the structure and working of the blockchain. Chapter 5 starts with related works, then describes the proposal's construction — exposes the security protocols, the experimental results, and ends by discussing the results achieved towards privacy-preserving of patients in the mHealth system. Chapter 6 finalizes this thesis with the conclusion and future research opportunities.

1.6 Publication involved in this thesis

This thesis was developed based on Tomaz *et al.* (2020) — work elaborated by this author and published as follows:

TOMAZ, A. E. B.; NASCIMENTO, J. C. D.; HAFID, A. S.; SOUZA, J. N. D. Preserving privacy in mobile health systems using non-interactive zero-knowledge proof and blockchain. IEEE Access, IEEE, v. 8, p. 204441–204458, 2020.

2 OVERVIEW OF MOBILE HEALTH SYSTEMS

In this chapter, we present the main characteristics of the mHealth ecosystem — a field of Health Telematics that has promoted relevant changes in the healthcare industry. In recent years, significant improvements in mobile devices' performance and capabilities have made them suitable for real-time monitoring of a patient's vital signs. Using different types of wearable sensors, patients can collect their health data — such as pulse rate, temperature, respiration rate, blood glucose level, blood pressure — and then transmit it to a healthcare provider (WAZID *et al.*, 2016). However, the usage of mobile technologies in healthcare may multiply security and privacy risks. Security and privacy are a primary matter for individuals who decide to use a mHealth system. Patients expect the devices used to collect, transmit, and store their data to be able to protect them against unauthorized access and disclosure. Therefore, mHealth applications require robust security schemes since personal health data are among the most sensitive.

2.1 Internet of things in healthcare

Internet of Things (IoT) consists of a network of devices that monitor, collect, and exchange data over the Internet to enable intelligent applications — such as healthcare, manufacturing, smart cities, transportation (YÁNEZ *et al.*, 2020). The IoT technology in the healthcare area is growing as patients are more willing to be involved in making decisions about their health (MCGHIN *et al.*, 2019a). Also, the facility of using advanced, reliable, and miniaturized devices to collect patient health data in real-time has attracted many healthcare providers' attention. The combination of IoT devices and smartphones to provide healthcare is an attractive characteristic that makes mHealth popular.

According to Kotz *et al.* (2016), mHealth refers to the use of mobile technologies — wearable, implantable, environmental, or portable — by individuals who monitor or manage their own health, perhaps with the assistance of individual caregivers or provider organizations. The technologies include four categories described below.

- *Physiological monitoring*: measuring, recording, and reporting physiological parameters such as heart rate and blood pressure.
- *Activity and behavior monitoring*: measuring, recording, and reporting movement and physical and social activity as well as health-related behaviors such as eating and addictive behaviors.

- *Information access*: accessing health-related data — for example, medical records, activity, or behavior data — and decision-support tools.
- *Telemedicine*: communication between patients and caregivers or providers — for example, a virtual doctor visit or a patient receiving personal encouragement from a caregiver support team.

Overall, wearable devices offer low cost, real-time access to dynamic information on diseases, disorders, behavior, social interactions, environmental toxins, metabolites, and a host of other physical and physiological variables (BAJWA, 2014).

For the healthcare industry, mHealth represents a more efficient and cheaper way to care for patients suffering from chronic diseases that require continuous monitoring. However, this same technology can cause severe damage to patients if the systems that collect, transmit, and process the data are not designed under robust security and privacy-preserving schemes. Exposure to personal health data can cause irreparable harm.

2.2 Security and privacy in mHealth

Security and privacy of patient's information are crucial issues in mHealth systems. All the benefits of mHealth — such as monitoring vital signs through wearable devices and remote medical assistance — can only be truly enjoyed if patients can be confident that their data is safe.

Security implies that data is protected from unauthorized users when being transferred, collected, processed, and safely stored (AL-JANABI *et al.*, 2017). Note that security is intrinsically related to Confidentiality, Integrity, and Availability (CIA). In the context of mHealth systems, Zubaydi *et al.* (2015) defines the CIA triad as follows.

- *Confidentiality*. Health data should be confidential and available only to authorized physicians in addition to the original user. Confidentiality violation could cause damage to the patient, since the attacker could use the eavesdropped health data in illegal activities.
- *Integrity*. mHealth systems need effective mechanisms to secure stored data, verify that data has not been altered by an unauthorized party, and verify that data was sent by a trusted party.

- *Availability*. Health data should be available to the patient and responsible entities such as physicians at any time and from anywhere.

Privacy is the right of an individual to be secured from unauthorized disclosure of information about oneself that is contained in an electronic repository (KUMAR; KUMAR, 2020). There is no universally accepted definition of "privacy" — since there are many privacy perspectives in our life: social, financial, medical, legal, political, and technological. So that, privacy is more than just keeping information secure from violation by others (BINDAHMAN; ZAKARIA, 2011). A straight definition of privacy is presented by Bansal *et al.* (2010) as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others."

There are currently several regulations and laws worldwide to ensure the privacy of individuals — such as the General Data Protection Regulation (GDPR) in Europe, the *Lei Geral de Proteção de Dados* (LGPD) in Brazil, and the American Health Insurance Portability and Accountability Act (HIPAA) in the United States, specifically focused on health data. For example, HIPAA (HSS, 2020) provides a set of instructions in order to preserve patient privacy. Mobile health is still not restricted by HIPAA, but developers can benefit from HIPAA security rule standards to achieve higher security level of mHealth applications (ZUBAYDI *et al.*, 2015). Two of these privacy-preserving recommendations are highlighted by Al-Janabi *et al.* (2017): (1) systems need user identification from both consumers and medical staff, and (2) only authorized person has the right to access sensitive data and applications. Note that these recommendations are usually achieved by implementing a user authentication mechanism and access control.

- *User authentication*. In a system that records sensitive data, such as mHealth, the system must verify that the patient or healthcare professional really has the identity that he/she claims before accessing any health records.
- *Access control*. Health records in a mHealth system can be requested by several legitimate health professionals, such as doctors, nurses, and physiotherapists. Therefore, an access control mechanism must ensure that some data can be accessed only by professionals who have been duly authorized by the patient.

Table 2 presents the main characteristics related to the concepts of security and privacy.

Table 2 – Security and privacy characteristics.

	Security	Privacy
1	Security is related to confidentiality, integrity, and availability of data	Privacy is related to the appropriate use of user's information
2	Encryption and decryption algorithms are used for security	Third party cannot use data without permission of data owners
3	Confidentiality is provided in security	Confidentiality is provided in security
4	Security offers the capability of being confident, the decisions are honored	Privacy offers to decide what and to whom information be shared and at what extent

Source: Kumar and Kumar (2020).

2.3 Security and privacy requirements in mHealth

Researchers have been massively investigating which security requirements make a healthcare system widely reliable and acceptable to its users. Frequently, requirements such as confidentiality, integrity, user authentication, access control, and data availability are mentioned as the main security and privacy requirements for mHealth systems (ZUBAYDI *et al.*, 2015; WAZID *et al.*, 2016; AL-JANABI *et al.*, 2017). Additionally, we consider the following more specific requirements to be fundamental to address security and privacy challenges in mHealth environment.

Lightweight and efficient data authentication mechanisms. In mHealth, wearable devices integrated with biosensors are placed directly on the person's body to measure certain body parameters, such as heart rate, respiratory rate, body movement, temperature, blood pressure, and sleep disorders. In this scenario, the security and privacy challenges begin with the medical devices that detect the patient's body condition. A successful attack on data collection devices allows the attacker to change the patient's health data. In this way, the doctor can receive fake information and, consequently, prescribe wrong treatments that, in extreme cases, can lead the patient to death. Therefore, retrieving data on source devices requires a strong authentication mechanism to prevent misuse of data. Besides, an effective data authentication mechanism must ensure that data is sent by a legitimate device and not by an imposter.

Sharing of data managed by the patient. The collection of health data from patients has become easier in recent years due to IoT devices' popularization and wearable medical devices. Thus, there is a significant concern about patient data privacy and how this data can be shared securely with healthcare providers. In general, health data collected from the patient or other medical records require privacy and access control rules. Such information is traditionally

stored in databases in the cloud, and data access is done by adopting privilege-based access. By default, only authorized users can access these records. However, in this traditional model, highly sensitive data is at risk of being exposed if the database is attacked. Besides, a problem that stands out is that the patient loses control of his/her data after the data is under the healthcare provider's control. In this case, the healthcare provider may share the data without authorization and without the patient's knowledge. Another problem is that patients typically share data with several healthcare providers, who maintain their own data controls. Consequently, data is spread across multiple servers, and patients face difficulties in having a unified view of the environment where the data is shared. Another problem is that when the patient completes treatment or changes healthcare provider, he/she cannot revoke access to data since the data is under the control of healthcare providers. These problems make the mHealth ecosystem a challenging environment, both for the patient and for entities involved in developing applications, which must focus on privacy-preserving and security of patient health data.

Trusted user authentication. After the patient decides with whom to share their data, the confidentiality of the data must be guaranteed. As the data processed in mHealth systems reveals the patient's health status, this information's malicious exposure can lead the patient to embarrassing or even humiliating situations. Thus, a fundamental requirement to be faced by a mHealth system is the authentication scheme. The patient's consent or access policy determines who can access his/her data. However, mHealth systems must implement ways to accurately identify professionals who are authorized to use the system. In general, there are many ways to establish user authentication.

Data immutability guarantee. During transmission and, later, when the data is already stored, it is essential to provide mechanisms to identify possible changes in the data. The data can lose its integrity because of malicious attacks or natural interference on the communication channel. In any situation, it is necessary to implement mechanisms that guarantee the immutability of the data.

Patient-centered data management. Most mHealth systems collect data about a person's physiology, physical activity, or social behavior. Later, it stores the data for analysis by healthcare providers. Data sharing raises the issue of consent: how, with whom, which, and at what level of granularity the data should be shared (KOTZ *et al.*, 2016). The administration of health data is a critical point for the mHealth system. In most cases, after data is shared, users lose control over it. Healthcare providers start to manage this data and, consequently, can share

this data with other entities for use in applications not authorized by patients. The ideal scenario suggests that data management should be patient-centered — that is, the patient decides with whom he/she wants to share his/her data.

High availability. As mHealth systems carry essential, sensitive, and potentially life-saving information, the network must be available at all times for use by patients and health professionals.

2.4 Threats and attacks in mHealth

Many features in mHealth can be delivered to patients and healthcare providers via smartphones. As smartphones are designed as personal devices, it is usually safe to assume that the user is actually the owner of the device. However, according to Kotz *et al.* (2016), some mobile application architectures raise privacy concerns. In particular, the Android platform, which makes up 80 percent of the smartphone operating systems on the market, has a degree of openness that supports strong innovation but also puts users at risk of privacy violations. Bajwa (2014) affirms that most of the mobile devices, especially smartphones, possess only rudimentary security settings, making them vulnerable to hackers seeking access to patients' sensitive data stored in Personal Health Record (PHI), Electronic Health Record (EHR), and financial system — or any other sensitive information on the healthcare's server.

When investigating security and privacy issues in healthcare applications, Ameen *et al.* (2012) and Wazid *et al.* (2016) pointed out some of the attacks that a malicious agent can use to obtain information in healthcare systems, especially those that use wireless sensor networks, as is the case with mHealth applications. The attacks are as follows:

Data steal. Attacker of the mHealth system always seeks to access the data server illegally so that he/she can steal the data and misuse it to achieve his/her malicious objective. One of the attacker's main purposes is to sell the stolen data to healthcare product manufacturers.

Data modification. The attacker can replace part or all eavesdropped information and send the modified data to the healthcare professional to achieve some illegal purpose. This type of modification can significantly affect the patient, as the doctor may recommend a mistaken treatment based on false data.

Impersonation. There are two ways to carry out this attack. An attacker may be able to include a fake device in the network — cloned from information obtained from a legitimate device — and then insert fake data into the system to deceive healthcare providers. Another way

is when an adversary of a mHealth system tries to impersonate the legitimate user — as such a patient or a doctor — and is able to collect the health data.

Eavesdropping. Wireless devices can send information over the wireless network without encryption, so any intruder can easily intercept communications between mHealth devices. Eavesdrop and capture a message that contains personal health information can lead the attacker to discover that the patient suffers from a specific disease.

Replaying. The attacker can eavesdrop on a valid message and later replays it to deceive legitimate users. By reusing the data from the intercepted message, the adversary can prove his identity to the other party to obtain information such as the session key that can allow him to communicate with other legitimate users of the mHealth system.

Service disruption. In mHealth systems with centralized architecture, an adversary may interrupt the system's services by performing a Distributed Denial-of-Service (DDoS) attack. By sending an enormous amount of fake request messages to overload the medical server, the attacker can leave the server overloaded to the point that it is too busy to respond to legitimate users' requests.

Disclosure. The confidentiality of the health data stored at the healthcare provider's server is a significant security issue. Private health information that is disclosed can cause harm to the patient's reputation and personal life.

2.5 Standard architecture of the mHealth ecosystem

Nowadays, a large amount of health data has been generated through wearable and easy-to-use mobile medical devices. Typically, the mHealth ecosystem includes several types of compact devices, wireless networks, and smartphones or tablets.

2.5.1 Characteristics of mHealth devices

Health monitoring devices, equipped with biosensors, are available on the market in the most diverse shapes and sizes, from those that can be used on the skin to those that are implantable. The biosensors are capable of measuring various patient vital signs and patterns of behavior. Regardless of the device's purpose, they have some characteristics in common, such as the following:

Compact. One of the main concerns around mHealth devices is comfort and ease

of use. Thus, these devices are designed in compact sizes to ensure mobility and preserve the patient's well-being.

Limited resources. Due to the compact size and low capacity battery, the devices require low power consumption since they are designed to work for long periods. Thus, monitoring devices are built with limited computational resources — this includes computational power and memory space.

Wireless connectivity. After collecting the patient's health data, the monitoring devices use some wireless technology to transmit it. Typically, monitoring devices send data to a device with more robust computational resources, such as a smartphone or tablet. There are many wireless connectivity standards; but, for mobile health devices, the decisive factors are low power, low cost, physical size, and ease of use (ALMOTIRI *et al.*, 2016). Among the wireless technologies suitable for use in the mHealth environment pointed out by AlMotiri *et al.* (2016), such as ZigBee, Near Field Communication (NFC), but BLE is most common.

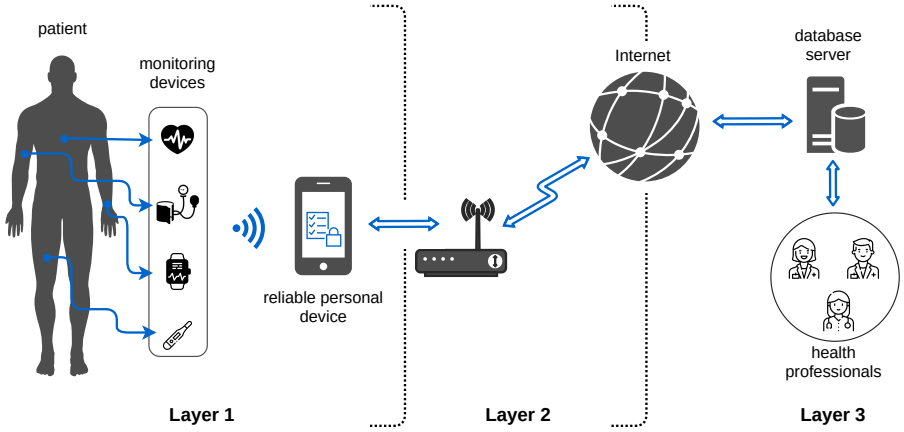
- BLE's main objective is to enable power-sensitive devices to be permanently connected to the network for a long time. BLE sensor devices may operate for many years without the need to replace the battery (ALMOTIRI *et al.*, 2016).
- BLE is gaining a high degree of importance among researchers due to its easy availability in gadgets and low power consumption (GHORI *et al.*, 2019).
- BLE provides a mechanism for small sensing devices to report their sensed data at set intervals and spend most of their time in a low power sleep mode, with a few commands allowed for configuring the node and requesting additional information if necessary (UHER *et al.*, 2016).

2.5.2 Wireless body area network

The monitoring devices of the mHealth ecosystem collect data through a Wireless Body Area Network (WBAN). WBAN is a wireless network used for communication among sensor nodes operating on, in, or around the human body in order to monitor vital body parameters and movements (RAMLI *et al.*, 2013). According to Li *et al.* (2017a), WBANs have broad prospects in the medical field since they can reduce healthcare providers' tasks, eliminate medical errors, increase hospital staff's efficiency, reduce the long-term cost of healthcare services, and improve the comfort of the patients.

In conventional WBANs, communication architecture consists of three layers (see Figure 1). Al-Janabi *et al.* (2017) provide an overview of the architecture in WBANs by describing the layers as follows.

Figure 1 – Communication layers in a WBAN.



Source: elaborated by the author based on the model by Al-Janabi *et al.* (2017).

Layer 1: Intra-WBAN communication. At this layer, the interaction of the monitoring devices is limited around the patient's body. The devices transmit the data collected to a personal device with more high computational power within this layer, such as a smartphone or tablet. The personal device acts as a gateway, which transfers the information to the next level — an Internet access point present in Layer 2.

Layer 2: Inter-WBAN communication. This level establishes communication between the trusted personal device, usually using Wi-Fi, and a healthcare service provider (data user) through Internet access points. Essentially, communication at this level is responsible for connecting WBANs to other systems or networks so that users can easily retrieve information.

Layer 3: Beyond-WBAN Communication. The data is taken to the healthcare providers' database servers over the Internet. Note that in some cases, the patient's device, such as a smartphone, can connect directly to Layer 3 using a 4G/5G connection without the need for an Internet access point. The healthcare provider's database is a crucial part of the system, as it accommodates the patient's health history and profile.

3 CRYPTOGRAPHIC PRELIMINARIES

Historically, cryptography has been seen as a technique for protecting valuable information, capable of making an incomprehensible message so that only the legitimate recipient is able to decipher it. In this traditional scenario, cryptography's main objective is to guarantee the confidentiality of communication between two distant parties. Nowadays, cryptography is concerned with achieving many more goals. Since the 1970s, problems such as constructing unforgeable digital signatures and designing fault-tolerant protocols have also been considered as falling within the domain of cryptography (GOLDREICH, 2007, p.1). Given this statement, a definition that best represents modern cryptography is as follows:

Definition 1 (***Cryptography***)**.** *Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication (MENEZES et al., 2018, p.4).*

In this chapter, we will present the cryptographic primitives used in the rest of this work. The concepts and techniques that we will review here are fundamental to our proposal's construction presented in Chapter 5. Occasionally, ideal scenarios will be presented to exemplify some concept or technique. To follow the tradition of work on information security, Alice and Bob will be the actors in the scenarios presented in this work.

3.1 Public key cryptography

Until the 1970s, the only way to encrypt information was through *symmetric cryptography*, also known as *secret key cryptography*. This cryptography method uses the same key to encrypt and decrypt a message. The sender and receiver agree to use a particular key and then share it between them. Note that the central challenge is how to share the key securely between the sender and receiver of the message. This problem of symmetric cryptography became known as the *key distribution problem*.

To solve the key distribution problem, Diffie and Hellman (1976b) introduced the concept of *asymmetric cryptography*, also called *public key cryptography*. This method uses a key pair, which consists of a private key and a public key. So, one key is used to encrypt, and the other key is used to decrypt. In their seminal work about asymmetric cryptography, Diffie and Hellman (1976b) established the conditions for a cryptosystem of this type to be built; however,

they did not present an algorithm that, effectively, performed the encryption and decryption of messages. The first cryptosystem based on the concept of public key cryptography was only presented two years later by Rivest *et al.* (1978a), which became known as RSA in honor of its creators — Rivest, Shamir, and Adleman. Diffie and Hellman (1976b) claim that there must be a mathematical relationship between the public key and the private key to ensure that if a message is encrypted with the private key, only the public key can decrypt it and vice versa. Thus, the following elements are essential for the implementation of a public key cryptosystem.

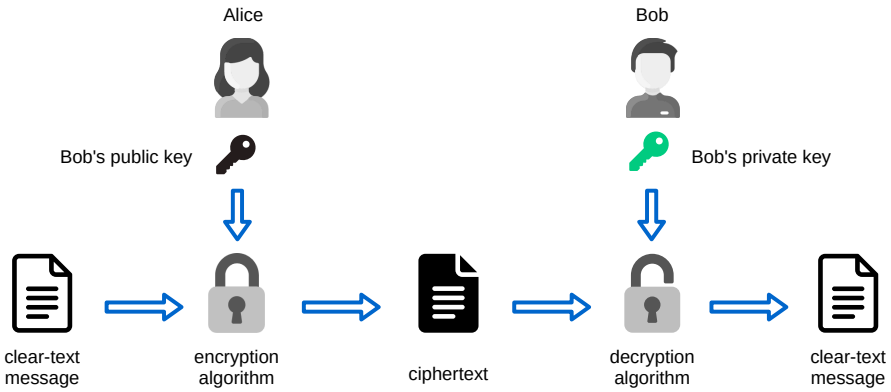
- A message to be encrypted $\mathtt{m} = \{\mathtt{m}_1, \mathtt{m}_2, \cdots, \mathtt{m}_n\}$, where $\mathtt{m}_i$ is each element of the clear-text message. In this context, the clear-text means the original message.
- A public key $\mathtt{k}_p$, for each participant, which is intentionally distributed by the owner or stored in public repositories. The key is simply a numerical value that will be used when encrypting or decrypting a message.
- A private key $\mathtt{k}_s$ corresponding to public key $\mathtt{k}_p$, for each participant, stored in a secure place, known only to the owner.
- An encryption algorithm $\mathtt{E}$ that takes a public key $\mathtt{k}_p$ and a clear-text $\mathtt{m}$ as input and outputs a ciphertext, $\mathtt{E}(\mathtt{k}_p, \mathtt{m}) = \mathtt{c}$. Ciphertext means a message that is non-understandable, and only the one who has the corresponding private key can decipher it.
- A decryption algorithm $\mathtt{D}$ representing the inverse operation of $\mathtt{E}$, which takes a private key $\mathtt{k}_s$ and a ciphertext $\mathtt{c}$ as input and outputs a clear-text, $\mathtt{D}(\mathtt{k}_s, \mathtt{c}) = \mathtt{m}$.

In addition to being usually used to encrypt information to ensure confidentiality, public key cryptography can also ensure message authenticity, so that the following two scenarios are supported.

Ensuring message confidentiality. In the first scenario, a user (a person or a computing device) aims to guarantee a message's confidentiality. In this case, the user, acting as the sender, encrypts the clear-text with the receiver's public key. Only the receiver's private key can decipher the ciphertext. Therefore, only the receiver user can access the message's content; hence, the message's confidentiality is guaranteed. Note that any public key cryptosystem is based on a fundamental assumption: *only the owner knows the private key*. Figure 2 helps visualize this scenario better, where we assume Alice and Bob as users of this scheme — Alice is the sender user, and Bob is the receiver user. As previously stated, the scheme adopted in this first scenario guarantees confidentiality but does not guarantee the message's authenticity and

non-repudiation. *Authenticity* means that the sender is univocally identified. *Non-repudiation* is a consequence of the message's authenticity — it is a property that prevents the sender of the authentic message from denying that he/she is the author of the message. On the other hand, when a message is transmitted by a user whose identity has not been confirmed, the user can deny that he/she is the author of the message to avoid unpleasant consequences.

Figure 2 – Use of public key cryptography to ensure the confidentiality of the message.

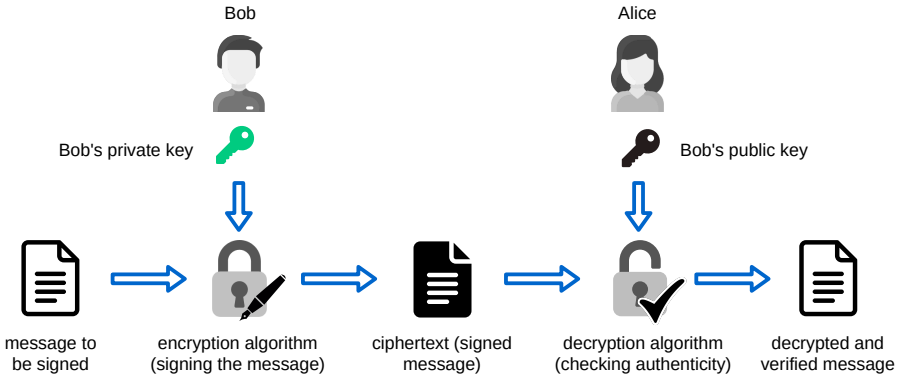


Source: elaborated by the author.

Ensuring the authenticity of the message. In this second scenario, the user aims to authenticate the message, so he/she uses the public key scheme as a method to obtain a digital signature. Essentially, theparties; they are limited to analyzing the data. *digital signature* is a cryptographic method that allows the creator of a digital document to be identified by the receiver. Thus, the sender must provide a message $\mathtt{m}$ and his/her own private key $\mathtt{k_s}$ as input to a cryptographic algorithm $\mathtt{E}$, which outputs a signed message $\mathtt{m'}$. After this, the message $\mathtt{m'}$ can only be decrypted with the sender's public key, which was previously distributed. Note that any user with the sender's public key can decrypt the message; therefore, there is no confidentiality. Indeed, there is no aim to protect the message's content in this scenario but rather to guarantee authenticity and non-repudiation. Figure 3 helps us visualize this scenario better, where we admit, once again, Alice and Bob as actors in this process. Suppose Alice receives a message supposedly signed by Bob — that is, encrypted with Bob's private key. The encryption of a message using the private key means the digital signature of the message, and the decryption process can be understood as verifying the signature. To confirm the authenticity of the message, Alice tries to decipher the message using Bob's public key. If the decryption process is successful, then the message was actually encrypted with Bob's private key; therefore, is confirmed the

message's authenticity. Note that it is necessary to assume that only Bob has access to his private key to confirm the message's authenticity.

Figure 3 – Use of public key cryptography to ensure the authenticity of the message.



Source: elaborated by the author.

Diffie and Hellman (1976b) established the following conditions for a public key cryptosystem to be feasible and secure.

1. It is computationally easy to calculate a key pair $(\mathbf{k_p}, \mathbf{k_s})$.
2. It is computationally hard to calculate the private key $\mathbf{k_s}$ from the public key $\mathbf{k_p}$.
3. It is computationally easy for a sender, with the receiver's $\mathbf{k_p}$, to calculate a ciphertext, such as $\mathbf{E}(\mathbf{k_p}, \mathbf{m}) = \mathbf{c}$.
4. It is computationally hard for an adversary, with only $\mathbf{k_p}$ and a ciphertext $\mathbf{c}$, to find the original message $\mathbf{m}$.

In contexts like this, where it is about how much computational resource is required to solve a given problem, a straightforward definition is essential to avoid ambiguous interpretations. One of the modern definitions of computational "easiness" or "tractability" is that the task is solvable in time-polynomial on the problem's input size. A task is considered "intractable" or "computationally difficult" if there is no polynomial-time algorithm to solve it (CREIGNOU *et al.*, 2001).

3.2 Formal languages versus problems

This section presents the basic definitions of formal languages and problem classes. We address these issues in this chapter because formal languages are often used to represent computational problems, and our proposal has as one of its mainstays the Zero-Knowledge Proof

(ZKP) — primitive cryptography that relies fundamentally on mathematical problems belonging of the class of **NP**-complete problems. Computationally intractable problems are intimately related to public key cryptography, which is of considerable importance in this work.

After the publication of the works of linguist Noam Chomsky (CHOMSKY, 1956; CHOMSKY, 1959), there was an intense concentration on the study of formal languages. Chomsky proposed a hierarchical classification of languages, which became known as *Chomsky hierarchy*. The main legacy of these works, especially for computer science, is to allow languages to be grouped into classes so that they can be hierarchized according to their complexity.

Every language, natural or artificial, is formed from the combination of basic symbols, which belong to a finite set called the *alphabet*. Formal languages are generated from precisely defined grammars. In this context, *grammar* is a finite set of rules that formally specify how symbols should be grouped to form the acceptable expressions of the language.

Definition 2 (***Language***)**.** *A language L over an alphabet Σ is a set of strings formed from Σ. Then $L \subseteq \Sigma^*$, where Σ^* represents the set of all strings of any length over the alphabet Σ, including the empty word ϵ.*

Usually, formal languages are used to represent computational problems (decision problems) since there is already a well-defined terminology for dealing with languages. A *decision problem* is one whose formulation leads to binary responses of the type *yes* or *no*. The investigation of the solvability of this type of problem can be treated as a *language recognition problem*. Therefore, it is possible to formulate several computational problems in terms of the verification of the membership of a string `s` in a language L. Demonstrating that a language is decidable is the same as demonstrating that the computational problem is solvable.

Definition 3 (***Decidable language***)**.** *A language L is said to be decidable if there is at least one Turing Machine* `TM` *such that:*

1. *For all string* $\mathtt{s} \in L$, `TM` *stops and accepts* `s`*;*
2. *For all string* $\mathtt{z} \in \Sigma^* - L$, `TM` *stops and rejects* `z`*.*

Informally, a *Turing machine* is a general-purpose computational model originally proposed by Alan Turing in his seminal paper (TURING, 1936). This model uses an infinite tape as its unlimited memory, and a tape head that moves around the tape to read and write symbols (SIPSER, 2012, p.165). A Turing machine supports two possible outcomes about an input string: *accept* or *reject* the string. If it does not enter an accepting or a rejecting state, the machine will

go on forever, never stopping — it will be in *loop*. However, there are variations of this model, including versions with multiple tapes or non-deterministic. Alternatively, computation on a non-deterministic Turing machine is based on a tree, whose branches correspond to different possibilities. If any branch of computing leads to the accepting state, the machine accepts its input. In contrast, a deterministic Turing machine has a single computational path to be followed. A formal presentation about Turing machines, both deterministic and non-deterministic, can be seen in Linz (2006, p.221).

The collection of strings that **`TM`** accepts is the language of **`TM`**, or the language recognized by **`TM`**. Thus, note that solving a decision problem is similar to the process of recognizing a given language *L*. For that, it is enough to check if the Turing machine *accepts* or *rejects* the input string and associate these outcomes with the *yes* or *no* answers of the problem represented. Input strings whose answers are negative are not recognized as belonging to the language *L*.

It is instructive to register that the relation between problems and languages was originally idealized by Edmonds (1965), who established the following correspondences for the general case, and were later reinforced by Garey and Johnson (1979, p.19):

- Decidable languages correspond to solvable decision problems;
- Undecidable languages correspond to problems of unsolvable decision.

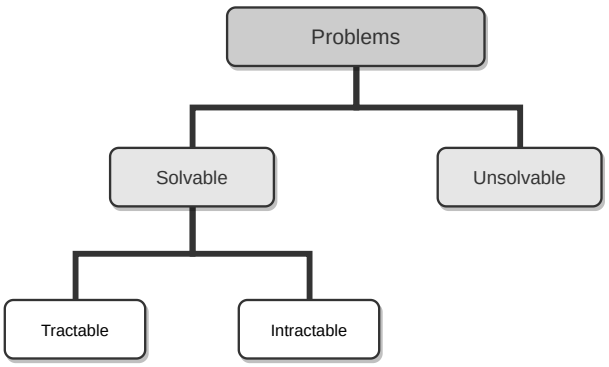
A problem is said to be *solvable* or *computable* if there is an algorithm (Turing machine) that solves this problem. A problem is said to be *unsolvable* or *uncomputable* if there is no algorithm to solve it (YAN, 2008, p.1).

Once confirmed that the problem is solvable, it is essential to establish parameters that indicate the degree of difficulty finding a solution to this problem. Some problems, although algorithmically solvable — i.e., even if there is an algorithm that solves the problem — can require an exorbitant amount of time, making such problems practically unsolvable or intractable. Thus, computational problems can be divided into two categories, as indicated by Yan (2008, p.1): solvable problems and unsolvable problems. The solvable problems can further be divided into *tractable* problems and *intractable* problems (see Figure 4).

In general, problems that can be solvable by some algorithm whose upper bound of complexity is polynomial are considered treatable (or easy). The problems solvable by algorithms whose lower bound of complexity is exponential are considered intractable (or hard).

To delineate the boundary between these two types of problems, Sipser (2012, p.278)

Figure 4 – Taxonomy of computational problems.



Source: elaborated by the author based on Yan (2008, p.2).

defines these bounds of the form n^d for d greater than 0, which are called *polynomial bounds*. Bounds of the form $2^{(n^e)}$ are called *exponential bounds* when e is a real number greater than 0. To classify problems according to their time requirement, we need some notions of time complexity classes — covered in the next section.

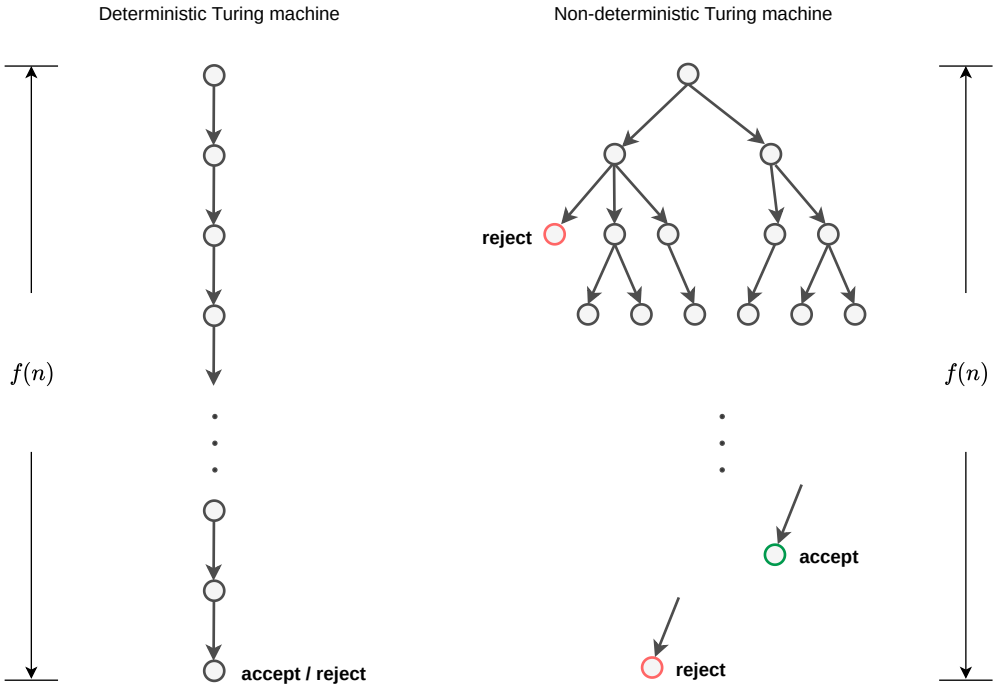
3.3 Time complexity class

To classify problems according to the amount of time demanded, it is first necessary to define how to measure a Turing machine's running time. Consider that problems solvable in polynomial-time on a Turing machine are exactly the same as those solvable in polynomial-time on a typical computer. This implies that Turing machines (or effective algorithms) and computers are equivalent concepts, and we shall use them interchangeably (YAN, 2008, p.1).

Definition 4 (***Time complexity in deterministic Turing machine***)**.** *Let* $\mathtt{D_{TM}}$ *be a deterministic Turing machine that halts on all inputs. The* ***running time*** *or* ***time complexity*** *of* $\mathtt{D_{TM}}$ *is the function* $f : \mathbb{N} \to \mathbb{N}$*, where* $f(n)$ *is the maximum number of steps that* $\mathtt{D_{TM}}$ *uses on any input of length n. If* $f(n)$ *is the running time of* $\mathtt{D_{TM}}$*, we say that* $\mathtt{D_{TM}}$ *runs in time* $f(n)$ *and that* $\mathtt{D_{TM}}$ *is an* $f(n)$ *time Turing machine. Customarily we use n to represent the length of the input (SIPSER, 2012, p.276).*

Definition 5 (***Time complexity in non-deterministic Turing machine***)**.** *Let* $\mathtt{ND_{TM}}$ *be a non-deterministic Turing machine that is a decider. The* ***running time*** *of* $\mathtt{ND_{TM}}$ *is the function* $f : \mathbb{N} \to \mathbb{N}$*, where* $f(n)$ *is the maximum number of steps that uses on any branch of its computation on any input of length n (SIPSER, 2012, p.283).*

Figure 5 – Measuring deterministic and non-deterministic time in a Turing machine.



Source: elaborated by the author based on Sipser (2012, p.283).

3.3.1 The class **P**

The class **P** plays a central role in complexity theory. Generally, polynomial differences in the algorithms' running time are considered small, whereas exponential differences are considered large (SIPSER, 2012, p.284).

Definition 6 (***Class* P**)**.** *A language L is in the complexity class* **P** *if L can be decided by the deterministic Turing machine* $\mathtt{D_{TM}}$ *and a polynomial $p(\cdot)$, such that:*

- *on input a string* $\mathtt{s}$, $\mathtt{D_{TM}}$ *halts after at most $p(|\mathtt{s}|)$ steps, and*
- *answers* ***yes*** *if* $\mathtt{s} \in L$*, or*
- *answers* ***no*** *if* $\mathtt{s} \notin L$*.*

According to Sipser (2012, p.286), **P** is invariant for all computation models that are polynomially equivalent to the deterministic Turing machine of a single tape. This means that **P** is a mathematically robust class that is not affected by the particularities of the computation model used. Besides, **P** roughly corresponds to the class of problems that are realistically solvable on a computer. In other words, when a problem is in **P**, there is a method of solving it that runs

in time n^d, where n is the size of the input, for some constant d. Whether this running time is practical depends on d and the application.

3.3.2 The class NP

In addition to the problems in class **P**, there are many other problems that the scientific community does not know if it is possible to solve in polynomial-time. In other words, there may be polynomial algorithms that solve these problems, but such algorithms are unknown; or, indeed, these problems cannot be solved in polynomial-time. In this case, as Sipser (2012, p.298) mentions, the best deterministic method currently known for deciding languages in class **NP** uses exponential time. However, it is unknown whether **NP** is contained in a smaller deterministic time complexity class. Nonetheless, Cook (1971) showed that many of these problems considered "intractable" can be solved in polynomial time by a non-deterministic Turing machine and, therefore, they are called **NP** *problems* (non-deterministic polynomial time problems). Moreover, this class of problems has an important characteristic called *polynomial verifiability*.

Definition 7 (***Polynomial verifiability***)**.** *A language L is polynomially verifiable if it has a polynomial-time verifier. A* ***verifier*** *for a language polynomial-time is an algorithm $\mathcal{V}$, where*

$$L = \{\, \mathtt{s} \mid \mathcal{V} \text{ accepts } \langle\, \mathtt{s}, \omega \,\rangle \text{ for some string } \omega \,\}.$$

The time of a verifier is measured only in terms of the length of $\mathtt{s}$. Sipser (2012, p.293) points out that the verifier uses additional information, represented by ω, to verify that a string $\mathtt{s}$ is a member of L. This information is called a *witness*, or *certificate*, or *proof* of membership of $\mathtt{s}$ in L. For polynomial verifiers, the witness ω has a polynomial length (in the length of $\mathtt{s}$) because that is all the verifier can access in its time-bound. Therefore, the complexity class **NP** is associated with a kind of computational problem whose solution, once provided, makes it possible to efficiently check its validity. In other words, although efficient algorithms (polynomial time algorithm) are not known to solve **NP** problems, it is possible to easily verify if a set of values provided as input is a valid answer to the problem.

Definition 8 (***Class* NP**)**.** *A language L is in* **NP** *if there is a boolean relation $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ and a polynomial $p(\cdot)$ such that:*

- *R_L can be recognized in deterministic polynomial time, and*

- $\mathbf{s} \in L$ *if and only if there is an* ω *such that* $|\omega| \leq p(|\mathbf{s}|)$ *and* $(\mathbf{s}, \omega) \in R_L$. *Such an* ω *is called a* ***witness*** *for membership of* $\mathbf{s}$ *in* L.

This information involving classes **P** and **NP** can be summarized, according to Sipser (2012, p.298), where the term "quickly" refers, in a non-strict way, to soluble in polynomial time, as follows:

- **P** = the class of languages for which membership can be *decided* quickly;
- **NP** = the class of languages for which membership can be *verified* quickly.

Thus, if $L \in \mathbf{P}$, then $L \in \mathbf{NP}$ because, if there is a polynomial-time algorithm to decide L, the algorithm can easily be converted to a two-argument verification algorithm that simply ignores any witness and accepts exactly the input strings that it determines to be in L. Therefore, $\mathbf{P} \subseteq \mathbf{NP}$ (CORMEN *et al.*, 2009, p.1064). However, the question of whether $\mathbf{P} = \mathbf{NP}$ is one of the biggest unresolved problems in computational complexity theory. Although it is unknown whether $\mathbf{P} = \mathbf{NP}$, most researchers believe that $\mathbf{P} \neq \mathbf{NP}$.

3.3.3 The class of problems NP*-complete*

Cook (1971) suggested that, among **NP** problems, there are specific problems whose individual complexity is related to that of the entire class. Such problems became known as **NP**-complete. This class of problems has a peculiar characteristic: if there is a polynomial-time algorithm that solves any of these problems, all problems in class **NP** can also be solved in polynomial-time, that is, $\mathbf{P} = \mathbf{NP}$. However, no polynomial-time algorithm has been discovered so far for any **NP** problem. This reinforces the belief of computer scientists that $\mathbf{P} \neq \mathbf{NP}$.

An essential concept for dealing with **NP**-complete problems is the concept of *reducibility in polynomial-time*. Intuitively, a problem $\mathcal{A}$ is said to be reduced to problem $\mathcal{B}$, if any instance of $\mathcal{A}$ can be efficiently reformulated as an instance of $\mathcal{B}$, so that a solution for $\mathcal{B}$ can be used to solve $\mathcal{A}$ (SIPSER, 2012, p.300). More precisely, using the terminology of formal languages to represent the problem, it is said that an L_1 language is reducible in polynomial-time to the language L_2, written $L_1 \leq_P L_2$, if there is a computable function in polynomial time $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that, for all string $\mathbf{s} \in \{0,1\}^*$, $\mathbf{s} \in L_1$ if and only if $f(\mathbf{s}) \in L_2$. The function f is called the *reduction function* from L_1 to L_2. The notion of polynomial time reduction provides a formal way of defining the class of **NP**-complete languages.

Definition 9 (***Class of problems* NP-*complete***). *A language L is* **NP***-complete if:*

- *L is in* **NP***, and*
- *all language L' ∈* **NP** *is polynomially reducible to L.*

3.4 Elliptic Curve Cryptography - ECC

The use of Elliptic Curve Cryptography (ECC) was initially proposed by Miller (1986) and Koblitz (1987). ECC has been widely used in recent years because it offers smaller key size, bandwidth savings, and faster implementations when compared to RSA cryptosystem (SEN, 2012, p.91). These features make ECC suitable for use in IoT environments, where, in general, devices have limited resources. Table 3 shows the key sizes needed to achieve a certain level of security for some encryption algorithms.

Table 3 – Key size of some cryptosystem to obtain different security levels.

Algorithms	**Security level in bits**			
	80	**128**	**192**	**256**
RSA	1024	3072	7680	15360
DL (discrete logarithm)	1024	3072	7680	15360
ECC	160	256	384	512
Symmetric	80	128	192	256

Source: Franco (2014), Bafandehkar *et al.* (2013), Bos *et al.* (2009).

Elliptic curves cryptosystems are closely related to certain mathematical abstractions, such as groups, rings, fields, and other concepts that we will address the following.

3.4.1 Groups

ECC is based on the generalized discrete logarithm problem; thus, finding a cyclic group is the first step to building a cryptosystem over elliptic curves. A group is the simplest algebraic structure. Then let us start our discussion about ECC with a formal definition of a group given by Lee (2018, p.38).

Definition 10 (***Group***). *A group* $(G, *)$ *is a set G, that has a binary operation* $*$*, satisfying the following conditions:*

1. ***Closure***. $a * b \in G$ *for all* $a, b \in G$*;*
2. ***Associativity***. $(a * b) * c = a * (b * c)$ *for all* $a, b, c \in G$*;*

3. ***Existence of identity***. *There exists an $e \in G$ such that $a * e = e * a = a$ for all $a \in G$ (the element e is called the identity of the group);*
4. ***Existence of inverses***. *For each $a \in G$, there exists a $b \in G$ such that $a * b = b * a = e$ (the element b is called the inverse of a, and is denote as a^{-1} or $-a$).*

A group operation is defined over a set of points. If $*$ is a group operation, we say that G is a group *under* $*$. Note that in group theory, we can use the operation $*$ for both additive groups, where $*$ denotes the addition, as well as for multiplicative groups, denoting multiplication. In fact, the group operation associates to each ordered pair (a, b) of elements in G an element $(a * b) \in G$.

A group need not necessarily have the commutative property. However, some groups have this property, and we call it especially abelian groups, in honor of the mathematician Niels H. Abel (1802 - 1829).

Definition 11 (***Abelian group***)**.** *A group G is said to be **abelian** or **commutative** if the operation $*$ is commutative, that is, $a * b = b * a$, for all $a, b \in G$.*

For cryptography, the groups mentioned above do not play a significant role because they have an infinite number of elements. For cryptographic applications, we need finite groups.

Definition 12 (***Finite group***)**.** *A group $(G, *)$ is finite when G has a finite number of elements. The **order** of the group is the number of elements in the set G, denoted by $|G|$.*

Cryptographic schemes over elliptic curves are strongly related to the cyclic behavior of a group and its generator element. The meaning of cyclic groups is given formally by Smart (2015, p.462), below.

Definition 13 (***Cyclic groups and generators***)**.** *A cyclic group G is a group which has an element g such that each element of G can be written in the form g^n for some $n \in \mathbb{Z}$. If this is the case then one can write $G = \langle g \rangle$ and one says that g is a **generator** of the group G, or that G is generated by g.*

Thus, a finite cyclic group with n elements is obtained from the powers g^i, such as $\{a^0, a^1, a^2, \cdots, a^{n-1}\}$, where g is a generator of the G. According to Paar and Pelzl (2009, p.213), cyclic groups have interesting properties, and the most important ones for cryptographic applications is the following: "for every prime p, $(\mathbb{Z}_p^*, *)$ is an abelian finite cyclic group." That

means that the abelian group of every prime field is cyclic. This has far-reaching consequences in cryptography, where these groups are the most popular ones for the building of cryptosystems based on discrete logarithm.

For the scope of this work, we will use a cryptographic scheme based on elliptic curves over finite fields. In the following sections, we present essential concepts about fields.

3.4.2 Rings

Informally, Smart (2015, p.468) defines a *ring* as an additive finite abelian group with an extra operation, usually denoted by multiplication, such that the multiplication operation is associative and has an identity element. The addition and multiplication operations are linked via the distributive law, such as $a \cdot (b + c) = a \cdot b + a \cdot c$.

More formally, a ring is defined by Paar and Pelzl (2009, p.16) as follows.

Definition 14 (*Ring*). *The integer ring $\mathbb{Z}_m$ consists of:*

1. *The set $\mathbb{Z}_m = \{0, 1, 2, \cdots, m-1\}$*
2. *Two operations "+" and "×" for all $a, b \in \mathbb{Z}_m$ such that:*

   (i) $a + b \equiv c \bmod m, (c \in \mathbb{Z}_m)$

   (ii) $a \times b \equiv d \bmod m, (d \in \mathbb{Z}_m)$

3.4.3 Fields

Informally, a field is a set that contains an additive and a multiplicative group. Paar and Pelzl (2009, p.92) formally defines fields as follows.

Definition 15 (*Field*). *A field F is a set of elements with the following properties:*

- *All elements of F form an additive group with the group operation "+" and the neutral element 0;*
- *All elements of F except 0 form a multiplicative group with the group operation "×" and the neutral element 1;*
- *When the two operations, addition and multiplication, are mixed, the distributivity law holds, i.e.: for all a,b,c ∈ F : $a(b + c) = (ab) + (ac)$.*

Note that if F is a field with its two binary operations "+" and "×", then $(F, +)$ is a commutative group; $(F \setminus \{0\}, \times)$ is a commutative group.

In cryptography, there is a special interest in fields with a finite number of elements, called *finite fields* or *Galois fields*. This is the most important structure for our proposal. According to Paar and Pelzl (2009, p.93), the most intuitive examples of finite fields are fields of prime order. Elements of the field $GF(p)$ can be represented by integers $\{0, 1, \cdots, p-1\}$. The two operations of the field are integer addition and integer multiplication *modulo p*.

Definition 16 (***Prime Field***)**.** *Let p be a prime. The integer ring $\mathbb{Z}_p$ is denoted as $GF(p)$ or $\mathbb{F}_p$ and is referred to as a* ***prime field****, or as a* ***Galois field*** *with a prime number of elements. All non-zero elements of $GF(p)$ have an inverse. Arithmetic in $GF(p)$ is done modulo p.*

Note that, an integer ring $\mathbb{Z}_m$ where m is a prime, $\mathbb{Z}_m$ is not only a ring but also a finite field.

3.4.4 Elliptic Curves

Most applications that use public key cryptography still use RSA (RIVEST *et al.*, 1978a); however, this scenario has begun to change in favor of ECC. Some of the more modern public key cryptosystems are designed on elliptic curves. The RSA problem is related to its key size. For RSA to remain safe for use, it is necessary to increase the key size. In recent years, the key size has increased a lot. This increase in the key generates a processing overhead, and this makes this cryptosystem unsuitable for use in some systems. Especially, it is unsuitable for running on devices with limited resources, such as IoT devices.

According to Stallings (2014, p.295), the main attraction of ECC, compared to RSA, is that it "appears" to offer equal security for a far smaller key size, thus reducing processing overhead (see Table 3). Stallings (2014) also explains the use of the term "appear" in this context: he comments that although ECC theory has been around for some time, applications using ECC have only recently become popular, attracting a sustained cryptanalytic interest in probing for weaknesses. Therefore, the level of trust in ECC is not yet the same as that in RSA because it has not yet undergone the same scrutiny as the older systems.

Elliptic curves are geometric objects that have algebraic properties that allow the construction of cryptographic systems. The mathematical concept of elliptic curves maintains a direct link with the finite fields discussed previously. Let us start our discussion about elliptic curves with a formal definition, based on the definition given by Rubinstein-Salzedo (2018, p.143).

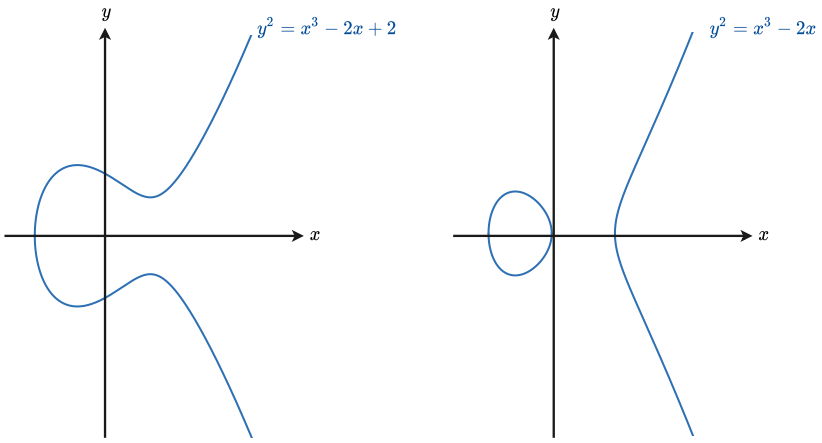
Definition 17 (***Elliptic Curve***)**.** *An elliptic curve E defined over a field $\mathbb{F}$, denoted by $E(\mathbb{F})$, is a collection of points $(x, y) \in \mathbb{F} \times \mathbb{F}$ that satisfy the equation*

$$y^2 = x^3 + ax + b \tag{3.1}$$

where a and $b \in \mathbb{F}$ and $4a^3 + 27b^2 \neq 0$.

This condition $4a^3 + 27b^2 \neq 0$ indicates that the polynomial $x^3 + ax + b$ has no repeated roots. In addition to the points (x, y), the curve must include an identity element, called *point at infinity* and denoted by ∞. Figure 6 represents an elliptic curve, depending on whether the cubic equation has one or three real roots.

Figure 6 – Elliptic curves of the form $y^2 = f(x)$. Left: f has one real root. Right: f has three real roots.



Source: Elaborated by the author.

As mentioned before, for cryptographic applications, there is a special interest in elliptic curves over finite fields. On an elliptic curve E defined over a finite field $\mathbb{F}_p = \{0, 1, \cdots, p-1\}$, in which p is a sufficiently large prime number, all variables and coefficients assume values in the set of integers $\mathbb{F}_p$. All operations must be calculated modulo p, such as

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{3.2}$$

with $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. All the points $P_i = (x_i, y_i)$ that satisfy this condition are said to belong to the elliptic curve.

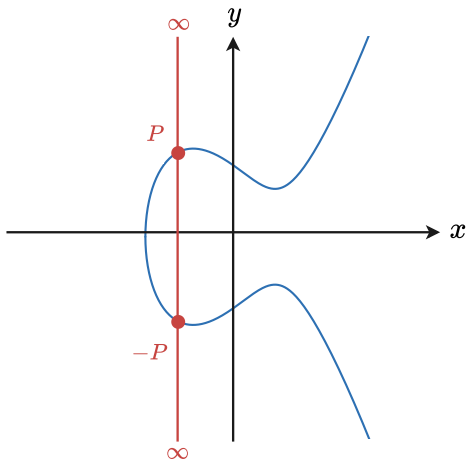
An interesting result of elliptic curves is that their set of points, together with a defined operation on that set, form an abelian group: the points will be the elements of the group, and the point at infinity (∞) is the neutral element for such an operation (FRANCO, 2014, p.66).

The curve $E(\mathbb{F}_p)$ can always be generated by at most two points — this is specific to the case of a finite field and may not happen for other fields like $\mathbb{Q}$. Assuming that P is a point in $E(\mathbb{F}_p)$, and let $\langle P \rangle$ denote the cyclic group consisting of all multiples of P in $E(\mathbb{F}_p)$, i.e., the group generated by P; then, the fact that at most two elements can generate $E(\mathbb{F}_p)$ can be understood as $E(\mathbb{F}_p) = \langle P \rangle \times \langle Q \rangle$ for some $P, Q \in E(\mathbb{F}_p)$ (RUBINSTEIN-SALZEDO, 2018, p.150). Sometimes $E(\mathbb{F}_p)$ can be generated by a single point — it is a cyclic group, but this does not always happen. However, when it does happen, E might be a "good" elliptic curve for certain aspects of cryptography. To define the group structure, we must define an operation — the addition operation. Geometrically, the rules for addition can be stated as follows.

Addition of a P and the point at infinity (∞). The point in infinity ∞ serves as the identity element of the group. Geometrically, the point at infinity is placed at infinity in the y axis. When we take $P + \infty$, the line that passes through these points is vertical and intersects the curve E at the point $-P$ (see Figure 7). Note that the reflection of the point $-P$ in relation to the horizontal axis is the point P. Thus, given a point $P \in E$ and the point ∞, we have

$$P + \infty = P. \tag{3.3}$$

Figure 7 – Elliptic curve identity element.



Source: Elaborated by the author.

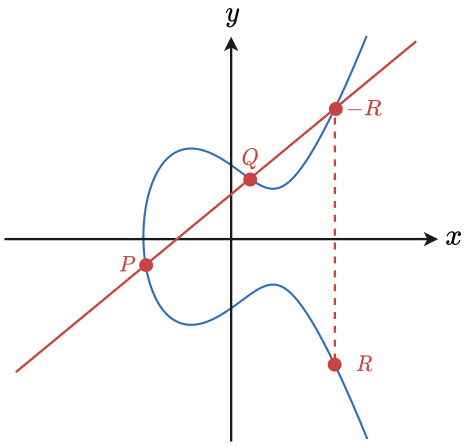
Addition of a point P and its inverse $-P$. The inverse of a point P is the point with the same x coordinate, but negative y coordinate, such as: if $P = (x, y)$, then the inverse is $-P = (x, -y)$. Note that a vertical line can join these two points. Figure 7 shows the geometric construction for the inverse of a point P. Thus, we have

$$P + (-P) = P - P = \infty. \tag{3.4}$$

Addition of two points. Given two different points, P and Q on the curve E, the addition of $P + Q$ results in a point $R \in E$. Geometrically, R is defined by drawing a line that passes through the points P and Q and intersects E at a third point $-R$, as shown in Figure 8. The point $-R$ is the reflection over the x axis of the point R. Thus, we have

$$P + Q = R. \tag{3.5}$$

Figure 8 – Elliptic curve addition.

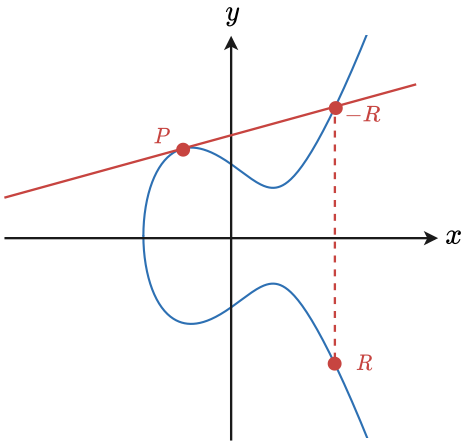


Source: Elaborated by the author.

Point doubling. To duplicate a point $P \in E$, a tangent line is drawn in P, which intersects the curve at a second point $-R$. The point $-R$ is the reflection over the x axis of the point R, as shown in Figure 9. Thus,

$$R = P + P = 2P. \tag{3.6}$$

Figure 9 – Elliptic curve point doubling.



Source: Elaborated by the author.

Algebraically, we can take the curve equation (Eq. 3.1), but we now consider it over prime fields $\mathbb{F}_p$. Given $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ two points of a curve $E(\mathbb{F}_p)$. The operation $P + Q = R = (x_3, y_3)$ is calculated as:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \bmod p, \\ y_3 &= \lambda(x_1 - x_3) - y_1 \bmod p, \end{aligned} \tag{3.7}$$

where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p, & \text{if } P \neq Q \\ \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p, & \text{if } P = Q. \end{cases} \tag{3.8}$$

The parameter λ, calculated in the Eq. 3.8, is the slope of the line through P and Q in point addition or the slope of the tangent through P in the case of point doubling.

3.5 Elliptic curve discrete logarithm problem - ECDLP

After mathematical preliminaries on groups, finite fields, and elliptic curves, the next step in exploring cryptosystems based on elliptic curves is to address the Discrete Logarithm Problem (DLP). Before Miller (1986) and Koblitz (1987) proposed the use of DLP over elliptic curves for cryptographic schemes, Diffie and Hellman (1976a) explored DLP based on modular arithmetic over multiplicative groups of the prime field $\mathbb{Z}_p^*$ for their proposed Key Exchange algorithm. Initially, we will here approve the background presented in the previous sections to discuss DLP. A formal definition is given by Paar and Pelzl (2009, p.217).

Definition 18 (*Discrete Logarithm Problem - DLP in* $\mathbb{Z}_p^*$). *Given is the finite cyclic group* $\mathbb{Z}_p^*$ *of order* $p - 1$ *and a generator element* $\alpha \in \mathbb{Z}_p^*$ *and another element* $\beta \in \mathbb{Z}_p^*$*. The DLP is the problem of determining the integer* $1 \leq x \leq p - 1$ *such that:*

$$\alpha^x \equiv \beta \bmod p. \tag{3.9}$$

Note that such an integer x must exist since α is a generator element and each group element can be expressed as a power of any generator element. Thus, the integer x is called the discrete logarithm of β to the base α,and we can formally write:

$$x = log_\alpha \beta \bmod p. \tag{3.10}$$

It is widely believed, although not proven, that there is no polynomial time algorithm for computing discrete logarithms in general. This means that, if the size of the group (or the prime p in the case of $\mathbb{Z}_p^*$) is very large, then it will take a very, very long time to compute discrete logarithms (RUBINSTEIN-SALZEDO, 2018, p.100).

DLP can be defined on a cyclic group, in which case it is called the generalized discrete logarithm problem.

Definition 19 (***Generalized Discrete Logarithm Problem - GDLP***)**.** *Given is a finite cyclic group G with the group operation $*$ and cardinality n. We consider a generator element $\alpha \in G$ and another element $\beta \in G$. The discrete logarithm problem is finding the integer x, where $1 \leq x \leq n$, such that:*

$$\beta = \underbrace{\alpha * \alpha * \cdots * \alpha}_{x\ times} = \alpha^x$$

The security of ECC, as well as other schemes — such as Diffie-Hellman Key Exchange and ElGamal encryption —, is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) over finite fields. Formally it is defined by Paar and Pelzl (2009, p.247) as follows.

Definition 20 (***Elliptic Curved Discrete Logarithm Problem - ECDLP***)**.** *Given is an elliptic curve E. We consider a generator element P and another element T. The DL problem is finding the integer d,where $1 \leq d \leq \#E$, and $\#E$ is the number of points on the curve, such that:*

$$T = \underbrace{P + P + \cdots + P}_{d\ times} = dP \tag{3.11}$$

We can say that the discrete logarithm operation is the inverse of point multiplication by a scalar. The multiplication of a point $P \in E(\mathbb{F}_p)$ by a scalar $d \in \mathbb{N}$, Eq.(3.11), is defined as adding P to itself d times. This operation results in a point $T \in E(\mathbb{F}_p)$. Thus, the discrete elliptic logarithm of T with respect to P is the integer d such that $T = dP$. If p, the order of the prime field $\mathbb{F}$, is sufficiently large, no efficient algorithm is known to solve this problem — that is, it is computationally infeasible to find the integer d. In ECDLP-based cryptosystems, the integer d is the private key, and the public key is a point $T = (x, y)$ on the E curve. However, knowing d and point P, it is computationally fast to calculate dP.

As mentioned before, ECC has attracted special attention because it achieves the same level of security, requiring less computational resources, when compared to traditional public key cryptography, such as RSA (RIVEST *et al.*, 1978b). This makes it ideal for security implementations on low-resource devices, such as the devices used in mHealth. For example, an implementation using ECC requires only a 256-bit key, while RSA needs to use a 3072-bit key to achieve the same level of security (BAFANDEHKAR *et al.*, 2013; BOS *et al.*, 2009).

3.6 Elliptic curve Diffie–Hellman - ECDH

Diffie–Hellman (DH) key exchange protocol (DIFFIE; HELLMAN, 1976a) was the first asymmetric scheme published in the open literature. DH protocol provides a practical solution to the key distribution problem — i.e., it enables two parties to derive a common secret key by communicating over an insecure channel (PAAR; PELZL, 2009, p.206). The basic idea behind the DH protocol is that exponentiation in $\mathbb{Z}_p$ (see Definition 18) — where p a large prime — is a one-way function and that exponentiation is commutative, i.e.,

$$\kappa = (g^a)^b \equiv (g^a)^b \bmod p. \tag{3.12}$$

Note that this protocol is not in itself a cryptosystem, but the value κ is the shared secret that can be used as the secret key between the two parties in a cryptography scheme symmetric. DH protocol over $\mathbb{Z}_p$ works as described in the Method 1.

Method 1 — Diffie–Hellman Key Exchange

1. Alice and Bob agree to use a large prime p and a generator $g \in \mathbb{Z}_p$. Both p and g are public.
2. Alice chooses a secret integer $a \in \mathbb{Z}_p$ and uses it to calculate $\Lambda = g^a \bmod p$. Then she sends Λ to Bob.
3. Bob chooses a secret integer $b \in \mathbb{Z}_p$ and uses it to calculate $\Upsilon = g^b \bmod p$. Then he sends Υ to Alice.
4. Alice calculates the secret key $k_a = \Upsilon^a \bmod p$.
5. Bob calculates the secret key $k_b = \Lambda^b \bmod p$.

Note that Alice and Bob compute a common secret key, $k_a = k_b$, such as

$$k_a = (g^b)^a = (g^a)^b = g^{ab} \bmod p = k_b. \tag{3.13}$$

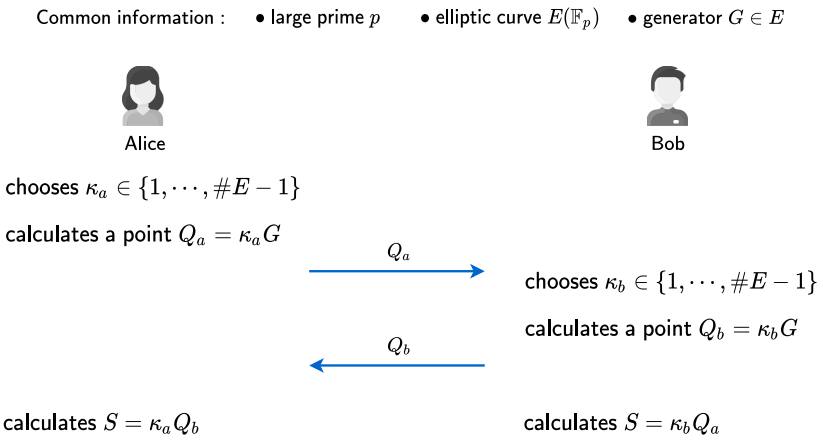
A version of the DH protocol works over elliptic curves — the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol. ECDH is based on classic DH protocol (Method 1). However, DH protocol is based on DLP defined in terms of modular arithmetic, while ECDH is based on ECDLP (see Definition 20).

Suppose that Alice and Bob want to share a secret key using ECDH protocol (see Method 2). Initially, Alice and Bob agree on three public parameters: a sufficiently large prime number p, an elliptic curve $E(\mathbb{F}_p)$, and a point $G \in E$.

Method 2 — Diffie–Hellman Key Exchange with Elliptic Curves

1. Alice randomly chooses an integer $\kappa_a \in \{1, \cdots, \#E - 1\}$, where $\#E$ is the number of points on the curve.
2. Alice calculates a point $Q_a = \kappa_a G$, where κ_a is the private key and Q_a is the public key of Alice. Then, she sends $Q_a \in E(\mathbb{F}_p)$ to Bob.
3. Likewise, Bob randomly chooses an integer $\kappa_b \in \{1, \cdots, \#E - 1\}$.
4. Bob calculates the point $Q_b = \kappa_b G$ where κ_b is the private key and Q_b is the public key of Bob. Then, he sends $Q_b \in E(\mathbb{F}_p)$ to Alice.
5. Alice calculates the shared secret key, using her private key and Bob's public key, $S = \kappa_a Q_b$.
6. Bob calculates the secret key, using his private key and Alice's public key, $S = \kappa_b Q_a$.

Figure 10 – Elliptic Curve Diffie–Hellman Key Exchange.



Source: elaborated by the author based on Paar and Pelzl (2009, p.250).

Note that S is the same point for Alice and Bob, such that

$$S = \kappa_a Q_b = \kappa_a(\kappa_b G) = \kappa_b(\kappa_a G) = \kappa_b Q_a. \tag{3.14}$$

Even though intruder Eve intercepts the points Q_a and Q_b that travel through the unsafe channel, it is computationally difficult to find the secret key S since she needs to know κ_a or κ_b. However, to compute κ_a or κ_b, she needs to solve the discrete logarithm problem.

3.7 Attribute-Based Encryption - ABE

Attribute-Based Encryption (ABE), initially proposed in Sahai and Waters (2005), is a cryptographic primitive that supports confidentiality and fine-grained access control over encrypted data. The decryption of data is authorized based on an access policy, defined by the data owner, considering a set of descriptive attributes. Loosely speaking, an access policy $\mathbb{P}$ is a rule that returns 0 or 1 given a set of attributes $\mathbb{A}$. We say that $\mathbb{A}$ satisfies $\mathbb{P}$ if and only if $\mathbb{P}$ answers 1 on $\mathbb{A}$. ABE schemes are classified into two types: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE).

- *CP-ABE*. In the CP-ABE scheme (BETHENCOURT *et al.*, 2007), each user is identified by a set of attributes and receives the secret key corresponding to those attributes from the authority. The set of attributes is assigned to the user's private key. The sender who wants to distribute a message will build an access policy associated with the ciphertext. Only users whose attributes satisfy the access policy can access the data.
- *KP-ABE*. In the KP-ABE scheme (GOYAL *et al.*, 2006), the ciphertext is labeled with a set of descriptive attributes, while a user's private key is associated with an access policy, which specifies the types of ciphertext that the key can decrypt.

In this thesis, we are especially interested in the CP-ABE scheme. The access policy adopted in the Chapter 5 is associated with the ciphertext, while the secret keys are labeled with a set of descriptive attributes. So, we set up our system using the CP-ABE scheme. According to Guo *et al.* (2014), compared to KP-ABE, CP-ABE is more appropriate for access control mechanisms since it allows the message encryptor to choose the access policy to decide who can access the messages.

3.7.1 Attributes and access policy

The attributes describe the users, and the access policy is used to label different sets of encrypted data.

Definition 21 (***Attributes universe***). *Let $\mathbb{U} = \{A_1, A_2, \cdots, A_n\}$ be an attributes universe, where A_i, for all $i = 1, 2, \cdots, n$, represents each of the attributes that can be used to describe one or more users of the CP-ABE schema (ODELU et al., 2017).*

Definition 22 (***Attributes set***). *Let $\mathbb{A}$ be an attributes set of a user, such that $\mathbb{A} \subset \mathbb{U} = \{A_1, A_2, \cdots, A_n\}$, $\mathbb{A}$ is represented with an n-bit string $a_1 a_2 \cdots a_n$, such that (GUO et al., 2014)*

$$\begin{cases} a_i = 1 : & A_i \in \mathbb{A} \\ a_i = 0 : & A_i \notin \mathbb{A}. \end{cases}$$

For example, let $n = 4$; the 4-bit string $A = 1011$ means the attribute set consists of the attributes $\{A_1, A_3, A_4\}$ from $\mathbb{U}$.

Definition 23 (***Access policy***). *Let $\mathbb{P}$ be an access policy (or access structure) defined by the data owner, such that $\mathbb{P} \subset \mathbb{U} = \{A_1, A_2, \cdots, A_n\}$, $\mathbb{P}$ is represented with an n-bit string $b_1 b_2 \cdots b_n$, such that (GUO et al., 2014)*

$$\begin{cases} b_i = 1 : & A_i \in \mathbb{P} \\ b_i = 0 : & A_i \notin \mathbb{P}. \end{cases}$$

For example, let $n = 4$; the 4-bit string $A = 1001$ means the access policy $\mathbb{P}$ requires the attributes $\{A_1, A_4\}$ from $\mathbb{U}$.

Thus, given an attribute set $\mathbb{A} = a_1 a_2 \cdots a_n$ and an access policy $\mathbb{P} = b_1 b_2 \cdots, b_n$ for all $i = 1, 2, \cdots n$, we say $\mathbb{A} \vDash \mathbb{P}$ if $a_i = b_i$ or $b_i = *$, where wildcard $*$ in $\mathbb{P}$ plays the role of "don't care" value. Note that we use the notation $\mathbb{A} \vDash \mathbb{P}$ to denote that $\mathbb{A}$ satisfies $\mathbb{P}$, and $\mathbb{A} \nvDash \mathbb{P}$ to denote that $\mathbb{A}$ does not satisfy $\mathbb{P}$. For example, in healthcare scenarios, given an access policy $\mathbb{P} = [\texttt{Hospital.X}, \texttt{Doctor}, *, \texttt{Brazil}]$, an attribute set $\mathbb{A}_1 = [\texttt{Hospital.X}, \texttt{Doctor}, \texttt{Woman}, \texttt{Brazil}]$, and an attribute set $\mathbb{A}_2 = [\texttt{Hospital.X}, \texttt{Nurse}, \texttt{Woman}, \texttt{Brazil}]$; then $\mathbb{A}_1 \vDash \mathbb{P}$, but $\mathbb{A}_2 \nvDash \mathbb{P}$ (GUO *et al.*, 2014; WANG *et al.*, 2018).

3.7.2 Ciphertext-Policy ABE

In traditional CP-ABE schemes, there are three types of entities: *Key Generation Center (KGC)*, the *encryptor*, and the *decryptors* (ZHANG *et al.*, 2015a; SONG *et al.*, 2019).

- *Key Generation Center*. KGC generates the private key to configure the system and generates/distributes the secret key for each user according to their attributes. The user uses this secret key to decrypt the ciphertext; however, he/she will only be successful if his/her attributes satisfy the corresponding access policy.
- *Encryptor.* The encryptor is the data owner. He/She encrypts the messages according to a designated access policy.
- *Decryptor*. The decryptors are the data users (ABE is a one-to-many public key encryption). They can decrypt the ciphertext successfully only if their attributes satisfy the corresponding access policy.

A drawback of the traditional scheme is that it depends on a trusted third party: the Key Generation Center. Usually, for the proper functioning of the system, we assume that KGC is an honest entity. However, in reality, there are many problems involved in this type of architecture, as indicated in Wang *et al.* (2018). Note that as KGC generates users' secret keys, it has the ability to decrypt all stored data. If KGC does not behave as an honest entity, it can abuse keys and leak private data. As such, the data owner loses the ability to control their own data. In practice, it is not easy to find KGC that is reliable.

A CP-ABE scheme consists of four algorithms, mathematically formalized in Bethencourt *et al.* (2007), and described below:

- **Setup** (ρ). The algorithm takes a predefined security parameter ρ and outputs a public key $\mathcal{P}_\mathcal{K}$ and a private master key $\mathcal{M}_\mathcal{K}$.
- **KeyGen** ($\mathcal{M}_\mathcal{K}, \mathbb{A}$). The key generator algorithm takes the master key $\mathcal{M}_\mathcal{K}$ and a set of attributes $\mathbb{A}$ as input. The output is a secret key $\mathcal{S}_\mathbb{A}$ corresponding to $\mathbb{A}$. Note that the attribute set describes one or more data users. Each data user will have his own secret key $\mathcal{S}_\mathbb{A}$.
- $\mathbf{Enc}_{\mathbf{ABE}}$ ($\mathcal{P}_\mathcal{K}, \gamma, \mathbb{P}$). The encryption algorithm takes the public key $\mathcal{P}_\mathcal{K}$, the information to be encrypted γ, and the access policy $\mathbb{P}$ as input. The output is the ciphertext $\gamma_\mathbb{P}$, which implicitly contains the policy $\mathbb{P}$.
- $\mathbf{Dec}_{\mathbf{ABE}}$ ($\mathcal{P}_\mathcal{K}, \gamma_\mathbb{P}, \mathcal{S}_\mathbb{A}$). The decryption algorithm takes the public key $\mathcal{P}_\mathcal{K}$, the encrypted information $\gamma_\mathbb{P}$, and the secret key of the data user $\mathcal{S}_\mathbb{A}$ as input. If $\mathbb{A}$, the set of user attributes, satisfies $\mathbb{P}$, then the algorithm decrypts the ciphertext, and the output is γ.

ABE is a one-to-many public key cryptographic primitive. The data owner encrypts

their data and defines various user groups who can decrypt it, as long as they satisfy the access policy. This property makes ABE very attractive for implementing fine-grained access controls.

3.8 Hash functions

Hash functions can be applied in several situations, but only their use in cryptography is of interest in this work. These functions belong to a group known as one-way functions.

Definition 24 (***One-way function***)**.** *A function f from a set X to a set Y is called a one-way function if $f(x)$ is "easy"*[1] *to compute for all $x \in X$ but for "essentially all" elements $y \in Im(f)$ it is "computationally infeasible" to find any $x \in X$ such that $f(x) = y$ (MENEZES et al., 2018, p.8).*

Loosely speaking, a *hash function* is a function $\mathcal{H}$ that receives a message $\mathbf{m}$ of arbitrarily size and returns a fixed-size string $\mathbf{h}$, called a *hash-value*, where, in general, $|\mathbf{m}| > |\mathbf{h}|$. This function can be denoted by

$$\mathcal{H}(\mathbf{m}) = \mathbf{h}.$$

Definition 25 (***Hash function***)**.** *A hash function is a unidirectional mapping $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^i$, for some $i > 1 \in \mathbb{N}$. In this mapping,* **D** *is the domain of the function representing the possible messages of arbitrary size, and* **I** *is the function image representing the possible hash values of size i; thus, the number of possible hash values is 2^i.*

Hash functions are directly related to verifying the integrity of messages. A hash function's central purpose is to generate a "tag" that provides a unique identifier for each message. Since set **D** is much larger than set **I**, two messages $\mathbf{m}$ and $\mathbf{m'}$ may be mapped to the same hash $\mathbf{h}$; this is known as a *collision*. A collision of the hash function $\mathcal{H}$ is when two messages $\mathbf{m} \neq \mathbf{m'}$ produce $\mathcal{H}(\mathbf{m}) = \mathcal{H}(\mathbf{m'})$. The way a hash function is constructed ensures that if there is any change in the original message, another hash-value will be generated even if a single bit is modified. In this way, it is possible to identify whether a message was tampered with during transmission.

To be used in cryptography, hash functions must satisfy some properties; if these properties are satisfied, they are called cryptographic hash functions (STALLINGS, 2014, p.323).

[1] See Section 3.2, which discusses computationally easy problems.

1. *Preimage resistant (one-way property)*. For any given hash value $\mathbf{h}$, it is computationally infeasible to find $\mathbf{m}$ such that $\mathcal{H}(\mathbf{m}) = \mathbf{h}$.
2. *Second preimage resistant (weak collision resistant)*. For any given data block $\mathbf{a}$, it is computationally infeasible to find $\mathbf{b} \neq \mathbf{a}$ with $\mathcal{H}(\mathbf{b}) = \mathcal{H}(\mathbf{a})$.
3. *Collision resistant (strong collision resistant)*. It is computationally infeasible to find any pair $(\mathbf{a}, \mathbf{b})$ such that $\mathcal{H}(\mathbf{a}) = \mathcal{H}(\mathbf{b})$.
4. *Pseudorandomness*. Output of $\mathcal{H}$ meets standard tests for pseudorandomness.

In addition to these properties required by the cryptographic hash functions above, every hash function must have two basic properties that are part of the definition itself:

- *Compression*. The function must map a string of bits of arbitrary size to a string of bits of fixed size.
- *Easy to compute*. Given a message $\mathbf{m}$, it is easy to compute $\mathcal{H}(\mathbf{m}) = \mathbf{h}$.

The safety of hash functions is directly related to the difficulty of generating collisions. The greater the difficulty of finding collisions, the greater the level of security of the function. According to Damgård (1990), a hash function is called collision-free if mapping messages of any size to fixed-length strings, but in such a way that finding messages $\mathbf{a}$ and $\mathbf{b}$ with $\mathcal{H}(\mathbf{a}) = \mathcal{H}(\mathbf{b})$ is a computationally hard problem.

The main hash functions are built based on a structure known as the Merkle-Damgård construction (MERKLE, 1990; DAMGÅRD, 1990). This method defines how to build collision-resistant hash functions from collision-resistant compression functions. The Merkle-Damgård construction uses a structure known as an iterated hash function that repeatedly uses a compression function. A compression function is a function that from a bit string of size $\mathbf{u}$ generates a bit string of size $\mathbf{v}$, where $\mathbf{u} > \mathbf{v}$. More precisely, a compression function is a mapping $\mathbf{C}$: $\{0,1\}^{\mathbf{u}} \rightarrow \{0,1\}^{\mathbf{v}}$, with $\mathbf{u}, \mathbf{v} \in \mathbb{N}$. If the compression function is collision-resistant, then the hash function built from it is also collision-resistant. Therefore, the problem of designing a secure hash function that takes messages of any size as input boils down to designing a collision-resistant compression function. (MERKLE, 1990; DAMGÅRD, 1990). According to Menezes *et al.* (2018), compression functions can be constructed using block ciphers, modular arithmetic, or even be built specifically for a given hash function.

Most hash algorithms considered to be safe are designed based on the Merkle-Damgård construction. Examples of hash algorithms considered safe by the National Institute of Standards and Technology (NIST) are the Secure Hash Algorithm (SHA) standard. SHA

standard comprises the SHA-224, SHA-256, SHA-384, and SHA-512 algorithms, which produce hash-values of 224, 256, 384, and 512 bits as output, respectively. These algorithms are used in several applications, including cryptographic systems.

3.8.1 Interactive zero-knowledge prof system

In the late 1980s, Goldwasser *et al.* (1989) introduced the concept of Zero-knowledge Proof (ZKP). Since its inception, traditional zero-knowledge proof and its many variants have been widely applied in cryptographic schemes. A ZKP system is a protocol that enables one party, called *prover*, to prove that some statement is true to another party, called *verifier*, but without revealing anything but the truth of the statement.

In general, these statements that the prover wishes to prove can be represented as a membership problem in languages. Thus, the prover must demonstrate to the verifier that a public statement α, in this context represented as a public string, belongs to a specific language L of the class **NP** (see Section 3.3.2). Thus, the zero-knowledge proofs capture a generalization of the class **NP**.

A traditional zero-knowledge proof protocol — proposed by Goldwasser *et al.* (1989) — consists of several rounds of information exchange between the prover and the verifier. At the end of the protocol, if the statement is, in fact, true, the verifier must be convinced of this. However, if the prover's statement is false, the verifier will discover the lie with a high probability. Each round is made up of 3-moves, which are three messages called *commitment*, *challenge*, and *response*. Initially, the prover generates a first message (commitment), which is the statement to be proved and sends it to the verifier. Then, the verifier randomly chooses a challenge and sends it to the prover. Finally, the prover calculates the response based on the challenge and sends it to the verifier.

A "proof" in this context is not a fixed and static object, but rather a randomized and dynamic (i.e., interactive) process in which the verifier interacts with the prover. Intuitively, one may think of this interaction as consisting of "tricky" questions asked by the verifier, to which the prover has to reply "convincingly" (ROSEN, 2007, p.4). Fundamentally, a proof reveals whether a statement is true.

A zero-knowledge proof system must satisfy three fundamental properties, which are described in a non-rigorous manner below.

1. *Completeness*. It is the ability of the prover to convince the verifier that specific

statements are true, provided that the prover has proof of this.

2. *Soundness*. It is the ability of the verifier to protect itself from being convinced of false statements, except with a very small probability.
3. *Zero-knowledge*. No malicious verifier can obtain extra knowledge from the interaction — that is, the interaction produces nothing (beyond conviction in the validity of the assertion).

According to Goldreich (2007, p.190), the definition of an interactive proof system refers explicitly to the two computational tasks related to a proof system: "producing" a proof and "verifying" the validity of a proof. In the following definition, the verifier's output is interpreted as its decision on whether to accept or reject the common input. Output 1 is interpreted as "accept," whereas output 0 is interpreted as "reject."

Definition 26 (***Interactive Proof System***)**.** *An n-round interactive proof system* $(\mathcal{P}, \mathcal{V})$ *between a prover* $\mathcal{P}$ *and a verifier* $\mathcal{V}$ *for a language* L *is a pair of random algorithms, modeled as interactive Turing Machines, such that* $\mathcal{V}$ *runs in probabilistic polynomial time and the following two conditions hold:*

- *Completeness: for every* $\alpha \in L$*,*

$$\Pr\left[\langle \mathcal{P}, \mathcal{V} \rangle(\alpha) = 1\right] \geq 2/3;$$

- *Soundness: for every* $\alpha \notin L$ *and every prover* $\mathcal{P}'$*,*

$$\Pr\left[\langle \mathcal{P}', \mathcal{V} \rangle(\alpha) = 1\right] \leq 1/3.$$

Sequential repetitions can amplify the constants 1/3 and 2/3 to get as close to 0 or 1, respectively, as desired.

Loosely speaking, it is said that an interactive proof system $(\mathcal{P}, \mathcal{V})$, between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for a language L, is *perfect zero knowledge* if there is an additional property (compared to Definition 26), in which $\mathcal{V}$ learns absolutely nothing — beyond the veracity of the statement — from the interaction with $\mathcal{P}$.

To show that a proof is zero-knowledge, we need to consider the verifier as a potential opponent, denoted by $\mathcal{V}^*$, which wants to get some extra knowledge from $\mathcal{P}$, in addition to the veracity of the statement α. Thus, the fundamental requirement that $\mathcal{V}^*$ "gains no knowledge" is met if any adversary strategy used by $\mathcal{V}^*$ produces nothing beyond the validity of the statement α. It is important to note that this is true for any efficient interaction strategy with $\mathcal{P}$, not necessarily

the honest way defined by the checker $\mathcal{V}$. Thus, it is clear that "zero-knowledge" is a property of the prover — this property captures the robustness of $\mathcal{P}$ against attempts to obtain knowledge by interacting with him.

The most precise definition of the "no gain" requirement can be formulated using the *Simulation Paradigm* postulated by Goldwasser *et al.* (1989). In such a paradigm, everything that a party can do on its own is not considered a gain of knowledge. In other words, a potentially malicious verifier $\mathcal{V}^*$ does not gain knowledge when interacting with $\mathcal{P}$ — with public input α — if all that $\mathcal{V}^*$ can calculate from the interaction with $\mathcal{P}$ can also be computed efficiently directly from the public input α, without any interaction. Essentially, this paradigm formulates that any feasible malicious behavior of $\mathcal{V}^*$ can be *simulated* by a benign behavior, which in practice is a feasible algorithm called *simulator* that receives only the public input (GOLDREICH, 2008, p.370). Therefore, the fact that such simulators exist means that there is no gain of extra knowledge through the malicious behavior of $\mathcal{V}^*$. The notion of zero-knowledge is based on the indistinguishability between two sets: the simulator's output and the interactions between the prover and the verifier. For this reason, this line of reasoning is called the simulation paradigm.

3.8.2 Perfect zero-knowledge

According to Goldreich (2008, p.370), the formulation of the zero-knowledge condition refers to two types of probability ensembles, where each ensembles associates a single probability distribution with each relevant entry — for example, a valid statement. In the case of interactive proof systems, the first ensemble represents the verifier's output distribution after interacting with a prover — on some common input, where the verifier is employing an arbitrary and efficient strategy (not necessarily the honest strategy specified). The second ensemble represents the output distribution of some probabilistic polynomial-time algorithm (a simulator), which receives only the corresponding input and does not interact with anyone. The basic paradigm of zero-knowledge asserts that for every ensemble of the first type, there is a "similar" ensemble of the second type. The interpretation given to the notion of similarity may be different. In this context, Goldreich (2008) says that the most strict interpretation, leading to *perfect zero-knowledge*, is that similarity means equality.

Definition 27 (***Perfect interactive zero-knowledge proof system***)**.** *An n-round interactive proof system* $(\mathcal{P},\mathcal{V})$ *for some language L is a* ***perfect zero-knowledge proof system*** *if for every probabilistic polynomial-time verifier* $\mathcal{V}^*$*, there is a probabilistic polynomial-time algorithm* $\mathcal{S}^*$*,*

so that for every $\alpha \in L$,

$$(\mathcal{P}, \mathcal{V}^*)(\alpha) \equiv \mathcal{S}^*(\alpha)$$

where,

- $(\mathcal{P}, \mathcal{V}^*)(\alpha)$ *is a random variable that represents the output of the verifier $\mathcal{V}^*$ after interacting with the prover $\mathcal{P}$ on the common entry α, and*
- $\mathcal{S}^*(\alpha)$ *is a random variable that represents the output of the algorithm $\mathcal{S}^*$ on the entry α. The algorithm $\mathcal{S}^*$ is called a simulator for the interaction of $\mathcal{V}^*$ with $\mathcal{P}$.*

3.8.3 Computational zero-knowledge

The definition of zero-knowledge in the previous section is quite rigorous and very difficult to achieve. A more relaxed interpretation is that "similarity" means *computationally indistinguishable* — that is, any efficient procedure fails to differentiate the two ensembles. This leads to the standard use of the term zero-knowledge, which can also be called *computational zero-knowledge*. Goldreich (2007) argues that, for practical purposes, there is no need to "simulate perfectly" the output of $\mathcal{V}^*$ after interacting with the prover $\mathcal{P}$. Instead, just generate a computationally indistinguishable probability distribution from the output of $\mathcal{V}^*$ after interacting with the provide $\mathcal{P}$. In this situation, we can understand computationally indistinguishable ensembles as being the same.

Definition 28 (***Computationally indistinguishable***)**.** *Two ensembles, $X \stackrel{def}{=} \{X_\alpha\}_{\alpha \in L}$ and $Y \stackrel{def}{=} \{Y_\alpha\}_{\alpha \in L}$ indexed by strings of a language L, are **computationally indistinguishable** if, for every probabilistic polynomial-time algorithm D, every positive polynomial $p(\cdot)$, and every $\alpha \in L$ long enough,*

$$| \Pr[D(\alpha, X_\alpha) = 1] - \Pr[D(\alpha, Y_\alpha) = 1] | < \frac{1}{p(|\alpha|)}.$$

That said, it is now possible to present a precise definition for a computational zero-knowledge proof system.

Definition 29 (***Computational interactive zero-knowledge proof system***)**.** *An n-round interactive proof system $(\mathcal{P}, \mathcal{V})$ for some language L is a **computational zero-knowledge proof system** (or just zero-knowledge) if, for every probabilistic polynomial-time interactive machine $\mathcal{V}^*$, there is a probabilistic polynomial-time algorithm $\mathcal{S}^*$, so that for every $\alpha \in L$, the two The following sets are computationally indistinguishable:*

- $\{(\mathcal{P}, \mathcal{V}^*)(\alpha)\}_{\alpha \in L}$ *is the output of the interactive machine* $\mathcal{V}^*$ *after it interacts with the interactive machine* $\mathcal{P}$ *on the common entry* α.
- $\{\mathcal{S}^*(\alpha)\}_{\alpha \in L}$ *is the output of the algorithm* $\mathcal{S}^*$ *on the entry* α . $\mathcal{S}^*$ *is called a simulator for the interaction of* $\mathcal{V}^*$ *with* $\mathcal{P}$.

3.8.4 Characteristics of interactive zero-knowledge proof systems

Three main features differentiate all known zero-knowledge proof systems from more traditional ones (BLUM *et al.*, 1988):

1. *Interaction*. The prover and the verifier interact repeatedly.
2. *Hidden randomization*. The verifier tosses coins that are hidden from the prover, and therefore unpredictable for him/her.
3. *Computational difficulty*. The prover inserts in his/her proofs the computational difficulty of some other problem.

3.8.5 Non-interactive zero-knowledge proof system

As the previous section shows, zero-knowledge proof systems are generally interactive: they involve the exchange of several messages between the prover and the verifier. However, in some scenarios, interactions are not desirable. In these situations, the standard approach uses a specific type of zero-knowledge proof, called Non-Interactive Zero-Knowledge Proof (NIZKP) — introduced by Blum *et al.* (1988), which consists of a single prover flow for the verifier after installing a single trusted configuration.

Section 3.8.1 describes protocols that do not make any assumption of a reliable configuration — that is, they are protocols with strong security guarantees. This scenario is known as *simple model*. However, according to Couteau (2017), the lack of any form of reliable configuration strongly restricts the viability of the system: several desirable properties related to the safety or efficiency of interactive proofing systems are proven to be unattainable in the simple model. On the other hand, the definition for NIZKP systems generally contains an assumption of initial security configuration. Thus, a fundamental question is whether there are assumptions of minimum reliable configuration that allow building a model in which NIZKP systems are efficient in practice and with strong security guarantees.

The common reference string model. In response to the question in the previous section, Blum *et al.* (1988) were the first to present an efficient model, with minimal reliable configuration assumptions, for building NIZKP. In this model, Blum *et al.* (1988) ensures that the standard interaction process for any ZKP can be eliminated. That is, the prover can, non-interactively and with zero-knowledge, convince the verifier of the veracity of a statement. To achieve this result, the random choice of the bit b representing the challenge, as mentioned in Section 3.8.1, can be replaced by a common string pre-shared between the prover and the verifier, which is honestly generated by a trusted third party in an initial setup phase. Such a model is known as the common reference string Common Reference String (CRS). Thus, NIZKP systems' definition generally contains an assumption of initial configuration. The CRS model consists of three entities: a prover, a verifier, and a bit sequence (the common reference string - **crs**) selected by a trusted third party and shared between the prover and the verifier (GOLDREICH, 2007). A **crs** is usually made up of uniformly random bits; for this reason, **crs** is also commonly referred to as a *common random string*. Each bit of the random sequence replaces the random challenge chosen by the verifier in interactive ZKP systems. In this case, the interaction is minimal (in fact, unidirectional: from the prover to the verifier). The CRS model enhances the simple model — in addition to the common input, i.e., the statement to be proved, the model provides both parties with access to **crs**, which is independent of the statements to be proved. The CRS model is not the only one that has been proposed for the NIZKP system. Santis *et al.* (1990) proposed the pre-processing model and, Cramer and Damgård (2004) proposed the secret key model. However, the most popular alternative to the CRS model is the Random Oracle (RO) model (BELLARE; ROGAWAY, 1993), which adopted by Fiat-Shamir heuristic discussed in Section 3.8.6.

Definition 30 (***Non-interactive zero-knowledge proof system***)**.** *A non-interactive zero knowledge proof system between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for some language L is a triple of probabilistic polynomial-time algorithms (*`Setup`*, $\mathcal{P}, \mathcal{V}$), so that:*

- `Setup` (1^{λ})*: takes a security parameter λ as input and generates a common reference string* **crs***;*
- $\mathcal{P}$ (**crs**,α, ω)*: takes* **crs***, a statement α, and a witness ω as input and output a proof π;*
- $\mathcal{V}$ (**crs**, α, π)*: takes* **crs***, a statement α, and a proof π as input and returns "1" if accepted the proof as valid and "0" if you reject the proof.*

This non-interactive proof system must satisfy the properties of completeness, solidity, and zero-knowledge defined below.

Completeness (perfect). Completeness states that a prover must be able to prove a true statement. A NIZKP system $(\mathtt{Setup}, \mathcal{P}, \mathcal{V})$ for a language L with relations R_L satisfies the (perfect) completeness property if, for every $\lambda \in \mathbb{N}$, for every statement $\alpha \in L$, and every witness ω such that $(\alpha, \omega) \in R_L$,

$$\Pr[\mathtt{crs} \leftarrow \mathtt{Setup}(1^\lambda);\ \ \pi \leftarrow \mathcal{P}(\mathtt{crs}, \alpha, \omega)\ :\ \mathcal{V}(\mathtt{crs}, \alpha, \pi) = 1] = 1. \tag{3.15}$$

For computational completeness, this requirement is relaxed, and we only consider malicious probabilistic polynomial-time prover, and the definition is changed so that the probability is close to 1.

Soundness (Computational). Soundness states that a malicious prover must not be able to prove a false statement; even when such prover deviates from the protocol. A NIZKP system $(\mathtt{Setup}, \mathcal{P}, \mathcal{V})$ for a language L with relations R_L satisfies the (adaptive) solidity property if, for all $\lambda \in \mathbb{N}$, for every statement $\alpha \notin L$ and every prover (adversary) $\mathcal{A}$,

$$\Pr[\mathtt{crs} \leftarrow \mathtt{Setup}(1^\lambda);\ \ (\alpha, \pi') \leftarrow \mathcal{A}(\mathtt{crs}, \alpha)\ :\ \alpha \notin L\ \wedge\ \mathcal{V}(\mathtt{crs}, \alpha, \pi') = 1] = \mathrm{negl}(\lambda). \tag{3.16}$$

Here, the adversary $\mathcal{A}$ has a false statement $\alpha \notin L$ regardless of $\mathtt{crs}$. In non-adaptive soundness, $\mathcal{A}$ chooses α before seeing $\mathtt{crs}$, while in adaptive soundness $\mathcal{A}$ can choose α based on $\mathtt{crs}$. This implies that adaptive soundness is a stronger notion of soundness.

Zero-knowledge (adaptive). The zero-knowledge property ensures that a non-interactive proof reveals nothing except the truth of the statement being proven. Like the interactive zero-knowledge proofs, this is also achieved here using the simulation paradigm. The simulator can generate $\mathtt{crs}$ and the simulated proof from an entry α. It is assumed that the simulator must have "extra power" to produce $\mathtt{crs}$ for itself and the simulation's trapdoor τ — that is, auxiliary information to be used by the simulator to generate an acceptable proof for α without knowing the witness. This trapdoor is used by the simulator but is not available to an adversary against the protocol. In other words, the simulator has access to more than just the common statement α to make simulation possible without a witness. Otherwise, an adversary would be able to create a convincing proof, even without a witness.

A NIZKP system $(\mathtt{Setup}, \mathcal{P}, \mathcal{V})$ for a language L with relations R_L satisfies the zero-knowledge property if it is possible to simulate the proof of a true statement without knowing the

witness. Formally, it is necessary to have a probabilistic polynomial-time simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for every $\lambda \in \mathbb{N}$,

- $\mathcal{S}_1(1^\lambda)$ takes a security parameter λ as input and generates a simulated $\mathbf{crs'}$ and a trapdoor τ;
- $\mathcal{S}_2(\mathbf{crs'}, \tau, \alpha)$ takes the pair $(\mathbf{crs'}, \tau)$ and a statement α as input and returns a simulated proof π';

and for every adversary $\mathcal{A}(\cdot)$ of polynomial-time, returns a declaration α and the witness ω such that $(\alpha, \omega) \in R_L$, both following ensembles are computationally indistinguishable:

$$\begin{aligned} &\Pr[\mathbf{crs} \leftarrow \mathtt{Setup}(1^\lambda);\ \ (\alpha, \omega) \leftarrow \mathcal{A}(\mathbf{crs})\ :\ \pi \leftarrow \mathcal{P}(\mathbf{crs}, \alpha, \omega)\ :\ \mathcal{A}(\mathbf{crs}, \pi) = 0] \\ &\Pr[(\mathbf{crs'}, \tau) \leftarrow \mathcal{S}_1(1^\lambda);\ \ (\alpha, \omega) \leftarrow \mathcal{A}(\mathbf{crs})\ :\ \pi' \leftarrow \mathcal{S}_2(\mathbf{crs'}, \tau, \alpha)\ :\ \mathcal{A}(\mathbf{crs'}, \pi') = 0]. \end{aligned} \tag{3.17}$$

This traditional definition of a NIZKP system corresponds to the notion of a *publicly verifiable* NIZKP system. This means that the verifier receives only public information, such as the common statement α and the witness ω. Alternatively, when there is an initial configuration phase, and there is the generation of secret information available only to the verifier, this variant is known as the *designated verifier* NIZKP system. Only the publicly verifiable model is considered in this work.

3.8.6 The Fiat-Shamir Heuristic

An n-round interactive ZKP system can be converted in a non-interactive ZKP system. A common way to obtain non-interactivity is by using CRS model (see Section 3.8.5), where a trusted third party produces a random common string that removes the need for a response from the verifier. The most popular alternative to the CRS model uses the Fiat-Shamir heuristic (FIAT; SHAMIR, 1987), where the responses from the verifier to the prover are replaced by a uniformly random output in the random oracle (RO) model. According to Iovino and Visconti (2019), RO model assumes the availability of a perfect random function to all parties.

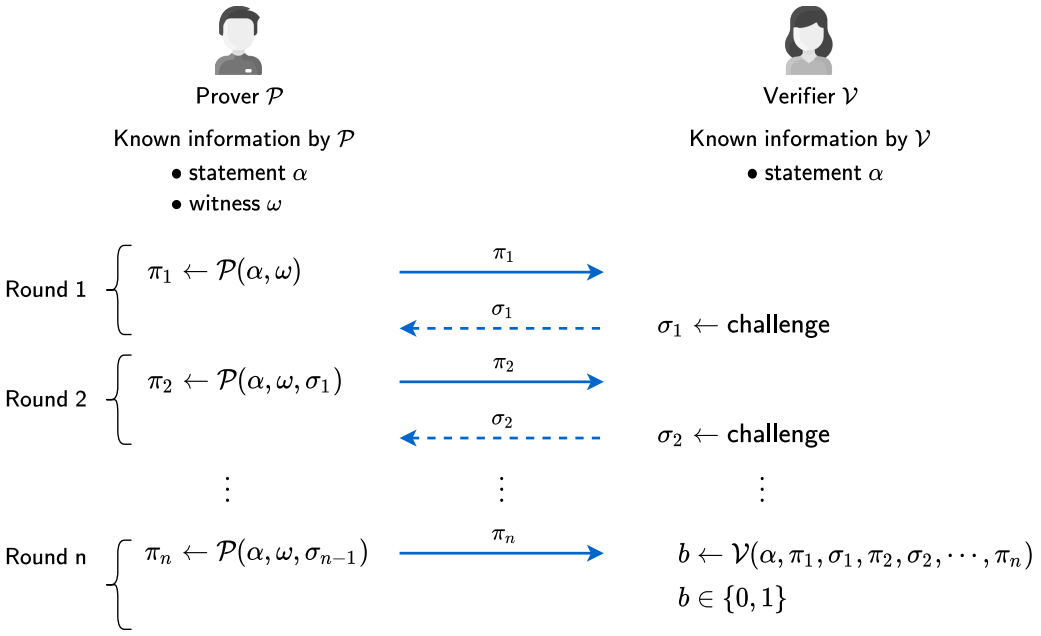
For feasible implementations, prover and verifier replace RO by some cryptographic hash function — that is, the Fiat-Shamir heuristic replaces the verifier's response by the outcome of the cryptographically secure hash function. In this transformation, it is assumed that the prover cannot predict the hash function's outcome — thus, the hash function acts as a random oracle. Therefore, it is said the proof of the NIZKP holds in the random oracle model (FRANCO, 2014, p.225). The advantages and limitations of the RO model are not part of the scope of this work.

However, many works in the literature have addressed these and other questions about the RO model, such as Wee (2009), Bootle *et al.* (2016), Kalai *et al.* (2017), Iovino and Visconti (2019).

Overall, NIZKP systems coexisted in two types: (1) inefficient NIZKPs, which are secure under standard assumptions in the CRS model, and (2) practically efficient NIZKPs built from the Fiat-Shamir heuristic, which are secure on RO model — hence only heuristically secure in the standard model (CHAIDOS; COUTEAU, 2018). Considering this statement by Chaidos and Couteau (2018), and given the ease of implementation and computing efficiency, in this thesis, the transformation from an interactive ZKP to a non-interactive ZKP is performed by applying the Fiat-Shamir heuristic.

To show the operation of a ZKP in n rounds more clearly, consider the generic interactive zero-knowledge proof protocol represented by the Figure 11 between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Intuitively, an interactive protocol is a pair of algorithms that send messages to each other until one of the algorithms ends. The common entry for prover $\mathcal{P}$ and verifier $\mathcal{V}$ is the statement α; and the secret entry for $\mathcal{P}$ is ω. Note that there is no secret entry for $\mathcal{V}$ because we are considering a publicly verifiable protocol, as mentioned in the previous section. Thus, as shown in the Figure 11, $\mathcal{P}$ and $\mathcal{V}$ perform the following actions in n rounds. The verifier accepts or rejects the proof with a very high probability, as discussed in Section 3.8.1.

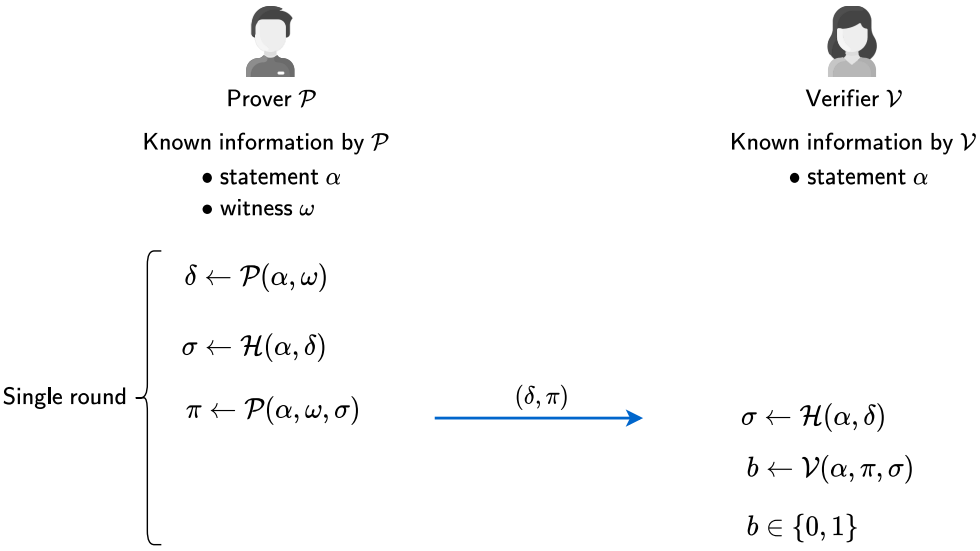
Figure 11 – Generic interactive zero-knowledge proof protocol.



Source: elaborated by the author.

In order to transform ZKP to NIZKP, the fundamental requirement is to include a cryptographic hash function $\mathcal{H}$ in the protocol. The prover calculates his messages as he would in the interactive protocol but the challenge σ, instead of chosen by $\mathcal{V}$, is replaced by a hash value. Thus, as shown in the Figure 12, $\mathcal{P}$ and $\mathcal{V}$ perform the following actions in single round.

Figure 12 – Transformation from ZKP to NIZKP using the Fiat-Shamir heuristic.



Source: elaborated by the author.

Method 3 — TRANSFORMATION FROM ZKP TO NIZKP USING THE FIAT-SHAMIR HEURISTIC

1. $\mathcal{P}$ calculates the committed message δ from the statement α and the witness ω.
2. $\mathcal{P}$ calculates the challenge σ using a hash function $\mathcal{H}$, which takes as input α and δ.
3. $\mathcal{P}$ calculates the proof π using α, ω, and σ, then sends pi and δ to $\mathcal{V}$.
4. $\mathcal{V}$ calculates σ, just like $\mathcal{P}$, using $\mathcal{H}$, which takes as input α and δ.
5. $\mathcal{V}$ checks the proof and decides whether to accept or reject it.

3.9 The Schnorr protocol

There are many ways to build zero-knowledge proof systems using a computationally intractable problem as its base mathematical problem. In 1991, Schnorr presented a zero-knowledge proof protocol based on the traditional discrete logarithm problem, known as Schnorr protocol (SCHNORR, 1991).

To use the Schnorr protocol as a ZKP system, we assume that the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ agree to use a finite field $\mathbb{F}_p$ of prime order p with generator g. $\mathcal{P}$ knows a number x, which is the discrete logarithm of a committed value z — that is:

$$z = g^x \bmod p.$$

Suppose that $\mathcal{P}$ wishes to convince $\mathcal{V}$ of this fact. A ZKP system using the Schnorr protocol is formulated as follows.

Method 4 — Schnorr Protocol Based on Traditional DLP

1. $\mathcal{P}$ randomly selects $r \in \{0, \cdots, p-1\}$, computes $t = g^r \bmod p$, and sends t to $\mathcal{V}$.
2. $\mathcal{V}$ selects and sends a random value $e \in \{0, \cdots, 2^l - 1\}$ to $\mathcal{P}$, where l is the bit length of the challenge.
3. $\mathcal{P}$ computes $u = r + x \cdot e \bmod p$ and sends u to $\mathcal{V}$.
4. $\mathcal{V}$ checks if the following equality holds: $t = g^u \cdot z^e \bmod p$. The verification succeeds only if the equality holds.

Note that x can be considered the private key of $\mathcal{P}$, chosen uniformly at random from $[0, p-1]$, and z is the public key of $\mathcal{P}$ and must be published. The value z must be an element in the field $\mathbb{F}_p$, which anyone can check.

To implement the NIZKP system used in this work, we adopted a variation of the Schnorr protocol, presented in Protocol 5. The variation consists of replacing the traditional discrete logarithm problem (see Method 4) with the discrete logarithm problem over elliptic curves. Furthermore, we are especially interested in the non-interactive form of the Schnorr protocol over elliptic curves, as presented by Hao (2017). The fundamental difference between the interactive and non-interactive versions is in the generation of the challenge. Instead of the challenge being produced by the verifier, it is produced through the Fiat-Shamir transformation.

To use the Schnorr protocol for NIZKP system based on ECDLP, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ must agree with an elliptic curve E over a field $\mathbb{F}_p$ of order p, a generator point $G \in E(\mathbb{F}_p)$. $\mathcal{P}$ and $\mathcal{V}$ know a point $Q \in \mathbb{F}_p$, which is the public key of $\mathcal{P}$, such that $Q = \kappa G$, where κ is the private key of $\mathcal{P}$. The prover wants to prove that know κ — the discrete elliptic logarithm of Q with respect to G — but $\mathcal{P}$ do not want revealing κ. Note that, as discussed in the

Section 3.5, it is computationally hard to find $\kappa \in \mathbb{F}_p$ such that $Q = \kappa G$. The Schnorr protocol based on ECDLP is described below.

Method 5 — Schnorr Protocol for NIZKP Based on ECDLP

1. The prover $\mathcal{P}$ chooses an integer $\upsilon \in \mathbb{F}_p$ at random and then calculates the point $A = \upsilon G$.
2. $\mathcal{P}$ calculates the challenge σ using a cryptographic hash function $\mathcal{H}$, such as $\sigma = \mathcal{H}(G||Q||A)$.
3. $\mathcal{P}$ calculates the response π to the challenge σ, such that $\pi = \upsilon + \sigma \cdot \kappa \ (mod\ p)$.
4. $\mathcal{P}$ sends a message carrying the point A, the response π.
5. $\mathcal{V}$ calculates $P = \pi G - \sigma Q = (\upsilon + \sigma\kappa)G - \sigma Q = \upsilon G + \sigma\kappa G - \sigma\kappa G = \upsilon G$.
6. $\mathcal{V}$ checks if $P = A$. If yes, the proof is valid; otherwise, the proof is invalid.

---

4 THE STRUCTURE OF BLOCKCHAIN AND HOW IT WORKS

Blockchain was initially proposed by Nakamoto (2008) as a technology underlying the Bitcoin cryptocurrency. For this reason, the word blockchain is usually associated with Bitcoin, but they do not mean the same thing, although they are directly related. More specifically, Bitcoin is a digital currency that functions as an electronic money system in a Peer-to-Peer (P2P) network and does not rely on a central trust entity. This type of digital money system, which uses virtual assets and uses encryption, is called cryptocurrency. On the other hand, blockchain is the technological platform behind the operation of the first and most popular cryptocurrency: Bitcoin.

Blockchain has been considered a disruptive technology since it digitally creates a decentralized trust entity, eliminating the need for a trusted central authority. It is possible to define the blockchain in several aspects. From a business point of view, Definition 31 is the more popular definition of blockchain — attributed to Tapscott and Tapscott (2016), authors of the book "*Blockchain Revolution*".

Definition 31 (***Blockchain***)**.** *The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions, but virtually everything of value (SHRIVASTAVA et al., 2020, p.219).*

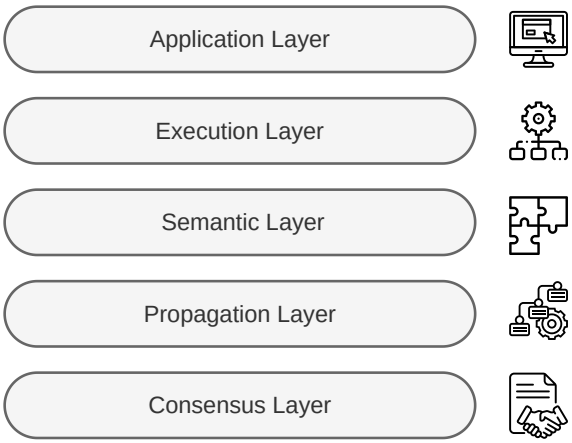
From a more technical point of view, we have the following definition.

Definition 32 (***Blockchain***)**.** *As a secure ledger, the blockchain organizes the growing list of transaction records into a hierarchically expanding chain of blocks, with each block guarded by cryptography techniques to enforce strong integrity of its transaction records. New blocks can only be committed into the global chain of blocks upon their successful competition of the decentralized consensus procedure (ZHANG et al., 2019).*

According to Singhal *et al.* (2018e, p.18) yet there are no agreed global standards that would clearly separate the blockchain components into distinct layers. On the other hand, there are different models presented in the literature to abstract blockchain in layers in order to facilitate understanding. For example, Gao *et al.* (2018) present a model in three layers: network layer, data layer, and application layer. Wang *et al.* (2019) represent the blockchain in four layers: application layer, virtual state machine layer, consensus layer, and data and network organization layer. In this work, we adopt a model based on the model presented by Singhal *et*

al. (2018e, p.18), which comprises five layers: application layer, execution layer, semantic layer, propagation layer, and consensus layer. Keep in mind that blockchain is never just a piece of technology, but a combination of business principles, economics, game theory, cryptography, and computer science engineering (SINGHAL *et al.*, 2018e, p.17).

Figure 13 – Blockchain layers.



Source: elaborated by the author based on Singhal *et al.* (2018e, p.18)

4.1 Application layer

The application layer allows for developing applications and provides an Application Programming Interface (API) for various scenarios and makes users interact with each other without thinking about the underlying technologies' details. It is responsible for providing services directly to customers.

Increasing the visibility of bitcoin and other cryptocurrencies has made research increased dramatically, transforming a diverse spectrum of applications. These can be subdivided into the categories of Internet of Things (IoT), Big Data, Cloud and Edge Computing, Identity Management, Cryptocurrency, Economics and Markets, Business Solutions, Smart Contracts and Automation, Traceability in Supply Chains, Medical Informatics, Communication and Networking, and others (GAO *et al.*, 2018).

Therefore, there are many possible applications on a blockchain network, but the details of implementing these various applications are outside the scope of this work. However, details on the construction of DApps can be seen in Singhal *et al.* (2018a, p.319).

4.2 Execution layer

The execution layer is where instructions coming from the application layer are executed by the blockchain network's nodes. The instructions executed can be simple, which are not Turing-complete as in the Bitcoin network. Bitcoin blockchain [1] allows only a few sets of instructions. On the other hand, Ethereum [2] and Hyperledger [3] networks can execute more complex instructions — executing smart contracts (see Section 4.2.1).

The blockchain network is structured like a P2P network on top of the Internet. It consists of several nodes that use protocols to perform the transaction validation and transmission. Each node joins the network for the chance to earn some reward — usually cryptocurrencies. A node can perform four functions: (1) routing, (2) database storage, (3) mining, and (4) running the client software (JESUS *et al.*, 2018). The node that includes all four functions is called the *full node* and contains a complete record of all transactions already registered on the blockchain. However, a typical user who only seeks to create a transaction in the network needs only to run the client software and routing. That way, he/she can connect to the network and make transactions using any computing device, even a smartphone, without the need to store the entire blockchain.

4.2.1 Smart contracts

A smart contract is a set of functions stored on the blockchain in the form of a script that all nodes can execute. The smart contract runs automatically if a predefined event occurs or can be activated by addressing a transaction to it. Like blockchain users, a smart contract is identified by an address (see Section 4.3.1). However, unlike a user, each smart contract has a unique address. The execution of the smart contract can result in a set of new transactions.

Once recorded on the blockchain, the smart contract is immutable. In other words, not even the contract creator can modify its code or subvert its execution. Like a transaction, a smart contract is replicated to all full nodes on the network so that each one can verify the contract's correct execution.

In the Ethereum network, smart contracts are written, most commonly, using the high-level solidity language, which is continuously evolving, adding more and more features.

[1] https://bitcoin.org
[2] https://ethereum.org
[3] https://www.hyperledger.org

Smart contracts are compiled in bytecodes, which are executed on the Ethereum Virtual Machines (EVM). The Hyperledger network supports the execution of smart contracts in machine codes that are compiled into images from the docker. It supports Java and other high-level languages. Singhal *et al.* (2018d, p.253) summarize the main features of Ethereum smart contracts below:

- Smart contracts reside inside the Ethereum blockchain;
- They have their own account, hence address and balance;
- They are capable of sending messages and receiving transactions;
- They get activated when they receive a transaction and can also be deactivated;
- Like other transactions, an execution fee and storage fee are applicable for them.

4.2.2 Types of nodes

In general, the device that connects to the blockchain network obtains a copy of it automatically. However, from a data structure point of view, blockchain is a growing list of data records, so not every node has the capacity to store the entire chain of blocks. Thus, to diversify nodes in the network, blockchain basically supports two types of nodes: full nodes and light nodes.

Full nodes. Once connected to the network, they receive updates on new transaction blocks, which they check and incorporate into their local blockchain. In this way, full nodes always maintain a complete and updated copy of the blockchain. As a practical result of this, the information stored on a blockchain cannot be lost or destroyed because that would mean having to destroy all the full nodes on the network.

Light nodes. They only store the block headers. In this case, with the hash of the previous block and the hash of the Merkle root (see Section 4.6.1), light nodes can verify some transactions without compromising their storage capacity with a full copy of the blockchain. A light node stores the user's wallet; however, it relies on third-party servers to access transactions. As light node does not store copies of all transactions, they must rely on third-party servers to validate other transactions. Also, they cannot mine new blocks.

4.3 Semantic layer

The blockchain uses a digital ledger to record all transactions shared between all users publicly. The ledger is not stored on a server of a supposedly trusted entity, such as a bank.

Instead, it is distributed over a broad worldwide network made up of particular devices that store data and perform computations. The ledger consists of blocks, and each block contains a part of the transactions and smart contracts generated on the network.

A transaction, whose set of instructions are performed at the execution layer, must be validated by the semantics layer. It is a logical layer since it imposes a process to check whether the transaction is valid or invalid. A typical example of this verification — in the case of the Bitcoin blockchain — is when someone is spending bitcoins: it needs to find out if this is a legitimate transaction or a double-spending attack. Therefore, in this layer are defined system rules, storage mode, and data structures, such as the Merkle tree. Furthermore, this is where a crucial feature of blockchain is defined: the way the blocks connect. Each block contains the hash of the previous block, up to the chain's first block — called genesis —, resulting in a logical chain of block.

4.3.1 Transaction

The transaction is the basic unit of information on the blockchain. A transaction records a process of data exchange between network participants. The transaction does not necessarily represent money but rather facts. In this context, a fact can have several meanings, such as a cash transfer, a property transfer, or a medical record. When a node wants to add a new fact to the ledger, a consensus is required between the network nodes to decide whether it can be registered or not. If there is a consensus, the fact will be written and cannot be deleted or changed (see Section 4.5).

A typical transaction involves the sender's digital signature, the recipient's address, and the transaction's content. However, the data structure and content of a transaction are defined according to the application's purpose. The network identifies users by large sequences of letters and numbers, called *addresses*, generated from their public key. To make transactions, the user needs a blockchain client software called a *wallet*, which allows storing and exchange currencies and performs other transactions. The wallet is also responsible for managing users' addresses and storing the corresponding public/private key pair.

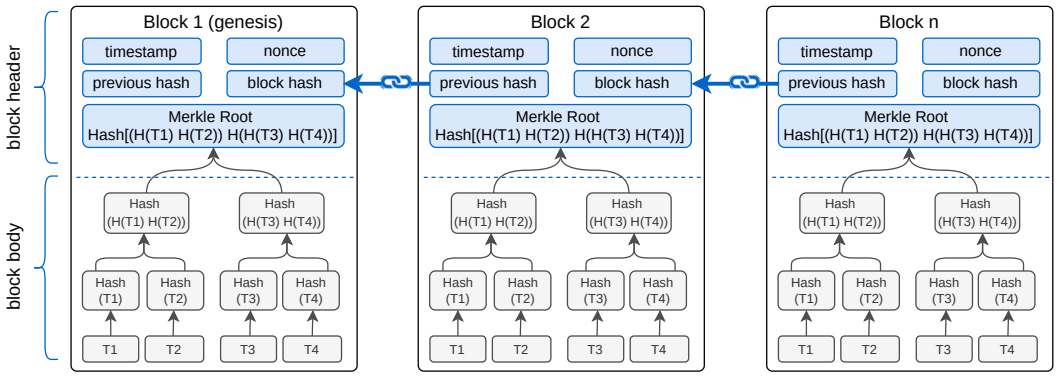
As an example, suppose that Alice wants to send a transaction to Bob. For this, she needs to transmit, from her wallet, a signed transaction with her private key. This action generates a digital signature, which is used to verify the transaction's origin and authenticity by blockchain nodes. Note that, again, the fundamental assumption is that only the private key

owner can use it. Then, each network's node, not only Bob, can verify that the transaction comes from Alice, decrypting the transaction with her public key.

4.3.2 Block

The blockchain organizes transactions in blocks. The structure of the blocks is fixed and has specific fields with their corresponding mandatory data. A block consists of two main parts: the header and the body, as shown in Figure 14.

Figure 14 – Simplified chain of blocks structure.



Source: elaborated by the author.

The header of a block can consist of many fields. In general, the fields that appear in the block header can be seen in Table 4, especially in the blockchain that adopts Proof of Work (PoW) (see Section 4.5.1). The block body is made up of transactions and a transaction counter. The maximum amount of transactions in a block relies on the block's size and the size of each transaction.

4.4 Propagation layer

The propagation layer is the P2P communication layer that allows the nodes to discover each other, communicate and sync with each other concerning the network's current state (SINGHAL *et al.*, 2018e, p.22). Since there is no central server to respond to requests, when a node enters the network for the first time, it does not know any full nodes' IP address. However, they are equipped with some mechanisms to find the pairs, for example, *DNS seeds*. In this mechanism, several hostnames are kept in the DNS system and resolved to the full nodes' IP addresses.

Table 4 – Typical block header components.

Field	Size	Description
Previous Block Hash	32 bytes	It contains the hash of the block header of the previous block in the chain. When all the previous block header fields are combined and hashed with SHA256 algorithm, it produces a 256-bit result.
Merkle Root	32 bytes	The hashes of transactions in a block form a Merkle tree. Thus, this field contains the hash of the tree's root. If a transaction is modified in the block, it will no longer correspond to the Merkle root when computed. In this way, it guarantees that maintaining the previous block header's hash is enough to keep the blockchain protected. Also, the Merkle trees help determine whether a transaction is part of the block in time $O(n)$, which is quite fast (see Section 4.6.1).
Timestamp	4 bytes	There is no notion of a global time in the network. Therefore, this field indicates an approximate time of block creation in Unix time format. That is, it is the current time in seconds since 01/01/1970.
Difficulty Target	4 bytes	PoW difficulty level that was set for this block when it was mined (see Section 4.5.1).
Nonce	4 bytes	This is the random number that satisfied the PoW puzzle during mining.

Source: Singhal *et al.* (2018b, 162).

As for the transmission protocols for new blocks, according to Li *et al.* (2017b), the propagation mechanisms can be of the following types.

Advertisement-based propagation. This propagation mechanism originated from Bitcoin and can be summarized as follows. When a node *A* receives information about a new block, *A* sends a *inv* message (a message type in Bitcoin) carrying some preliminary data about the block to its connected peers. If a node *B* that receives the message already has the block information, it will do nothing. If node *B* does not have the information, it will respond to the *A* node; then, node *A* sends the complete information for that block to the node *B*.

Sendheaders propagation. It is an improvement over the previous mechanism. Here, the node *B* starts by sending a *sendheaders* message — a message type in Bitcoin— to the node *A*. When the node *A* receives data about the block, it will directly send the block header information to node B. Compared to the advertisement-based propagation mechanism, the *A* node does not need to send a message carrying preliminary data about the block and, therefore, accelerates the block's propagation.

Unsolicited push propagation. In this mechanism, after a block is mined, the miner will directly transmit the block to other nodes. There are no *inv* or *sendheaders* messages here. In this case, this method can further improve the speed of propagation of the block.

Relay network propagation. This propagation mechanism is an improvement to the unsolicited push mechanism. Here, all miners share a pool of transactions. Each transaction is

replaced by a global ID, which will significantly reduce the transmission block's size, further reducing the network overhead and improving propagation speed.

In general, nodes are designed to immediately transfer a transaction or block to nodes to which they are directly connected as soon as they know of a new transaction/block.

4.5 Consensus layer

A central issue in the blockchain is: who can propose a new block on the network? It is trivial that to ensure that all network's full nodes have the same chain of blocks, only one node must propose one block at a time, and all other nodes must validate the block's transactions and add it to their blockchains if they agree that the transactions are valid.

The primary purpose of consensus layer is to get all the nodes to agree on one consistent state of the ledger. According to Singhal *et al.* (2018e, p.22), there could be different ways of achieving consensus among the nodes, depending on the use case. In Bitcoin or Ethereum, the consensus is achieved through proper incentive techniques called *mining* and use PoW consensus mechanism to select a node that can propose a block.

As the public blockchain is an open network formed by unknown nodes and without a central authority, it is necessary to have a mechanism that selects who has the right to propose a block at any given time. In other words, there must be a mechanism capable of maintaining consensus on the network and keeping it running by offering some incentive to nodes for the public blockchain to be self-sustaining.

These consensus mechanisms come from Game Theory. Their system should be designed such that the nodes get the most benefit if they play by the rules. One way to ensure the nodes behave honestly is to reward for honest behavior and punish for fraudulent activities (SINGHAL *et al.*, 2018c, p.131). Currently, the two main consensus mechanisms used in blockchains are Proof of Work (PoW) and Proof of Stake (PoS).

4.5.1 Proof of Work (PoW)

The Proof of Work consists of the user proving that he/she spent some time to find an answer that satisfies a computational challenge proposed by the protocol. These concepts applied as requirements for proof of work were initially proposed by Back (2002) and were later used in the Bitcoin blockchain. PoW is based on two principles.

- It must be difficult and laborious to produce, but not impossible. In this context, the term "difficult" should be understood as a solution that requires great computational power and a few minutes of calculation. Consequently, it is financially expensive due to the electricity spent during the computation of the solution.
- The verification of this solution must be quick and easy to be performed by the other nodes.

The process of generating a new block is called *mining* and the nodes that perform this process are called *miners*. Each miner node starts this process by organizing, based on *timestamp*, the transactions it has received. Then, the miner records the transactions in a block (a type of block sketch to be proposed), on which the PoW will be calculated.

PoW is calculated by applying a hash function, SHA-256, to the block header. Thus, by definition, the output of this function is a number between 0 and 2^{256}. The block header has, among other fields, a *nonce* that the miners must frequently modify in order to obtain a hash that satisfies the difficulty criterion. The level of difficulty defines a target value, also called the target of the difficulty (see Section 4.5.1). Thus, the systematic change of nonce by the miner during the PoW calculation aims to find a partial hash collision; that is, the miner must find a hash value that is equal to or less than the target value of that block. For example, when the difficulty is set to 1 bit zero, a valid solution is any hash that starts with 1 zero followed by any value for the other $n - 1$ bits, where n is the size of the function's output hash. When set to 2 bits, the hash to be found must start with two zeros followed by any value for the other $n - 2$ bits, and so on.

Note that the difficulty of finding a hash that satisfies the challenge increases as the number of zeros increases in the initial part of the hash and decreases the space of possibilities. Generally speaking, if the difficulty is set to d bits, then there are 2^{n-d} possible hashes that satisfy the challenge. Therefore, finding a hash that satisfies this partial collision relies essentially on the miner's computational power, which we can call hash power (see about collisions in hash functions in Section 3.8). When a miner finds a valid solution, it propagates the block to the other nodes, and everyone must mutually confirm the validity of the hash found. If the block is validated, we say that the block has been mined, and the other nodes attach this new block to their own locally stored blockchains.

The developers of each blockchain project define the rate at which new blocks are added to the chain. The mining process takes place independently and simultaneously by each of the full nodes. Note that all the network's miners compete with each other to find a valid

solution. The first one they find has the right to propose the new block.

As mentioned earlier, when transactions are ordered and written to a new block, the block header's hash needs to be calculated to be less than the target of the difficulty. This work difficulty must be adjustable so that there is control over how quickly the blocks can be mined. In this way, the target of the difficulty can be calculated from the bits present in the difficulty field in the block header, using a pre-defined formula that each node has by default:

$$\texttt{target} = \texttt{coefficient} \cdot 2^{(8(\texttt{exponent}-3))}.$$

To better explain this formula, Singhal *et al.* (2018b, p.166) presents the following example. If the difficulty bits (four bytes in hexadecimal) are `0x1b0404cb`, then the first two digits, `0x1b`, form the `exponent`. The rest is coefficient, `0x0404cb`. Solving the equation with these values:

$$\texttt{target} = \texttt{0x0404cb} \cdot 2^{(\texttt{0x08}(\texttt{0x1b}-\texttt{0x03}))}$$

$$\texttt{target} = 4,\texttt{04CB} \cdot 10^{52}$$

$$\texttt{target} = \texttt{0x0000000000404CB000000000000000000000000000000000000000000000000}.$$

Therefore, the mining process is, essentially, to increment *nonce*, starting at zero, until finding a hash that is less than `target`.

To show the process of calculating the difficulty target, let us take the Bitcoin blockchain as an example, but without losing generality. In Bitcoin, the difficulty is projected to increase or decrease every 2016 block, which takes around two weeks. This means that a block must be generated, on average, every 10 minutes. As the system is decentralized, every 2016 blocks, all nodes individually calculate the new difficulty target, as follows:

$$\texttt{new}_{\texttt{target}} = \texttt{old}_{\texttt{target}} \cdot (\texttt{T}/2\ \texttt{weeks}).$$

Where `T` is the time spent to generate the 2016 blocks. Note that `T` can be found by comparing the *timestamp* that two blocks, separated by 2016 positions, were generated. If

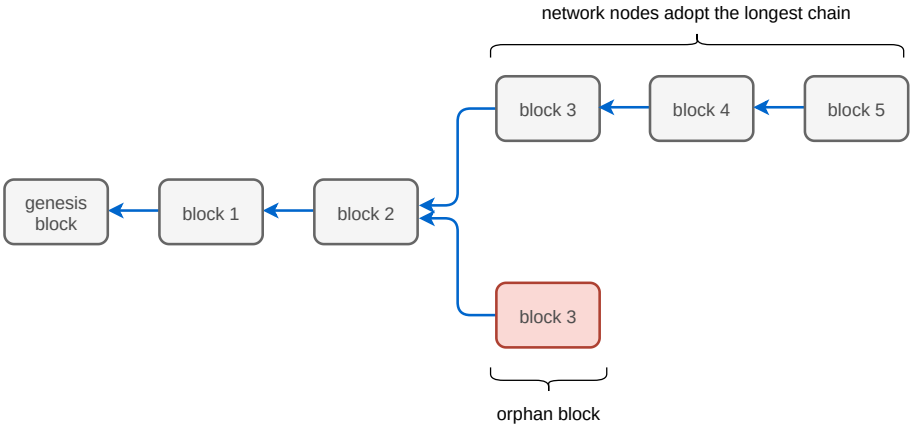
$$\texttt{new}_{\texttt{target}} < \texttt{old}_{\texttt{target}},$$

then it is because $\texttt{T} < 2\ \texttt{weeks}$. This means that the new target must be reduced, increasing the difficulty level, consequently taking more time to mine a block. Otherwise, the target will be increased in order to decrease the difficulty level. This difficulty adjustment aims to maintain an average of 10 minutes between blocks. This is part of the Bitcoin rules and is encoded on

each node. Note that in the PoW consensus, a miner's chances of proposing a block depend on how much hash power it has, taking as a comparison parameter the global hash power of all the miners together.

As the mining process happens simultaneously and independently by each miner, there may be a situation in which two or more nodes generate new blocks at almost the same time, causing a fork, as shown in Figure 15.

Figure 15 – Blockchain branches — the longer branch must be admitted as the main chain while the shorter one would be deserted.



Source: elaborated by the author.

When two blocks are proposed simultaneously, and both are valid, the consensus mechanism needs to decide which one will be part of the authentic chain and which one will be orphaned. This problem is solved as follows. By definition, each block has only one parent, so each miner, who is mining a new block, must choose which block to adopt as its block's parent. If one part of the nodes adopt a block and another part adopt the other, these two chains will coexist until one becomes longer than the other. In this way, nodes that behave honestly — following the consensus mechanism's rules — will always adopt the largest chain, and the fork will be resolved. Blocks that have been orphaned will have their transactions considered pending and should be included in the next blocks.

4.5.2 Proof of Stake (PoS)

Proof of Stake (KING; NADAL, 2012) is an alternative to PoW, which does not require a lot of computational power and high energy expenditure. Instead of solving a computational challenge, the creator of the next block is chosen in a probabilistic way. The chance of a

node being chosen is proportional to the balance of cryptocurrencies it has. This means that the more cryptocurrencies owned by a user, the more mining power he/she has. In PoS, it can also apply the term “validate” instead of “mining”; thus, the blocks are validated rather than mined. Note that both PoW and PoS effectively require a blockchain that supports cryptocurrency. In PoW, to encourage the miners to run its computing power to hash calculations needed to mine a block. In PoS, to make miners have their cryptocurrency at stake when mining (CHRISTIDIS; DEVETSIKIOTIS, 2016).

In this type of mechanism, any user who has cryptocurrencies can require to be a validator. For this, the node sends a special transaction, which blocks a part of the balance of its cryptocurrencies, as a type of deposit. Thus, the algorithm pseudo-randomly selects a validator in each time interval (usually around 10 to 14 seconds) and assigns that node the right to create a new block, pointing out the chain’s last block.

As the bets are public, each user can predict, with reasonable precision, which node will win the right to validate a block. In other words, the higher the bet, the greater the chance of being selected. Compared to PoW, PoS saves more energy, is more effective, and the cost of “mining” is almost zero. A more detailed description and a more extensive evaluation of the PoW and PoS mechanisms can be found in Tschorsch and Scheuermann (2016).

4.6 Building the chain of blocks

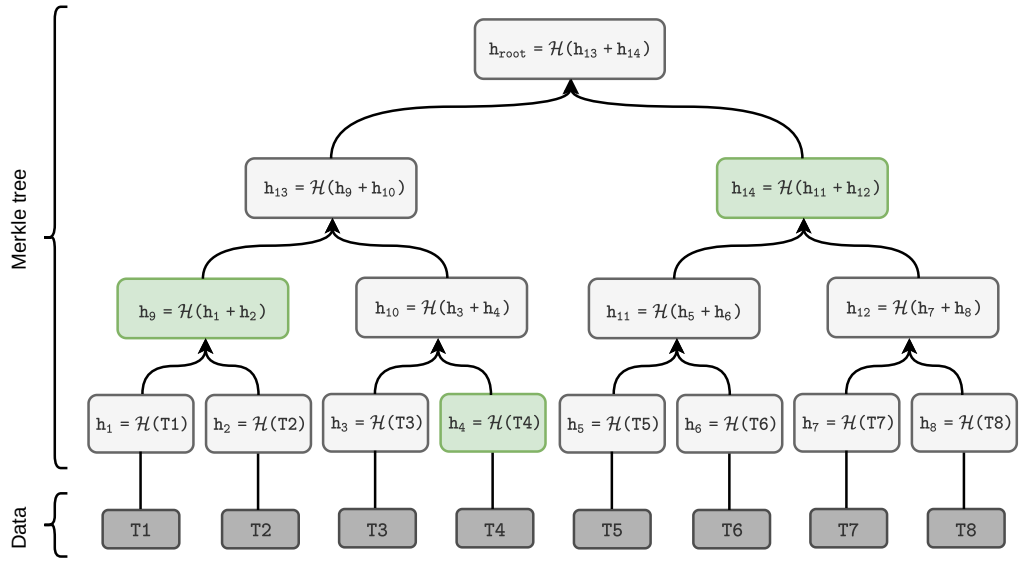
A blockchain can be seen as a global ledger shared between all nodes in a P2P network and used to store all transactions. This ledger consists of blocks, and each block stores a certain number of transactions. Every time a new block is created by the network, that block is attached to the global ledger. The new block always records the hash value of the previous block. Thus, the ledger is essentially formed by a chain of blocks: from the initial block, called genesis, to the most recent block, linked by hash pointers.

4.6.1 Merkle tree

Each block includes a summary of all transactions belonging to the block through a Merkle tree. A *Merkle tree*, proposed by Merkle (1988), is a data structure in which the value of a *leaf node* is assigned by calculating the hash of the data to be represented, and the value of an *internal node* is assigned based on its children’s hash value, as shown in Figure 16. This structure

is capable of efficiently verify the integrity of large data sets since it produces an identification that unequivocally represents the entire set of transactions in that block.

Figure 16 – Merkle tree built from eight messages and with Markle path highlighted in green.



Source: elaborated by the author.

The Merkle tree is built from the bottom up from pairs of nodes, containing their respective hashes, recursively until there is only one hash, called the Merkle root. At the last level of the tree are leaf nodes with hash pointers to the data; in the case of blockchain, these are the transactions themselves. Note that transactions are not stored in the tree. Instead, a hash is calculated for each transaction, and the corresponding hash is stored in each tree's leaf node. Then, at each level above the leaves, i.e., in the parent nodes, the hash pointers are formed from their children's hash, combining them two by two [4], until the tree's root is reached.

For example, to build the parent node $\mathbf{h_9}$ in Figure 16, the two 32-byte hashes of the children $\mathbf{h_1}$ and $\mathbf{h_2}$ are concatenated to create a 64-byte string, which is passed as input to the hash function $\mathcal{H}$. This process produces the hash of the parent node, such as: $\mathtt{h}_9 = \mathcal{H}(\mathtt{h}_1 + \mathtt{h}_2)$. This hashing procedure applied recursively results in the hash of the tree root, which summarizes all transactions, and is stored securely within the block header, along with other information presented in the Table 4. No matter how many transactions there are in the block, the Merkle root will always be the same size since the outcome of a hash function is always a fixed size. This makes this structure efficient in terms of saving space.

[4] Since the Merkle tree is a binary tree, it needs a number pair of leaf nodes. If there is an odd number of transactions to summarize, the last hash of the transaction will be doubled to create an even number of leaf nodes

A Merkle tree structure allows to build up a change-sensitive structure for efficient verification of large databases' contents — that is, as any change of data results in a change of its hash value of the corresponding nodes, and hence also in a change of the tree root. Trus, any modification of data can be detected quickly and safely, ensuring an efficient verification of integrity (BOLFING, 2020, p.91).

To verify if a specific transaction is included in the tree, it is necessary to check at most $2 * log_2(n)$ hashes, where n is the number of transactions (JESUS *et al.*, 2018). To prove that a transaction is included in the tree, it is enough to provide the block header and the path that the transaction will go through in the tree — this constitutes an authentication path called the *Merkle path*. This path is formed of the complementary nodes with the same height in the tree.

For example, suppose that a user wants to check if the transaction `T3` is in the block represented by the Figure 16. The user needs the block header and the hashes $\mathbf{h_4}$, $\mathbf{h_9}$, $\mathbf{h_{14}}$. In this case, Merkle path consists of four hashes — highlighted in green in the Figure 16. With this data, any node can calculate the Merkle tree root to prove that `T3` is included in the tree. This procedure can be done as follows:

- The node interested in verifying the transaction `T3` calculates the hash $\mathbf{h_3}$, which is concatenated with the hash $\mathbf{h_4}$ to form the input of the hash function $\mathcal{H}$ that outcome the hash $\mathbf{h_{10}}$.
- Now, the hash function $\mathcal{H}$ takes the $\mathbf{h_9}$ and $\mathbf{h_{10}}$ as input and produces $\mathbf{h_{13}}$.
- Then, the node can calculate the tree's root using $\mathbf{h_{13}}$ and $\mathbf{h_{14}}$ as input for $\mathcal{H}$ to obtain the hash $\mathbf{h_{root}}$.
- Finally, the verifier node compares the tree's root calculated by it with the root hash on the block header. If the hashes are the same, then the translation `T3` is in the block.

4.6.2 Ensuring immutability with hash pointers

In the blockchain, hash pointers are employed to create a chained block structure. Since the hash function used is resistant to collision (see Section 3.8), it is possible to use this technique to detect whether a block has been modified.

The blockchain guarantees the immutability of transactions by storing them in interconnected blocks through a hash. Since each block references the hash of the previous block, any change in a transaction modifies the Merkle root of the block and, consequently, changes

the block's hash as a whole. Note that, in general, the hash value of a block is calculated by applying the hash algorithm SHA256 to all fields of the block header concatenated in a single entry. Therefore, modifying the block hash results in breaking the chain of blocks and is easily identified by the network.

To better exemplify this situation, consider a blockchain attack scenario, where a dishonest node tries to modify a transaction. Before describing this scenario, consider the following characteristics of PoW:

1. by default, the nodes must consider only the most extensive chain as valid, the smaller chains will be disregarded (see Section 4.5.1);
2. an attempt to change the blockchain is made, initially, on the local node, only after that the malicious node tries to impose the fraudulent chain on the rest of the network.

Now suppose that a given transaction is in a block B and that there are already r blocks after it in the chain. For a successful attack, the attacker must produce an alternative chain larger than the legitimate chain. Note that honest nodes, including the one that mined the B block, are already mining the next block $r + 1$. Thus, to successfully modify a given block B, the attacker needs to recalculate the proof of work for the block B and all subsequent blocks to catch up and still overcome the work of honest nodes.

According to Nakamoto (2008), this race between honest and dishonest nodes can be modeled as a Binomial Random Walk. Thus, the success event is the honest chain being extended by one block, increasing its advantage by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by −1. Therefore, consider that the attacker's probability of catching up the legitimate chain, being r blocks behind, can be calculated by:

$$q_r = \begin{cases} 1, & \text{if } p \leq q \\ \left(\frac{q}{p}\right)^r, & \text{if } p > q \end{cases}$$

where

- p = probability an honest node finds the next block;
- q = probability the attacker finds the next block;
- q_r = probability the attacker will ever catch up from r blocks behind.

Note that trust in the blockchain relies on the percentage of dishonest nodes in the network. Considering that honest nodes' population must be greater than the population of dishonest nodes, $p > q$, the probability drops exponentially as the number of blocks that the attacker has to reach increases. This means that the blocks' immutability is protected due to the high computational cost necessary to modify a data set. Therefore, immutability could only be broken, in fact, if the computing power under the control of a single dishonest miner (or pool of miners) is greater than the rest of the network.

The scenario where a dishonest miner has the ability to attack the network, manipulate the blockchain, surpass all other nodes, and recalculate the recent block hashes is known as a 51 % attack. A comprehensive discussion about the 51% attack to be found in Eyal and Sirer (2014) and Bastiaan (2015).

4.7 Blockchain types and their characteristics

Blockchain networks can be classified into two major groups: *public blockchains* (permissionless blockchain) and *private blockchains* (permissioned blockchain). This classification is based on the data access way and how the nodes participate in the consensus mechanism.

In *Public Blockchains*, the network offers open access — permission is not required to join the network. The nodes are essentially anonymous, and the network allows inputs and outputs of the nodes at random. In this scenario, nodes compete to obtain the right to generate new blocks, and this generates mutual distrust among the nodes. All nodes can create transactions and participate in the consensus mechanism. According to Jesus *et al.* (2018), this creates so-called resistance to censorship, which means that no player can prevent a transaction from being added to the chain. Participants maintain the integrity of the chain by reaching a consensus on their status.

In *Private Blockchains*, the network offers access only to identified and authorized users. Not all nodes can participate in the consensus process; usually, when a new record is added, the integrity of the ledger is verified by a consensus process conducted by a limited number of trusted players. In this scenario, the network is typically maintained and used by an organization or group of organizations that have decided to share the ledger with each other. This type of blockchain is more suitable for corporate environments because they are in compliance with corporate rules. According to Zheng *et al.* (2017) a private blockchain is regarded as a centralized network since it is fully controlled by one organization.

Table 5 – Comparison between public and private blockchains.

Characteristics	**Public**	**Private**
Access	Read/write for anyone	Read/write for a single organization
Speed	Slower	Lighter and faster
Efficiency	Low	High
Security	Proof-of-work, proof-of-stake, and other consensus mechanisms	Pre-approved participants and voting/multi-party consensus
Immutability	Nearly impossible to tamper	Could be tampered
Consensus process	Permissionless and anonymous	Permissioned and known identities
Network	Decentralized	Centralized*
Asset	Native asset	Any asset

Source: Kim *et al.* (2019, p.11).

* According to Zheng *et al.* (2017), private blockchain is fully centralized as it is controlled by a single group.

The blockchain offers a set of properties that bring several benefits to the system's security and reliability. Below we highlight the main characteristics of a public blockchain, and a comparison between public and private blockchain can be seen in Table 5.

- *Decentralization*. It is one of the main reasons for interest in blockchain by industry and academia. The applications are executed in a distributed way so that the network does not need a central node to validate the transactions.
- *Immutability*. The transactions recorded in the ledger are immutable. Once registered, they cannot be changed or repudiated unless a 51% attack is successful. The ledger can only be updated by adding new transactions. Changing any information unit on the blockchain implies using an enormous computational power to change each block's hash (see Section 4.6.2).
- *Availability and Integrity*. Transactions transmitted to the network are verified and recorded in blocks distributed throughout the network. Thus, the blockchain's availability is generally high because some offline nodes do not impede the other nodes' functioning. Also, any forgery will be easily detected since any change in a transaction changes the entire block's hash. This mechanism ensures that all nodes on the network have precisely the same ledger.
- *Auditable*. If someone wants to verify that a transaction has occurred, this verification is quick and efficient, as shown in the Section 4.6.1. Besides, the blockchain's source codes are often open; however, the audit does not mean that

it is possible to obtain the identity behind a transaction easily, since the published addresses do not reveal the participants' identity.

4.8 Ethereum blockchain

Ethereum platform was proposed as an evolution of the Bitcoin platform. Like Bitcoin, Ethereum is a public blockchain platform. However, Ethereum is not only focused on cryptocurrency; it includes the concept of smart contracts integrated into the platform. Unlike Bitcoin, Ethereum supported Turing-complete language so anyone could write smart contracts that could virtually do anything and everything from a programming perspective (SINGHAL *et al.*, 2018d, p.221).

The Ethereum platform offers a computational infrastructure with decentralized virtual machines called EVM, which execute smart contracts. The smart contracts can be written in a programming language called Solidity, which is a high-level language that tries to abstract out particularities of the blockchain. Programs written in Solidity are translated into an intermediate code (bytecode) that is executed by EVM. In this context, smart contracts are composed of a set of functions and are associated with addresses (or account).

4.8.1 Ethereum accounts

The Ethereum platform, unlike Bitcoins, keeps track of account balances. On the Bitcoin network, it only records all approved transactions; and to determine the balance of an account, it is necessary to check all transactions that have already occurred on the network related to a particular account. The output of a transaction that is not used (or spent) by the entry of another transaction remains Unspent Transaction Output (UTXO) until it is spent. The sum of all assigned UTXOs to an account determines the account balance. Therefore, there is no notion of the state of account in Bitcoin's design. Ethereum, on the other hand, is stateful, and its basic unit is the account. Each account has a state associated with it and also has a 20-byte (160 bits) address through which it gets identified and referenced (SINGHAL *et al.*, 2018d, p.228).

In the Ethereum platform, there are two types of accounts: *External Owned Account (EOA)* and *Contract Account*.

- *EOA* or simple accounts are controlled by the external users using their private key. It has a corresponding balance and can send a transaction to another address

or contract accounts. The transaction between two EOAs is usually to transfer any form of value. When EOA sends a transaction for contract accounts aims to trigger the execution of the contract code.

- *Contract account*. The contract account is controlled by the code stored in the account. This code associate with an address is the smart contract. Each contract account has a balance and storage space associated with theirs.

4.8.2 Ether and Gas

In the Ethereum platform, the miners generate Ether, a tradeable cryptocurrency, because of which the public blockchain network is self-sustainable. Any application that is running on Ethereum has to pay transaction fees that eventually the miners get for running the nodes and sustaining the whole network (SINGHAL *et al.*, 2018d, p.221).

When the contract code is triggered by a transaction, each miner node in the network checks and executes transactions at the EVM as part of the block validation protocol. Each operation on the EVM has a specific consumption, which is accounted for by Gas units. Gas can be purchased via Ether, and the sender of the transaction needs to pay for the Ether for the operations it wants to perform, i.e., computing or data storage (WANG *et al.*, 2018). The transaction cost is calculated as

$$Ether = Gas\ used * Gas\ price.$$

4.8.3 Design principles of Ethereum

According to Singhal *et al.* (2018d, p.223), Ethereum borrows many Bitcoin Core concepts since it has stood the test of time, but it is designed with a different philosophy. The development of Ethereum was carried out following certain principles.

- *Simplistic design*. Ethereum blockchain is designed to be as simple as possible so that it is easy to understand and developing decentralized applications. The implementation's complexities are kept to a bare minimum at the consensus level and are managed at a level above it.
- *Freedom of development*. Ethereum is designed to encourage any kind of decentralization on its blockchain platform and does not discriminate or favor any specific kinds of use cases.

- *No notion of features*. In an effort to make the system more generalized, Ethereum does not have built-in features for the developers to use. Instead, Ethereum provides support for Turing-complete language and lets the users develop their own features the way they want to.

There are different blockchain models to satisfy different types of applications. Table 6 presents the main blockchain examples and summarizes their fundamental features.

Table 6 – Example of public and private blockchains and comparison between them.

Characteristics	**Bitcoin**[a] (NAKAMOTO, 2008)	**Ethereum**[b] (WOOD *et al.*, 2014)	**Hyperledger**[c] (CACHIN, 2016)
Nature	Permissionless	Permissionless	Permissioned
Validation	PoW SHA-256	Ethash PoW	PBFT
Purpose	Cryptocurrency	Smart contract	Chaincode
Language	Stack based scripts	Internal code Turing complete (Solidity)	Go, Java
Block processing time	~ 600 s	~ 15 s	~ Real time

Source: Jesus *et al.* (2018).

[a] https://bitcoin.org
[b] https://ethereum.org
[c] https://www.hyperledger.org

5 ENSURING SECURITY AND PRIVACY-PRESERVING OF DATA IN MOBILE HEALTH SYSTEMS

Privacy and security in electronic health record systems, including mHealth systems, are among the most relevant issues. Mobile health devices are, typically, wearable with resources-limited. Thus, traditional security mechanisms may consume more resources than the device can offer; hence, these solutions cannot be used in some cases. To address these issues, we present an approach that combines blockchain, InterPlanetary File System (IPFS), Non-Interactive Zero-Knowledge Proof (NIZKP), and Attribute-Based Encryption (ABE) to ensure the privacy and security of patient data.

In Section 1.1, we emphasize the importance of authenticating monitoring devices on mHealth systems. In this chapter, we started our proposal by elaborating on an authentication mechanism based on NIZKP over Elliptic Curve Discrete Logarithm Problem (ECDLP). The elimination of interaction, typical of the Zero-Knowledge Proof (ZKP), leads to process optimization, especially in scenarios that only have devices with restricted resources. As we are concerned with security and privacy-preserving throughout the system, we have included the Ethereum blockchain, which — through smart contracts — assumes the role of a decentralized authenticator that guarantees access only to legitimate users of the system. Due to the public and distributed nature of the blockchain, there are deficiencies in the privacy requirements. Thus, we present a scheme in which the health data transmitted, stored, or shared are protected by ABE.

We started this chapter by discussing related works focusing on privacy-preserving in personal health systems, especially in the mHealth system. Next, we detail the construction of our proposals, expose the security protocols, the experimental results, and end by discussing the results achieved.

5.1 Related work

This section organizes related work into three groups. The first group gathers the works that only address the authentication issues between a monitoring device and the mobile gateway. The second group covers works that include the blockchain as a new tool to contribute to privacy-preserving in mHealth systems. The third group presents works that, although addressing issues of privacy and security, do not address these issues from end-to-end, leaving security or privacy flaws in some parts of the system.

5.1.1 Authentication between wearable devices and a mobile terminal

The communication method between the monitoring devices and the central node (smartphone) must provide strong security mechanisms to ensure that confidential patient data cannot be accessed by an attacker (BAKER *et al.*, 2017). Liu *et al.* (2016b) report that there are severe attacks on wearable devices since the communication channels are exposed. To counter these attacks, they proposed a lightweight authentication protocol between wearable devices and the smartphone using a challenge-response scheme. However, the proposed protocol is designed to authenticate two wearable devices simultaneously; furthermore, after local authentication, a cloud server needs to verify the two wearable devices' legitimacy to complete the authentication process. Liu *et al.* (2016b) provided a formal security analysis of the protocol; however, they did not conduct experiments that show the time spent or memory consumption of the scheme.

Das *et al.* (2018) proposed a lightweight authentication protocol between wearable devices and a mobile terminal. Once the mutual authentication is successful, data received by the mobile terminal can be uploaded to a cloud server. However, the authors did not address the issue of authentication among healthcare professionals, who wish to access data, and the cloud storage server. Similarly, Liu *et al.* (2016a) proposed an authentication scheme between wearable devices and a mobile terminal. However, unlike our proposal, the authors do not consider devices with limited resources. Their focus was on wearable devices, with considerable computational resources, able to generate and read QRCode as a part of the authentication process.

Le *et al.* (2011) proposed a mutual authentication and access control based on Elliptic Curve Cryptography. The objective is to authenticate biosensors and mobile terminals in a healthcare environment. Their proposal requires less computational overhead due to the use of ECC. However, it requires one or more trusted third parties (i.e., Key Distribution Center) to generate and control the key of the devices and users.

Huang *et al.* (2017) proposed an integrated PHI framework for privacy-preserving. The integrated PHR system collects patient data from multiple healthcare providers and stores it on a PHR Cloud Server to give the patient more control power. The proposed access control is based on ABE. However, their proposal requires all stakeholders to be registered with a trusted authority which generates and distributes users' keys. According to the authors, once a patient visits a healthcare provider, his/her related medical record is created and kept by that provider. Note that the healthcare providers collect data and create health records, and only then, PHR system collects the data from the records created by the providers. Therefore, nothing prevents

healthcare providers from keeping a copy of patient records. Thus, the authors' efforts to provide patient-centered access control, through a centralized PHR, do not guarantee the privacy of the patients.

5.1.2 Privacy-preserving using blockchain

There are many efforts to preserve patients' privacy; several contributions have proposed integrating blockchain technology with personal health records systems (ROEHRS *et al.*, 2017; YUE *et al.*, 2016; AZARIA *et al.*, 2016; SHEN *et al.*, 2019; DWIVEDI *et al.*, 2019). In this section, we also describe some of these contributions that propose blockchain-based access control for mHealth systems.

Genestier *et al.* (2017) presented a model in which patients manage access consent to their data in a decentralized way using blockchain. Although a smart contract performs access control, at least two entities are operating as centralized intermediaries between the patient's application and the blockchain: (1) a data management server; and (2) a consent management server. Upon data access request, the data management server consults the consent management server, which checks recorded authorizations in the blockchain. These servers are single points of failure that can impair system availability. Also, the authors do not appear to employ any data protection mechanism while the data is stored on the server. An attack on that server can expose the patient's sensitive data.

Liang *et al.* (2017) proposed a mobile healthcare system integrated with blockchain for sharing health data. Each data access request is sent to the blockchain, where it is processed to obtain permission from the data owner. However, the system's implementation relies on a trusted third party, which receives storage or data query requests. The system requires the user to register with a cloud storage service provider to synchronize data. Although the authors have developed a Merkle tree-based method to ensure data integrity, the data is stored without any cryptographic scheme that guarantees the data's confidentiality in an eventual attack on the server.

Silva *et al.* (2019) presented a cryptographic scheme to guarantee confidentiality, integrity, and authenticity of data in mHealth applications. The authors proposed a hybrid approach using symmetric and asymmetric encryption algorithms. However, RSA — the asymmetric algorithm employed by them — requires a very large key to provide an adequate level of security. For example, to achieve the same level of security as an elliptic curve cryptosystem

with a 256-bit key (used in our work), RSA needs a 3072-bit key (BAFANDEHKAR *et al.*, 2013; BOS *et al.*, 2009). Due to the size of the key and the time required for processing, algorithms based on modular arithmetic, such as RSA, may make the scheme proposed in Silva *et al.* (2019) unsuitable for mHealth systems with resource-limited devices.

Thwin and Vasupongayya (2019) proposed an access control model for personal health record systems. As in our approach, the authors store metadata corresponding to health records in the blockchain. Health records are stored encrypted on a cloud server. However, access control is performed using a Proxy Reencryption Scheme. This approach makes the data sharing process dependent on an intermediary, which is the proxy server responsible for re-encryption. Thus, the encryption keys and other information necessary for an authentication process are under the proxy server's control. The approach of Thwin and Vasupongayya (2019) suffers from the problem of single point of failure since it relies on a centralized third party to control part of the system operations. Dagher *et al.* (2018) proposed a framework that uses smart contracts in an Ethereum-based blockchain for access control. However, similar to the proposal of Thwin and Vasupongayya (2019), Dagher *et al.* (2018) also use proxy re-encryption technique, making the system dependent on a third party.

Li *et al.* (2019) proposed a fine-grained access control for mHealth systems, which uses multi-authority on ABE scheme. They argue that the multi-authority model in ABE has advantages over the single-authority model. However, there is a dependency on a trusted third party to generate and distribute the decryption keys on both models. Similar to the model in Li *et al.* (2019), Rahulamathavan *et al.* (2017) presented an approach to privacy-preserving in IoT ecosystems which employs ABE scheme with multi-authority integrated with blockchain. Our approach has a simpler architecture when compared to Li *et al.* (2019) and Rahulamathavan *et al.* (2017). In our approach, there is no dependency on third parties; furthermore, the patient himself, assisted by his smartphone, can generate and distribute ABE scheme's keys to the system's users.

Lunardi *et al.* (2018) proposed the architecture of a ledger-based access control scheme for IoT. Their focus is not on mHealth systems; however, it is related to our proposal since they investigate the use of Blockchain in the context of resources-limited IoT devices. The authors implement cryptographic algorithms on an Arduino device similar to the one we used in our experiments to evaluate our approach. They show that the Arduino was able to run the RSA and AES algorithms with an acceptable response time. However, it is not possible to evaluate the implemented algorithms' security level since the authors did not specify the size of the keys

used. Thus, we cannot make a direct comparison (of the security level and of the execution time) between the algorithms implemented in Lunardi *et al.* (2018) and the algorithms implemented in our work. Nevertheless, our experiments show that our security scheme for resource-limited devices has an acceptable response time and a high security level.

5.1.3 Limited solutions for privacy-preserving in mHealth

The scientific community has massively investigated the security and privacy in healthcare systems; indeed, several surveys have been published in recent years (BAKER *et al.*, 2017; VITHANWATTANA *et al.*, 2016; MUTLAG *et al.*, 2019; HATHALIYA; TANWAR, 2020; MCGHIN *et al.*, 2019b; HOUTAN *et al.*, 2020). However, not all existing contributions did address privacy and security issues holistically. In the context of healthcare systems assisted by wearable devices, several contributions (GENESTIER *et al.*, 2017; GIA *et al.*, 2017; AHMAD *et al.*, 2016; FARAHANI *et al.*, 2018; YANG *et al.*, 2017; ZHANG *et al.*, 2015b; SATHYA; KUMAR, 2017) did not address the authentication between these devices and the smartphone (or some type of a gateway). They assume that data that arrives at the smartphone is intact and is sent by legitimate devices. However, this assumption does not hold in most cases (NAVEED *et al.*, 2014; LIU *et al.*, 2016b). In this work, we propose a holistic solution for mHealth systems. We protect data from collection to storage/sharing. We are concerned with providing secure interactions between the smartphone, controlled by the user, and wearable devices with limited resources. Our proposal creates an exclusive association between the wearable device and the official mHealth application to solve the problems presented by Naveed *et al.* (2014) and presented in Section 1.1.

Generally, patients are very concerned about the privacy of their data that is taken care by third-party cloud providers (VORA *et al.*, 2018). In contrast to existing contributions (GENESTIER *et al.*, 2017; LIANG *et al.*, 2017; SILVA *et al.*, 2019; THWIN; VASUPONGAYYA, 2019; DAGHER *et al.*, 2018; LI *et al.*, 2019; RAHULAMATHAVAN *et al.*, 2017), our proposal does not rely on trusted third parties or intermediate servers. Instead of storing patient health data on centralized servers, we integrate the mHealth architecture with blockchain/IPFS to maintain a distributed database where data can be managed exclusively by the patient. Thus, our approach minimizes the risk of DDoS attacks, does not suffer from the problem of single point of failure, and guarantees availability. By comparing our approach, for example to the approach in Huang *et al.* (2017), we eliminate healthcare providers' power to control patient

data. Healthcare providers do not maintain the patient data, and therefore cannot share it with third parties; they are limited to analyzing the data.

Table 7 summarizes the security and privacy characteristics of the works described in this section and allows us to compare them with our proposal.

Table 7 – Comparison of the characteristics of security and privacy of several works.

Approach	**Characteristics**					
	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**
Liu *et al.* (2016b)	yes	yes	*	yes	yes	no
Das *et al.* (2018)	yes	no	no	yes	yes	yes
Liu *et al.* (2016a)	yes	no	*	no	no	no
Le *et al.* (2011)	yes	no	yes	yes	yes	yes
Huang *et al.* (2017)	no	no	yes	yes	yes	no
Genestier *et al.* (2017)	no	no	yes	yes	yes	no
Liang *et al.* (2017)	no	no	yes	yes	yes	yes
Silva *et al.* (2019)	no	no	yes	yes	yes	no
Thwin and Vasupongayya (2019)	no	no	yes	yes	yes	no
Dagher *et al.* (2018)	no	no	yes	yes	yes	no
Li *et al.* (2019)	no	no	yes	yes	yes	yes
Rahulamathavan *et al.* (2017)	no	no	yes	yes	no	yes
Huang *et al.* (2017)	no	no	yes	yes	yes	no
Our approach	yes	yes	yes	no	no	yes

1. Own authentication mechanism of the monitoring device on the smartphone (or some type of a gateway).
2. Exclusive association between the monitoring device and the official mHealth application.
3. Authentication of data users.
4. Depends on third parties.
5. Centralized architecture.
6. Suitable for devices with limited resources.

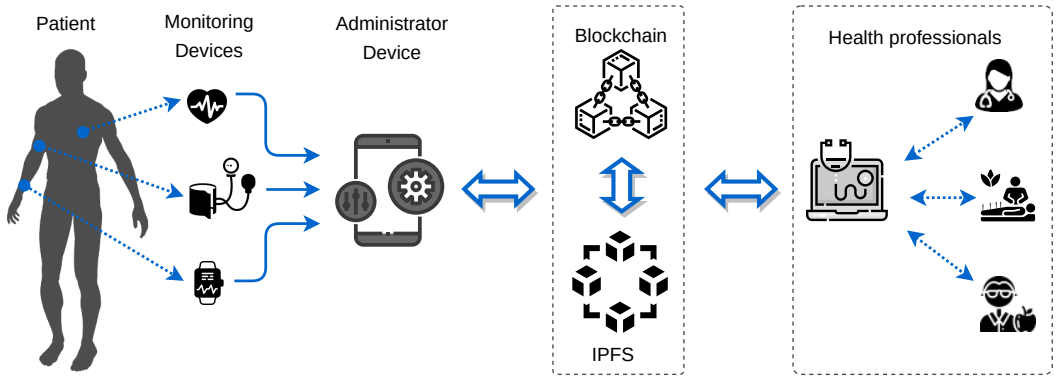
* means that it is a feature that does not apply to the approach.

Source: elaborated by the author

5.2 Model overview

The model proposed in this work has a distributed architecture that integrates mHealth technology with blockchain technology to preserve patient privacy. Our model considers six players as the entities of the system, as shown in Figure 17.

Figure 17 – Overview of the mHealth system integrated with the blockchain.



Source: elaborated by the author.

1. *Patient*. The data owner. He/she is responsible for administering the system and may grant or deny access to healthcare professionals.
2. *Monitoring devices*. They are miniaturized devices equipped with biosensors, microcontrollers, and wireless data transmission. These devices can be incorporated into clothing or worn on the body as accessories. They can capture the patient's physiological signals, such as blood pressure, blood sugar rate, heart rate, sleep conditions, breathing patterns, among others. The collected data will be sent to a storage service (which, in our case, is the blockchain/IPFS), where it is available for analysis by authorized healthcare professionals.
3. *Administrator device*. Due to limited processing and memory resources, monitoring devices must transmit the collected data to a more robust processing device. The patient may use their smartphone as a trusted device to configure and administer the system. The administrator device is equipped with an application capable of receiving, formatting, and encrypting the data before sending it for storage/sharing.
4. *Blockchain*. Blockchain is an append-only, shared, fault-tolerant, and distributed database which maintains a set of records in the form of blocks. The blocks are transparent and are accessible by every blockchain node; however, they cannot be modified or deleted (HASSAN *et al.*, 2019). The blockchain network has three functions in the proposed system:

   (i) through a smart contract, it verifies the legitimacy of health professionals;

   (ii) record the metadata of patients' health records and ensure that they are accessed only by authorized healthcare professionals;

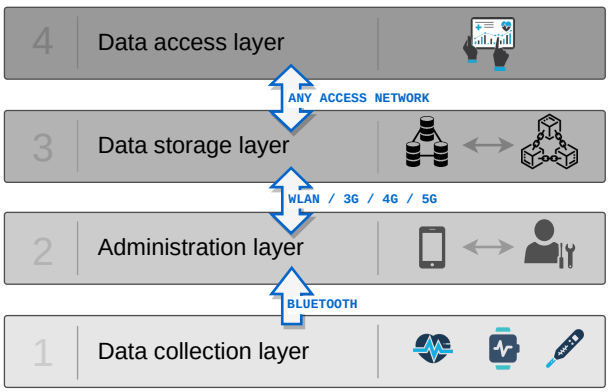
(iii) provide robustness against availability failures and data breach attacks.

5. *IPFS*. InterPlanetary File System (BENET, 2014) is a peer-to-peer protocol for storing distributed data. In an IPFS-based network, stored files are referenced through a hash that is calculated exclusively based on their content. Files stored on an IPFS network are immutable — if the file is changed, IPFS considers the changed file as a new object, so a new hash is calculated. The IPFS network is used here because the financial cost of storing large files on the blockchain is very high. In this case, an IPFS network can be used to store health records, while the blockchain stores only the data hash and metadata. More details about the interactions between blockchain and IPFS can be found in Wang *et al.* (2018).
6. *Healthcare professionals*. They are the users of the data, adequately authorized by the patient, such as doctors, dentists, nutritionists, specialized clinics, among others. These healthcare providers can analyze the data and provide guidance or indicate treatments.

5.2.1 The logical data architecture

Given the logical architecture of the data, the model is structured in four layers (see Figure 18). The architecture represents how data is handled from collection by monitoring devices to analysis by healthcare professionals. Regarding the communication between layers, our model follows the standard layered communication architecture of a WBAN, as presented in Al-Janabi *et al.* (2017), Latré *et al.* (2011).

Figure 18 – Logical data architecture.



Source: elaborated by the author.

Layer 1 - Data Collection. The data collected by the body sensors are used to monitor the patient's health. At a predefined time, the devices collect the patient's physiological data and transmit it to the next layer. Here, a WBAN using BLE is employed to connect the monitoring devices to the administrator device.

Layer 2 - Administration. This layer receives data coming from Layer 1 for processing before sending it to the storage service. The device used for this is usually the patient's smartphone, which acts as a gateway between the monitoring devices and the blockchain. Here, the patient has an application that provides system settings and access control functions. The communication between this layer and Layer 3 takes place through traditional home Wireless Local Area Networks (WLAN) or 3G/4G/5G mobile networks.It is also in this layer that the patient defines the data access policy based on ABE (see Section 3.7) — a policy for each group of data users (healthcare professionals).

Layer 3 - Data Storage. This layer refers to the data storage infrastructure using the blockchain and an IPFS network. Our approach proposes a wholly distributed architecture where no centralized third parties manipulate or store patient data. However, storing all health data on the blockchain is not adequate since it is expensive (THWIN; VASUPONGAYYA, 2019). Thus, we decided to use a distributed storage system, such as IPFS, to store most health data. The data stored in IPFS is referenced by its hash, which is immutably stored in the blockchain's smart contract. Users interested in the data can consult the smart contract, discover the hash, and request the corresponding health record from IPFS, which provides secure and immutable data storage.

Layer 4 - Data Access. Once collected and properly stored, the data is ready to be analyzed by authorized healthcare professionals. Users view data through applications available on their personal or institutional devices. Access to data by healthcare professionals must respect the access policy based on ABE, which is defined by the data owner in Layer 2.

5.2.2 Distributed Application

Our approach proposes a distributed application composed of three parts: an administrator application, a smart contract, and a data access application. The administrator application, installed on the administrator device, is for the patient's exclusive use. The smart contract is deployed on the blockchain and is responsible for controlling access to data. The data access application runs on the devices of authorized healthcare professionals.

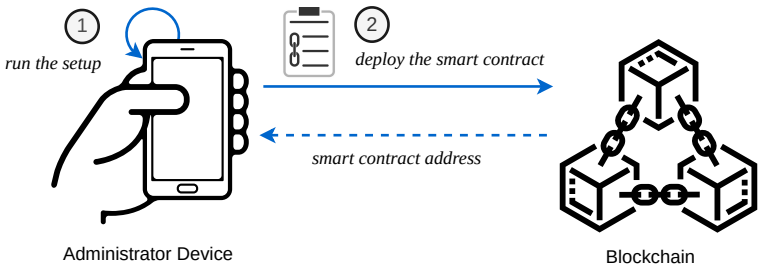
5.3 Initial system configuration

To ensure patient privacy, our approach proposes the use of Attribute-Based Encryption (ABE) at various points in the system, starting with the initial configuration. ABE is a cryptographic primitive that supports confidentiality and fine-grained access control over encrypted data, as we discussed in Section 3.7.

5.3.1 Initial configuration process

An initial configuration process is required to prepare the system for use. The configuration is performed only once by the data owner (patient) using the administrator device. Note that the administrator device assumes the role of the *Key Generation Center*, required in traditional ABE systems. The configuration consists of the following steps.

Figure 19 – Initial configuration process.



Source: elaborated by the author.

1. The administrator device must run the algorithm **`Setup`** $(\rho) \rightarrow (\mathcal{M}_\mathcal{K}, \mathcal{P}_\mathcal{K})$. The algorithm takes a predefined security parameter ρ, which can be the size of the groups on bilinear pairing, as defined by Bethencourt *et al.* (2007). The algorithm returns the system configuration parameters: the private master key $\mathcal{M}_\mathcal{K}$, which will be known only by the data owner, and the public key $\mathcal{P}_\mathcal{K}$ of the ABE scheme.
2. The administrator device deploys the smart contract, previously coded in the mHealth application, on the blockchain. After deployment, the mHealth application obtains the public address of the smart contract.

The smart contract stores the blockchain address of the mHealth application to ensure that only the administrator device can write data to the blockchain. The smart contract must verify the administrator device's legitimacy before recording data sent by it (see Section 5.5).

5.4 Data collection subsystem

The feasibility of remote health monitoring relies, fundamentally, on the correct and safe data collection by healthcare devices. Upon receiving data, the administrator device processes and transmits them to storage services, where healthcare professionals access them. However, the system's functioning may be impaired if the communication between the monitoring devices and the other players is not secure. Thus, it is crucial to authenticate these devices. In this section, to address the issues presented in the Section 1.1, we present the design of the device authentication mechanism.

5.4.1 The NIZKP-based authentication scheme

We propose a lightweight mechanism to verify the legitimacy of the monitoring devices. The mechanism is based on NIZKP (see Section 3.8.5) due to its security guarantees.

To build any NIZKP system, the choice of the mathematical problem that forms its base is a fundamental element. In this work, the basic problem chosen is the Elliptic Curve Discrete Logarithm Problem (ECDLP), discussed in the Section 3.5. Systems based on elliptic curves, when compared to RSA, require less computing power and less memory consumption while providing the same level of security. Our scheme replaces heavy asymmetric cryptography used in traditional Public Key Infrastructure (PKI) with one that is more suitable for resource-limited devices.

There are many ways to build ZKP systems. To implement the NIZKP system used in this work, we adopted a variation of the Schnorr protocol developed based on ECDLP. As mentioned before, we are especially interested in the non-interactive form of the protocol, presented by Hao (2017). To transform the interactive Schnorr protocol (see Section 3.9), into Schnorr protocol for NIZKP, we use the Fiat-Shamir heuristic, as shown in the Section 3.8.6.

Typically, a NIZKP system consists of two steps, as follows:

First stage. It involves a configuration process and, therefore, still requires some interactions between the parties. More specifically, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ must share some information. In our case, one of the agreed public information is the elliptic curve `secp256k1` (SECP256K1, 2019), the same used by the Bitcoin system. In practice, we assume that this proposal is implemented using the `secp256k1` curve for all ECC-based protocols. Specifically, the applications present on the monitoring devices, the administrator device, and

the blockchain are programmed to use the parameters defined in `secp256k1`. The second public information agreed between the parties is the public key of $\mathcal{P}$ that will be used in the authentication process. Finally, we include the third information agreed between the parties, exclusively for this proposal — the shared secret key for data encryption, which is known only to $\mathcal{P}$ and $\mathcal{V}$. This last two information is generated in the registration phase (see Section 5.4.2).

Second stage. This stage is where, in fact, NIZKP occurs. This is the authentication phase of the device, which must occur quickly and entirely in a non-interactive way. Here the proof is generated by $\mathcal{P}$ and validated by $\mathcal{V}$. The proof is generated based on the first phase information. The prover $\mathcal{P}$ sends the proof in a single message. Upon receiving the message, $\mathcal{V}$ processes the information and decides whether to accept or reject the proof.

Our authentication scheme requires that monitoring devices go through a registration process first. Thus, our scheme consists of two phases: registration and authentication.

5.4.2 Device registration phase

Initially, the monitoring device expects the administrator device to initiate a connection. After that, the parties must execute Protocol 1 (see Figure 20), which is based on ECDH (see Section 3.6). The shared secret key can be used in symmetric cryptographic systems to encrypt communication between the parties.

We assume that the registration process is carried out in a secure mode. For example, the patient must register a device in a private place where he/she is sure that only legitimate devices are present. With the device in hand, the patient can pair and register the device. This precaution minimizes the possibility of a malicious device getting registered.

Protocol 1 — Device Registration

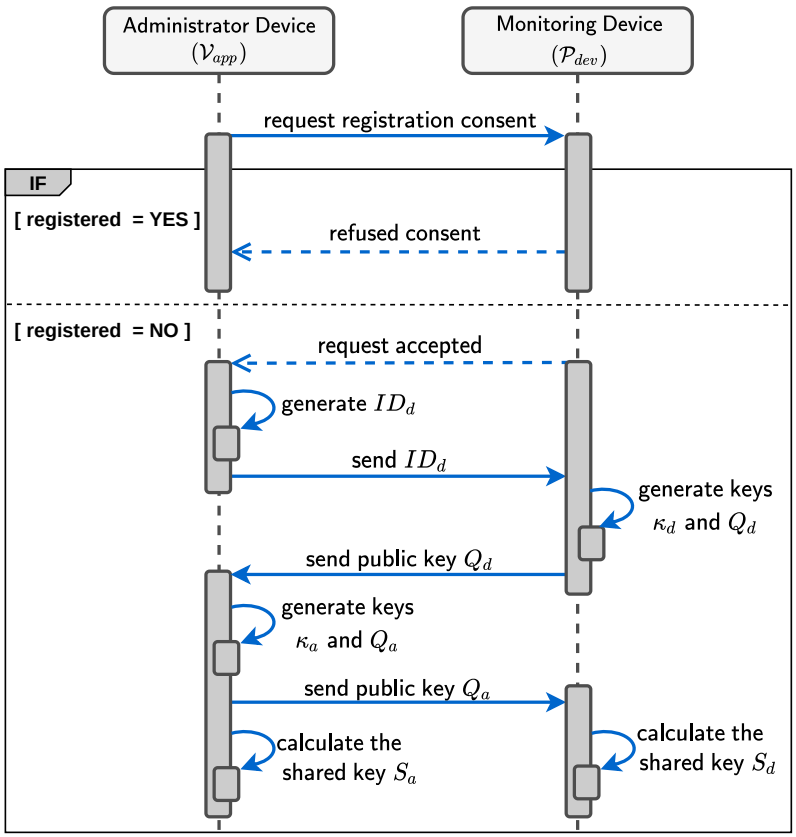
- ***Goal:*** *register a monitoring device.*
- ***Players:*** *the monitoring device $\mathcal{P}_{dev}$, operating as a provider, and the mHealth App $\mathcal{V}_{app}$ installed on the administrator device, operating as a verifier.*
- ***Common input:*** *curve $E(\mathbb{F}_p)$; generator $G \in E(\mathbb{F}_p)$.*

Steps:

1. *Assuming there is a Bluetooth connection between the devices, $\mathcal{V}_{app}$ sends to $\mathcal{P}_{dev}$ a registration consent message.*

2. *$\mathcal{P}_{dev}$ checks if there is already a previous association with another $\mathcal{V}_{app}$. If the answer is YES, the registration request is rejected because the device can not be linked to another App. Otherwise, the process continues at step 3.*
3. *$\mathcal{V}_{app}$ generates and sends an identifier ID_d to $\mathcal{P}_{dev}$.*
4. *$\mathcal{P}_{dev}$ generates its private key by choosing a secret integer $\kappa_d \in \mathbb{F}_p$ at random. It then generates its public key by calculating $Q_d = \kappa_d G$, in which $Q_d \in E(\mathbb{F}_p)$.*
5. *$\mathcal{P}_{dev}$ sends Q_d to $\mathcal{V}_{app}$ to allow $\mathcal{V}_{app}$ to calculate the shared secret key.*
6. *$\mathcal{V}_{app}$ generates its private key by choosing a secret integer $\kappa_a \in \mathbb{F}_p$ at random. It then generates its public key $Q_a = \kappa_a G$, in which $Q_a \in E(\mathbb{F}_p)$.*
7. *$\mathcal{V}_{app}$ sends its public key Q_a to $\mathcal{P}_{dev}$.*
8. *Now, both can calculate the shared secret key. $\mathcal{V}_{app}$ calculates $S_a = \kappa_a Q_d$ and $\mathcal{P}_{dev}$ calculates $S_d = \kappa_d Q_a$. So, the secret key is $S_a = S_d$.*

Figure 20 – The workflow of the registration phase of the monitoring devices.



Source: elaborated by the author.

5.4.3 Device authentication phase

In an ECDLP-based NIZKP system, each monitoring device uses a public key: point Q_d (generated by Protocol 1), to prevent a malicious prover from proving false statements. Thus, any proof prepared by a legitimate prover must be constructed based on Q_d. Authentication occurs according to Protocol 2 (see Figure 21).

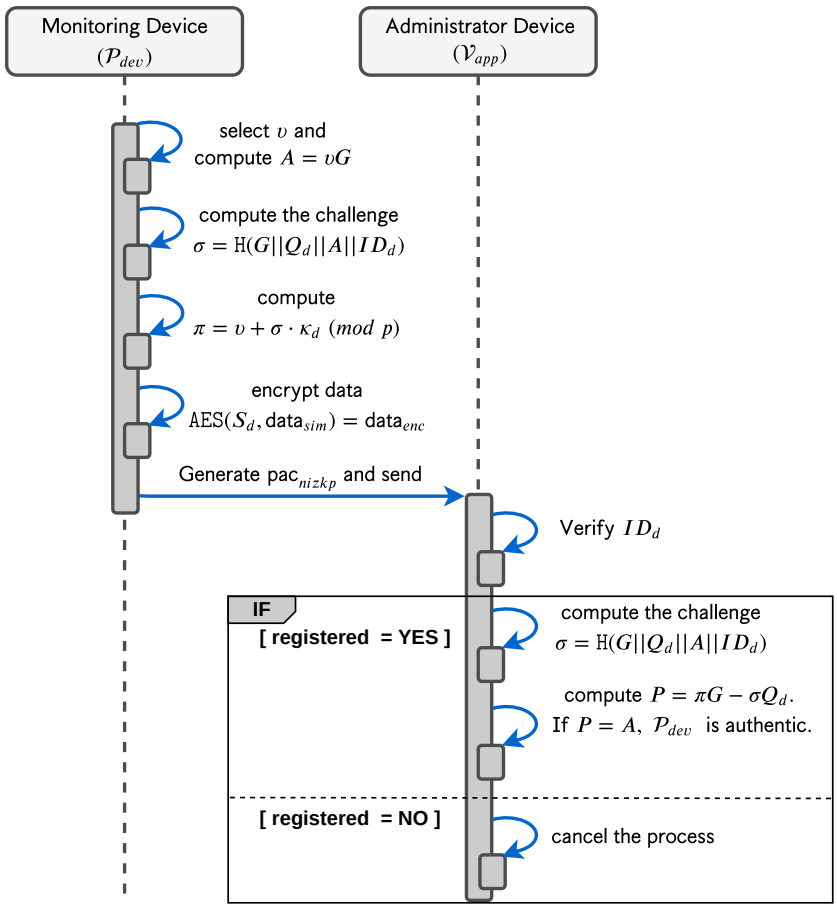
Protocol 2 — GENERATION OF NIZKP

- ***Goal:*** *authenticate a monitoring device to carry out the data transmission.*
- ***Players:*** *the monitoring device $\mathcal{P}_{dev}$, operating as a prover; the mHealth App $\mathcal{V}_{app}$ installed on the administrator device, operating as a verifier.*
- ***Common input:*** *curve $E(\mathbb{F}_p)$; generator $G \in E(\mathbb{F}_p)$; public key Q_d and ID_d of $\mathcal{P}_{dev}$.*
- ***Private input:*** *secret key S_d of $\mathcal{P}_{dev}$.*

Steps:

1. *$\mathcal{P}_{dev}$ collects, through body sensors, the health data set, denoted by* $\texttt{data}_{\texttt{sim}}$.
2. *$\mathcal{P}_{dev}$ chooses an integer $\upsilon \in \mathbb{F}_p$ at random and then calculates the point $A = \upsilon G$.*
3. *$\mathcal{P}_{dev}$ calculates the challenge σ using a cryptographic hash function $\mathcal{H}$, such as $\sigma = \mathcal{H}(G||Q_d||A||ID_d)$.*
4. *$\mathcal{P}_{dev}$ calculates the response π to the challenge σ, such that $\pi = \upsilon + \sigma \cdot \kappa_d \pmod{p}$.*
5. *$\mathcal{P}_{dev}$ encrypts health data. To do this, it runs the algorithm* $\texttt{AES}(S_d, \texttt{data}_{\texttt{sim}}) = \texttt{data}_{\texttt{enc}}$, *where* **AES** *is the symmetric encryption algorithm, S_d is the shared key generated by Protocol 1 and* $\texttt{data}_{\texttt{enc}}$ *is the encrypted data.*
6. *$\mathcal{P}_{dev}$ generates a package* $\texttt{pac}_{\texttt{nizkp}}$ *containing NIZKP and encrypted health data* $\texttt{data}_{\texttt{enc}}$. *The package* $\texttt{pac}_{\texttt{nizkp}}$ *is logically partitioned into four segments, containing the following information:*
   - *(i) The first contains the point A, calculated in step 2;*
   - *(ii) The second contains the response π, generated in step 4;*
   - *(iii) The third contains the device identifier ID_d;*
   - *(iv) The fourth contains the encrypted health data* $\texttt{data}_{\texttt{enc}}$.
7. *Finally, $\mathcal{P}_{dev}$ sends the package* $\texttt{pac}_{\texttt{nizkp}}$ *to $\mathcal{V}_{app}$.*

Figure 21 – Authentication process: generation of NIZKP and sending of data by the monitoring device; verification of NIZKP by the mHealth App to ensure the legitimacy of the device.



Source: elaborated by the author.

To ensure that the authentication package came from a legitimate device, the verifier must execute Protocol 3 (see Figure 21).

Protocol 3 — VERIFICATION OF NIZKP

- ***Goal:*** *verify the legitimacy of the monitoring device and receive the data.*
- ***Players:*** *the monitoring device $\mathcal{P}_{dev}$, operating as a prover; the mHealth App $\mathcal{V}_{app}$ installed on the administrator device, operating as a verifier.*
- ***Common input:*** *curve $E(\mathbb{F}_p)$; generator $G \in E(\mathbb{F}_p)$; public key Q_d and ID_d of $\mathcal{P}_{dev}$; package* $\mathtt{pac}_{\mathtt{nizkp}}$.

Steps:

1. *$\mathcal{V}_{app}$ receives the package* $\mathtt{pac}_{\mathtt{nizkp}}$ *formed by: point A; response π; ID_d; encrypted*

data $\texttt{data}_{\texttt{enc}}$.

2. *$\mathcal{V}_{app}$ checks if ID_d is from a registered device. If true, the process continues at step 3. Otherwise, the process is canceled.*
3. *$\mathcal{V}_{app}$ uses the public key Q_d associated with ID_d to calculate the challenge, as calculated by $\mathcal{P}_{dev}$, $\sigma = \mathcal{H}(G||Q_d||A||ID_d)$.*
4. *$\mathcal{V}_{app}$ calculates a point $P = \pi G - \sigma Q_d$ and checks if $P = A$.*
   - *If $P = A$, then $\mathcal{P}_{dev}$ is a legitimate device and Protocol 4 can be run to process the data.*
   - *If authentication fails, the process is terminated.*

When a security protocol is based on Schnorr's NIZKP, the threat of Replay Attacks must be considered. To avoid this specific attack, we can add more information to compose the hash function input that calculates the challenge. Such information must be a sequential number that identifies the package $\texttt{pac}_{\texttt{nizkp}}$. The verifier must observe whether the ID_d of the device and the package number form a unique identification.

5.4.4 Experimental evaluation

We conducted experiments to evaluate whether the protocols proposed in this section are suitable for running on devices with limited resources. The experiments' goal is to evaluate the algorithms' performance under two aspects: computational cost to generate NIZKP and the consumption of RAM and flash memories in the monitoring devices.

The environment of the experiments involves the following aspects:

- *Implementation.* The functions that involve ECC, within the protocols proposed in this section, were implemented based on the library `micro-ecc` (MACKAY, 2017). This library allows implementing the ECDH algorithm and Elliptic Curve Digital Signature Algorithm (ECDSA) on 8-bit processors using the C language. We use some functions from this library to implement a part of our protocols.
- *Elliptic curve.* We chose to implement the scheme using the elliptic curve `secp256k1` (SECP256K1, 2019), whose parameters are recommended by Standards for Efficient Cryptography Group (SECG) (Standards for Efficient Cryptography Group, 2010).

- *Hardware.* We chose a very limited device to represent the monitoring device: the Arduino Nano. It is a small prototyping board based on the Atmel ATMega 328P microcontroller (8-bit) clocked at 16MHz and only 2KB of RAM and 32KB of flash memory. To send the data, we added to the Arduino a Bluetooth module of type BLE V4.0 HM-10. As an administrator device, to receive and process data, we used an Android 9 smartphone with a quad-core 1.8 GHz processor and 4GB of RAM.

In this subsection, we present the experiment performed from the monitoring device. Table 8 shows the time required for the devices to perform each of the operations related to the registration process (Protocol 1). Table 9 shows the time required to perform operations related to the generation and verification of NIZKP (Protocol 2 and Protocol 3, respectively). Table 10 shows the amount of memory needed to run the proposed scheme on the monitoring device.

Table 8 – Average registration protocol runtime (Protocol 1).

Operations	**Arduino**	**Smartphone**
Key pair generation	3.784 s	0.195 s
Initial data exchange and processing (e.g., public key)	4.139 s	-
Generation of shared secret	3.785 s	0.045 s
Total execution time (including transmission): **13.144 s**		

Source: elaborated by the author.

Table 9 – Average NIZKP protocols runtime (Protocols 2 and 3).

Operations	**Arduino**	**Smartphone**
Encryption of health data	0.055 s	-
Challenge generation	0.033 s	-
Generation of NIZKP (including the challenge)	4.300 s	-
Formation of the data package (Protocol 2, step 6)	3.747 s	-
Verification of NIZKP	-	0.213 s
Decryption of data	-	0.014 s
Total execution time (including transmission): **9.384 s**		

Source: elaborated by the author.

Table 10 – Memory used by the authentication scheme on the monitoring device.

Data	Used Memory
Private key	32 Bytes
Public key	64 Bytes
Shared secret	32 Bytes
Complete data package	113 Bytes
Runtime algorithms	1.3 KB
Compiled algorithm (in flash memory)	24.5 KB

Source: elaborated by the author.

To calculate the energy consumption by the device when executing our authentication scheme, we use the following equation suggested in Chatzigiannakis *et al.* (2011), Ma *et al.* (2014), Moosavi *et al.* (2016), Li *et al.* (2017)

$$E = V \cdot I \cdot t, \tag{5.1}$$

where

- E is energy consumption in millijoules (mJ),
- V is the operating voltage in volts (V),
- I is the current draw in milliamps (mA),
- t is the time in seconds of each operation.

We estimate the power consumption by the Arduino Nano during the processing of each operation of the authentication mechanism. According to the Arduino Nano datasheet [1], it has a current consumption of 19 mA and a supply voltage of 5 V. As mentioned earlier, we have used a Bluetooth module of type BLE V4.0 HM-10 connected to the Arduino Nando to enable communication with the smartphone. The data rate for this module is 6 KBytes/sec, according to its datasheet [2]. Thus, considering the equation

$$E = 5 \cdot (19 + 1.5) \cdot t, \tag{5.2}$$

where 19 mA is the current draw of the Arduino and 1.5 mA is the current draw of the HM-10 module in sleep mode, we obtain the approximate energy consumption for each operation, as shown in Table 11.

[1] https://store.arduino.cc/usa/arduino-nano
[2] https://seeeddoc.github.io/BLE_Bee/res/Bluetooth40_en.pdf

Table 11 – Energy consumption by Arduino Nano during the processing of the operations of the authentication mechanism.

Operations	**Energy consumption**
Key pair generation	387.86 mJ
Initial data exchange and processing (e.g., public key)	424.25 mJ
Generation of shared secret	387.96 mJ
Encryption of health data	5.64 mJ
Challenge generation	3.38 mJ
Generation of NIZKP (including the challenge)	440.75 mJ
Formation of the data package (Protocol 2, step 6)	384.07 mJ

Source: elaborated by the author.

The energy consumption when receiving or transmitting a message x bytes can be estimated according to the following equation given by Wang *et al.* (2006), Cao *et al.* (2008), Shim (2014)

$$E = V \cdot I \cdot x \cdot 8(bits)/r, \tag{5.3}$$

where r is the data rate in bytes/second; and $I = 19 + 8.5$, where 19 mA is the current draw of the Arduino and 8.5 mA is the current draw of the HM-10 module in active mode. Thus, we can estimate the energy consumption of our scheme when transmitting and receiving the data packets. The results are shown in the Table 12.

Table 12 – Energy consumption by Arduino Nano during the communication with the smartphone.

Operations	**Energy consumption**		
	Message size	**Transmission**	**Reception**
Registration request	1 byte	0.12 mJ	0.12 mJ
Monitoring device ID	2 bytes	-	0.25 mJ
Public key exchange	64 bytes	8.10 mJ	8.10 mJ
NIZKP package (Protocol 2, step 6)	113 bytes	14.31 mJ	-

Source: elaborated by the author.

Note that the first three operations in Table 11 and Table 12 are performed only once — during the monitoring device's registration process. The remaining operations are performed whenever the monitoring device receives a request for data collection. Thus, our scheme, running

on the Arduino Nano, consumes a total of 1212.35 mJ for the registration of a monitoring device; and a total of 854.55 mJ during the data collection and transmission process. Assuming that the Arduino Nano is powered by a new battery with a capacity of 2200 mAh, easily found in specialized stores, we can perform the registration of the device and around 43.3 thousand monitoring operations, where each of these operations consists of the entire collection process and data transmission.

Based on these results, we found that the Arduino Nano is able to run the Device Registration algorithm (Protocol 1) and the NIZKP Generation algorithm (Protocol 2) while consuming few resources. At runtime, our scheme occupies only 63.5% of RAM available on this device. As for flash memory, note that the compiled code from Protocols 1 and 2 occupy only 24.5KB. The data encryption/decryption times refer to a 16-byte data block.

We believe that the execution times of the proposed are acceptable for a real mHealth environment. Especially, if we consider the device's low processing capacity and the level of security offered by the scheme. The level of security is compared to an RSA-based scheme with a 3072-bit key. It is difficult to make a broad/comprehensive comparison of our approach with related work (see Section 5.1); most existing contributions do not address authentication between wearable devices and smartphones (or other types of gateways). To the best of our knowledge, our proposal is unique in the set of security characteristics it supports. However, we found that our proposal can be qualitatively compared with Moosavi *et al.* (2015); the comparison concerns the authentication process. Moosavi *et al.* (2015) proposed an authentication scheme between medical devices and a smart e-health gateway. They developed a public key-based handshake protocol. To evaluate their proposal, they used a medical device equipped with a 16MHz MSP430 microcontroller, 128KB of ROM, and 16KB of RAM. The authentication process between the device and the gateway takes approximately 15 seconds. Note that the device used in the experiments of Moosavi *et al.* (2015) is similar to the one we used in our experiments; however, our authentication scheme achieves a shorter response time of approximately 9 seconds. Furthermore, in our scheme, the transmission overhead is only 113 bytes, whereas in the proposal of Moosavi *et al.* (2015), it is 1190 bytes.

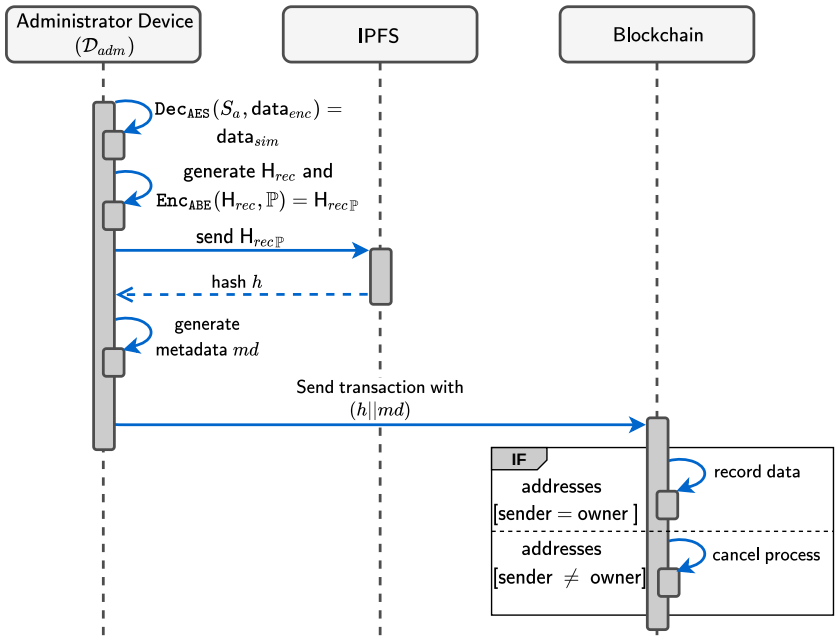
The idea of using the Arduino Nano is to show that our scheme is capable of running on most current mHealth devices. This is confirmed by the results obtained here, which show that our scheme consumes fewer resources than the minimum suggested in many reference projects for the mHealth industry (Texas Instruments, 2020a; Texas Instruments, 2020b; Microchip,

2020). The results of Table 8, Table 9 and Table 10 allow us to say that our security scheme is suitable for resources limited mHealth devices.

5.5 Data administration subsystem

In this section, we present the details of the administration layer. Here, the data owner uses the smartphone to receive the data from the monitoring devices. Once the authenticity of the monitoring device, which sents the package $\texttt{pac}_{\texttt{nizkp}}$, is verified, the administrator device can execute Protocol 4 to process the data (see Figure 22).

Figure 22 – Data processing: formatting of data received from monitoring devices, administrator device authentication, and ABE-based data sharing.



Source: elaborated by the author.

Protocol 4 — DATA PROCESSING

- ***Goal:*** *processing and sharing of data.*
- ***Players:*** *the administrator device* $\mathcal{D}_{adm}$ *and the smart contract.*
- ***Common input:*** *the package* $\texttt{pac}_{\texttt{nizkp}}$.
- ***Private input:*** *shared secret key* S_a *of* $\mathcal{D}_{adm}$.

Steps:

1. $\mathcal{D}_{adm}$ *decrypts the package* $\texttt{data}_{\texttt{enc}}$ *running the algorithm* $\textbf{Dec}_{\textbf{AES}}(S_a, \texttt{data}_{\texttt{enc}}) = \texttt{data}_{\texttt{sim}}$,

where $\mathbf{Dec_{AES}}$ *is the symmetric decryption algorithm,* S_a *is the shared key generated by the Protocol 1 and* $\mathtt{data_{sim}}$ *is the decrypted data package.*

2. $\mathcal{D}_{adm}$ *uses* $\mathtt{data_{sim}}$ *to generate a file* $\mathtt{H_{rec}}$ *that corresponds to a health record.*
3. $\mathcal{D}_{adm}$ *encrypts* $\mathtt{H_{rec}}$ *using an ABE algorithm for a given access policy* $\mathbb{P}$*, such as* $\mathbf{Enc_{ABE}}(\mathtt{H_{rec}}, \mathbb{P}) = \mathtt{H_{rec}}_{\mathbb{P}}$.
4. $\mathcal{D}_{adm}$ *sends the encrypted file* $\mathtt{H_{rec}}_{\mathbb{P}}$ *to the IPFS network.*
5. *IPFS generates a hash* h *for the uploaded file and returns* h *to* $\mathcal{D}_{adm}$*. In IPFS, the hash is used as the location of the file.*
6. $\mathcal{D}_{adm}$ *generates the metadata* md *corresponding to the health record* $\mathtt{H_{rec}}$*. Metadata will be used to search health records by healthcare professionals.*
7. $\mathcal{D}_{adm}$ *sends a blockchain transaction carrying the following information:*
   - *hash* h*, which represents* $\mathtt{H_{rec}}_{\mathbb{P}}$ *on the IPFS;*
   - *metadata* md *linked with the health record.*
8. *After receiving the transaction, the smart contract verifies that the transaction was sent by* $\mathcal{D}_{adm}$*, comparing the sender's address with the contract owner's address.*
   - *If the addresses are the same, then* $\mathcal{D}_{adm}$ *is the legitimate device; and then the data (metadata and file hash) are recorded on the blockchain.*
   - *If not, the data are rejected.*

5.5.1 Experiments with blockchain

The experiments in this section are intended to evaluate the operations involving the administrator device and the blockchain. We consider two aspects for evaluation:

1. The time spent by the administrator device to perform the operations in Table 13;
2. The Ether cost of each transaction in Table 14.

When evaluating the time cost of operations, we do not consider communication costs between the administrator device and the blockchain. The reason for this is that we use the Ethereum blockchain, where we have no control over the processing time of its operations. Thus, we focus on the impact that our approach has on administrator device.

The environment of the experiments involves the following aspects:

- *Blockchain platform.* We use the Ethereum [3] blockchain to conduct the experi-

[3] https://ethereum.org

ments. For the interactions between the administrator device and the blockchain, we used the Rinkeby[4] network, an Ethereum tool for testing and development. It allows calls to the blockchain at no financial cost for transactions.

- *Implementation.* We developed the mHealth administrator application using the Android platform. To implement the operations related to Attribute-Based Encryption, we use the libraries java `cpABE` (WANG, 2012) and `jPBC` (CARO; IOVINO, 2011). We developed the smart contract, which operates as a legitimacy checker, using the Solidity [5] language. The smart contract is deployed on the Ethereum blockchain by sending the transaction *deploy*(). All interactions between the Android application and the smart contract were implemented using the `web3j` [6] library. Although the smart contract address is public, only the administrative device can write data to the blockchain, according to Protocol 4. In order to interact with the IPFS network, our application uses the `IPFS-lite`[7] library to instantiate and run an IPFS client. Thus, it is possible to send the data and receives the hash of the file sent in return.
- *Hardware.* We use an Android 9 smartphone with a quad-core 1.8 GHz processor and 4GB of RAM as an administrator device.

Table 13 shows the average time spent by the device administrator to perform each of the operations related to data processing or sharing. Table 14 shows an estimate of the cost, incurred by the patient, to execute the blockchain transactions. Although we have presented the price of transactions in Ether, it is possible to convert that price to the dollar or another currency.

Table 13 – Time spent by the administrator device to perform operations on the data management subsystem.

Operations	**Average Time**
The time required to encrypt data using ABE	0.912 s
The time required to generate the metadata	0.054 s
The time required to deliver data to the IPFS network	0.239 s

Source: elaborated by the author.

[4] https://www.rinkeby.io/
[5] https://solidity.readthedocs.io
[6] https://github.com/web3j/web3j
[7] https://github.com/textileio/android-ipfs-lite

Table 14 – Estimated cost per transaction on the Ethereum blockchain.

Operations	Gas Used	Price (Ether)
Deployment of the smart contract	971,548	0.0039833468
Sending data	1,055,691	0.0043283350

Source: elaborated by the author.

5.6 Data access subsystem

In our approach, the player who wants to access patient data is equipped with an application capable of interacting with the smart contract on the blockchain. The smart contract verifies the legitimacy of the user.

In this section, we will refer to healthcare professionals as *data users*, which is the term traditionally used in ABE-based schemes. Our approach ensures that patient data is shared only with duly authorized users. To do this, the scheme that authenticates users consists of two phases: the user registration phase (see Figure 23) and the authentication phase (see Figure 24).

5.6.1 User registration phase

Our approach requires an interactive step between the patient and the healthcare professional before the registration phase — a type of *pre-authentication*. In this interaction, the patient must transmit to the healthcare professional the address of the smart contract, denoted by $\texttt{add}_{con}$, and the professional's identifier ID_u, randomly generated. Note that, usually, the first contact between the patient and the health professional is in person; therefore, the patient can share ID_u and $\texttt{add}_{con}$ during this meeting. Then, the data owner registers the user according to the Protocol 5.

Protocol 5 — Healthcare professional Registration

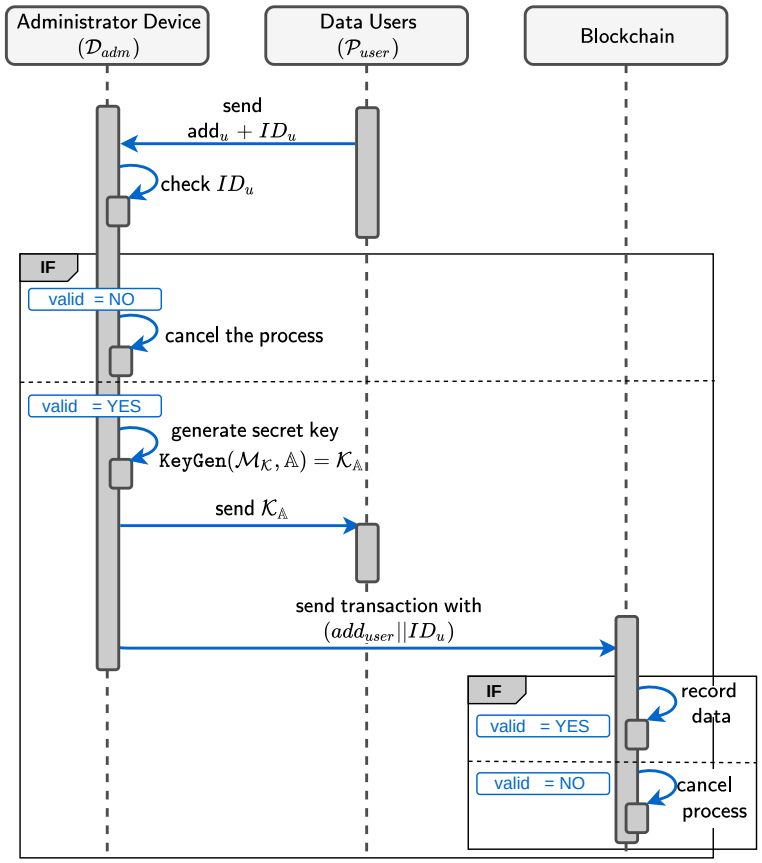
- ***Goal:*** *register a data user.*
- ***Players:*** *data user $\mathcal{P}_{user}$; administrator device $\mathcal{D}_{adm}$; and smart contract.*
- ***Secret input:*** *private key $\mathcal{M}_{\mathcal{K}}$, generated in the initial system configuration.*

Steps:

1. *$\mathcal{P}_{user}$ sends his/her blockchain address $\texttt{add}_{user}$ and his/her ID_u received from the patient previously.*

2. *$\mathcal{D}_{adm}$ checks if ID_u received is the same as the one sent in the pre-authentication step.*
   - *If true, the process continues at step 3.*
   - *Otherwise, the registration process is canceled.*
3. *$\mathcal{D}_{adm}$ generates the user's secret key by running the algorithm* $\mathbf{KeyGen}(\mathcal{M}_{\mathcal{K}}, \mathbb{A}) = \mathcal{K}_{\mathbb{A}}$*; which takes the master key $\mathcal{M}_{\mathcal{K}}$ and the attribute set $\mathbb{A}$ as input.*
4. *$\mathcal{D}_{adm}$ sends to $\mathcal{P}_{user}$, via a secure channel, the secret key $\mathcal{K}_{\mathbb{A}}$. Note that at this stage, both $\mathcal{P}_{user}$ and $\mathcal{D}_{adm}$ have the computational power to use a communication channel that implements, for example, secure asymmetric encryption techniques.*
5. *$\mathcal{D}_{adm}$ sends a transaction to the smart contract, in order to record the user registration data, containing the user's ID_u and* add_{user}.
6. *After receiving the transaction, the smart contract verifies that the transaction was sent by $\mathcal{D}_{adm}$ (as in Protocol 4).*
   - *If true, user data is recorded in the smart contract.*
   - *If not, data are rejected.*

Figure 23 – The workflow of the registration phase of data users.

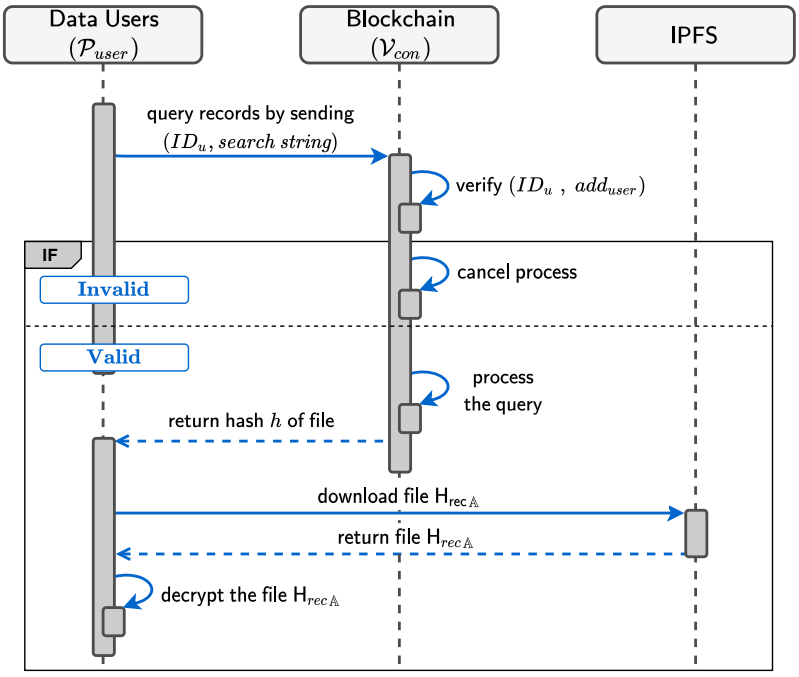


Source: elaborated by the author.

5.6.2 Data access phase

In this phase, the users duly registered by the patient can access the data. Healthcare professionals can monitor patient health through applications available on their computing devices. In each access request, a user must be authenticated according to Protocol 6 (see Figure 24).

Figure 24 – The workflow of the authentication phase of data users.



Source: Elaborated by the author.

Protocol 6 — DATA ACCESS

- ***Goal:*** *allow access to data to monitor patient health.*
- ***Players:*** *data user $\mathcal{P}_{user}$; smart contract; IPFS network.*
- ***private input:*** *user's secret key $\mathcal{K}_{\mathbb{A}}$ for the ABE scheme.*

Steps:

1. *To search for patient health records, $\mathcal{P}_{user}$ sends a blockchain transaction containing ID_u and the search string:*
2. *The smart contract checks if the sender's address and ID correspond to any pair (add_{user}, ID_u) stored in the smart contract. If so, the process continues at step 3. Otherwise, the process is canceled.*

3. *The smart contract processes the query and returns the hash* $\mathtt{h}$ *of all records matching the search.*
4. $\mathcal{P}_{user}$ *downloads the file* $\mathtt{H}_{\mathtt{rec}\mathbb{P}}$ *corresponding to* $\mathtt{h}$.
5. $\mathcal{P}_{user}$ *decrypts the health record using algorithm* $\mathtt{Dec}_{\mathtt{ABE}}(\mathtt{H}_{\mathtt{rec}\mathbb{P}}, \mathcal{K}_{\mathbb{A}}) = \mathtt{H}_{\mathtt{rec}}$*, which takes the encrypted file* $\mathtt{H}_{\mathtt{rec}\mathbb{P}}$ *and the secret key* $\mathcal{K}_{\mathbb{A}}$ *as input. The output is the file* $\mathtt{H}_{\mathtt{rec}}$*. But only if* $\mathbb{A}$ *satisfies* $\mathbb{P}$*, where* $\mathbb{A}$ *is the set of attributes associated with the user's key* $\mathcal{K}_{\mathbb{A}}$.

5.7 Discussion

In this section, we will discuss each layer of our approach to point out its impacts.

The *data collection layer* (Section 5.4) consists of preparing and transmitting the data. Note that we are not interested in the type or form in which health data is collected. We are interested when the data is already available for transmission. For the first part of the approach, we present an authentication scheme for the monitoring devices. The scheme is based on NIZKP, having the ECDLP as the mathematical problem. With this, we achieve the following outcomes.

- An exclusive association between the monitoring device and the mHealth application; this is, a monitoring device can be paired only with the official mHealth application. Thus, the scheme prevents a malicious application, eventually installed on the administrator device, from communicating with the monitoring device and stealing the data. Besides, we can also prevent an illegitimate device from injecting false data into the system. This improvement is because the fake device is not able to discover the private key of a legitimate device to perform authentication.
- Data traffic between the monitoring device and the smartphone is symmetrically encrypted. The algorithm implemented in the monitoring devices encrypts the data, and only the mHealth application can decipher it. This means that a malicious application, even if it can establish communication with the monitoring device, will receive the stream of encrypted data; therefore, it will not be able to read it. As this scenario requires sharing a secret key between the mHealth application and the device, we propose using the ECDH protocol to generate and share the key.
- As mentioned earlier, ECC offers the same level of security when compared to

other asymmetric encryption systems, using a significantly smaller key. Our scheme uses a 256-bit key only to provide the same security level as RSA with a 3072-bit key (BAFANDEHKAR *et al.*, 2013; BOS *et al.*, 2009). Even with a high level of security, we were able to implement this scheme on resource-limited devices, requiring low execution time and little memory space, as shown in the results of Section 5.4.4.

Note that the security of the proposed scheme, for the data collection layer, is based on the difficulty of solving the ECDLP. Thus, respecting the elliptic curve parameters, as recommended by SECG (Standards for Efficient Cryptography Group, 2010), no algorithm solves the ECDLP in polynomial time. Thus, with the results shown above, we can say that our authentication scheme solves the security problems presented in Section 1.1.

At the *data administration layer*, we employ a combination of ABE, blockchain, and IPFS. This combination results in an efficient administration of the system by the patient. More specifically, we achieve the following outcomes on this layer.

- We eliminate the need for a trusted third party, which is very common in cryptographic systems, responsible for generating and distributing the encryption/decryption keys to users. The administrator device assumes the role of the trusted authority. In this way, the patient himself/herself can generate the keys and allow access to the data only to the desired users.
- We assume that a particular administrator device controls the system. Therefore, to prevent other devices from writing data to the blockchain, the system requires authentication of the administrator device before allowing health data storage.

In the *data storage layer*, we chose to include a decentralized storage system, which involves the blockchain and the IPFS network. With that, we achieve the following.

- We have eliminated the problem of single point of failure, which is one of the biggest concerns in traditional centralized storage systems.
- The data is stored encrypted using the ABE scheme. Thus, we have been able to guarantee fine-grained access control. Indeed, the data owner chooses who can access and what data can be accessed based on an access policy.
- The data cannot be changed or deleted due to the immutability property. Each transaction stored in the blockchain has a corresponding hash, and a Merkle tree is generated from the hashes of the transactions included in the block. The

Merkle tree's hash value is stored in the block header together with a timestamp and the hash of the previous block. Therefore, if an attacker wants to tamper with a record in the blockchain, he/she needs not only to modify the hash of the block, but also to modify the hash of all subsequent blocks which are nearly impossible to achieve (JIANG *et al.*, 2019). Note that, with the guarantee of immutability and ABE scheme, our approach eliminates the risk of data being unduly exposed or tampered with in the event of attacks.

In the *data access layer*, a user who wants to access the data must obey the following mechanisms.

- The first step is to search on metadata associated with health records stored in the blockchain. The smart contract is responsible for ensuring the legitimacy of the data users.
- Once authenticated, the user obtains the encrypted health record using the ABE scheme. To decipher the record, the user needs a decryption key that satisfies the access policy defined by the data owner. Note that, even if an attacker randomly gets the hash that identifies a health record in IPFS, the file cannot be decrypted without the decryption key that satisfies the access policy.

Our proposal results in a system that guarantees the patient's privacy from end-to-end, that is, from collection to data storage. All players in the system must go through an authentication process. In the case of monitoring devices, the authentication process is done on the smartphone. In the case of data users, authentication is done in the smart contract on the blockchain. In case of attacks, an attacker would only be able to subvert the authentication scheme if he/she can resolve ECDLP. However, this is considered a computationally hard problem, and there is no polynomial-time algorithm to solve it.

5.7.1 Implementation issues

For implementations in real scenarios, the application that adopts the proposal presented in this thesis must offer a data backup feature for the user. As we have seen throughout this work, the proper working of the mHealth app relies fundamentally on the private keys and other identification data of the devices. As the private data is exclusively stored on the user's administrator device (smartphone), some mechanism is required to recover this data in case

of loss or theft of the smartphone. Note that a backup feature is part of the standard set of features that a mobile application should offer its user — that is, it is related to the design of the application. In this thesis, we are essentially concerned with designing the security and privacy-preserving mechanism for health data. Thus, the design of the backup mechanism is outside the scope of this work; however, we emphasize that it is a fundamental resource for any mHealth application.

Our approach is based on a public blockchain, which involves costs related to the cryptocurrency used by the blockchain network. Our proposal can be migrated to any blockchain that supports smart contracts, even private blockchain if it offers a lower cost to the patient. Despite this, we believe in public blockchain's potential due to the trust issues discussed in the Section 1.2.1. We believe that any service of security and privacy-preserving provided by healthcare applications requires a cost, which the patient will accept — as long as it is compatible with the benefits. Besides, according to Saito and Iwamura (2019), the evolution of cryptocurrency rates will become more stable over time. Even better, according to Hammi *et al.* (2018), Ethereum developers and the community are working to regulate and stabilize fee values related to the use of smart contracts.

6 CONCLUSION

We propose an approach for mHealth systems integrated with the blockchain that offers a high level of security and guarantee of patient privacy. We present an authentication scheme that associates each monitoring device exclusively with the official mHealth application. With this, we eliminate the risk of spoofed devices or malicious applications infiltrating the system. The experiments show that our NIZKP-based scheme over ECDLP is safe and, at the same time, sufficiently lightweight to run on resource-limited devices.

Our access control scheme, based on ABE and integrated with the blockchain, results in a significant improvement in patient privacy. To access data, users submit to two levels of security. Initially, an authentication process is performed by a smart contract on the blockchain. Once authenticated, users obtain the encrypted data and only decrypt it with a secret key that satisfies the patient's access policy. Our approach eliminates the need for a trusted central authority. Here, the patient's administrator device is responsible for generating and distributing secret keys to users duly registered in the system. This management method provides full control of the system to the data owner.

As the data collected by a mHealth system is extremely sensitive, some security and privacy requirements are essential; these include confidentiality, integrity, access control, availability, and patient-centered data control. We proposed solutions for all these security requirements to address the challenges of privacy-preserving in mHealth systems.

6.1 Future works and improvements

As future work, we consider implementing an important resource: the patient's ability to revoke access to his/her data. In some cases, the patient wishes that a particular health professional, who has been granted access to the data previously, no longer has access to their health data. This can happen because the patient has finished the treatment, changed doctors, or lost confidence in the health professional. We did not address the issue of revoking access to data in the scheme proposed in this thesis; however, we consider this a fundamental resource for improving the security and privacy of data in mHealth systems. In the future, we intend to design a mechanism for revoking access to patient data that should be easily integrated into the proposal presented in this work.

We performed the experiments aiming at feasible implementations in wearable

medical devices with computational power equal to or greater than the Arduino Nano, as specified in Section 5.4.4. Although the results on this type of device have been quite satisfactory, we recognize that implantable medical devices may have even more limited resources. Thus, to investigate the feasibility of implementing our proposal on implantable medical devices, we intend to conduct new experiments on certain groups of these devices.

When performing data collection, we set the size of the data block to 16 bytes. A block of data of this size can adequately carry most health data, such as heart rate, blood pressure, body temperature, and oxygenation rate. However, other health data can be more complex and require a data block larger than 16 bytes. We plan to investigate the working of our scheme in scenarios where health data requires more extensive data blocks in order to evaluate the impact of this on runtime, memory consumption, and energy consumption.

REFERENCES

AHMAD, M.; AMIN, M. B.; HUSSAIN, S.; KANG, B. H.; CHEONG, T.; LEE, S. Health fog: a novel framework for health and wellness applications. **The Journal of Supercomputing**, Springer, v. 72, n. 10, p. 3677–3695, 2016.

AL-JANABI, S.; AL-SHOURBAJI, I.; SHOJAFAR, M.; SHAMSHIRBAND, S. Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications. **Egyptian Informatics Journal**, Elsevier, v. 18, n. 2, p. 113–122, 2017.

ALI, M. S.; VECCHIO, M.; PINCHEIRA, M.; DOLUI, K.; ANTONELLI, F.; REHMANI, M. H. Applications of blockchains in the internet of things: A comprehensive survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 21, n. 2, p. 1676–1717, 2018.

ALMOTIRI, S. H.; KHAN, M. A.; ALGHAMDI, M. A. Mobile health (m-health) system in the context of iot. In: IEEE. **2016 IEEE 4th international conference on future internet of things and cloud workshops (FiCloudW)**. Vienna, Austria, 2016. p. 39–42.

AMEEN, M. A.; LIU, J.; KWAK, K. Security and privacy issues in wireless sensor networks for healthcare applications. **Journal of medical systems**, Springer, v. 36, n. 1, p. 93–101, 2012.

ANTONOPOULOS, A. **Bitcoin security model: trust by computation. Radar**. O'Reilly, 2014. Available at: http://radar.oreilly.com/2014/02/bitcoin-security-model-trust-by-computation.html. Accessed on: 21.10.2020.

AZARIA, A.; EKBLAW, A.; VIEIRA, T.; LIPPMAN, A. Medrec: Using blockchain for medical data access and permission management. In: IEEE. **2016 2nd International Conference on Open and Big Data (OBD)**. Vienna, Austria, 2016. p. 25–30.

BACK, A. Hashcash-a denial of service counter-measure. Working Paper, 2002. Available at: ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf. Accessed on: 05.12.2020.

BAFANDEHKAR, M.; YASIN, S. M.; MAHMOD, R.; HANAPI, Z. M. Comparison of ecc and rsa algorithm in resource constrained devices. In: IEEE. **2013 International Conference on IT Convergence and Security (ICITCS)**. Macao, China, 2013. p. 1–3.

BAJWA, M. mhealth security. **Pakistan Journal of Medical Sciences**, Professional Medical Publications, v. 30, n. 4, p. 904, 2014.

BAKER, S. B.; XIANG, W.; ATKINSON, I. Internet of things for smart healthcare: Technologies, challenges, and opportunities. **IEEE Access**, v. 5, p. 26521–26544, 2017.

BANSAL, G.; ZAHEDI, F. M.; GEFEN, D. The impact of personal dispositions on information sensitivity, privacy concern and trust in disclosing health information online. **Decision Support Systems**, v. 49, n. 2, p. 138 – 150, 2010.

BASTIAAN, M. Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin. University of Twente, 2015. Available at: https://fmt.ewi.utwente.nl/media/175.pdf. Accessed on: 05.10.2020.

BELLARE, M.; ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In: ACM. **Proceedings of the 1st ACM Conference on Computer and Communications Security**. Virginia, USA, 1993. p. 62–73.

BENET, J. **IPFS - Content Addressed, Versioned, P2P File System**. 2014. Available at: https://arxiv.org/abs/1407.3561. Accessed on: 21.09.2020.

BETHENCOURT, J.; SAHAI, A.; WATERS, B. Ciphertext-policy attribute-based encryption. In: IEEE. **2007 IEEE symposium on security and privacy (SP'07)**. Berkeley, CA, USA, 2007. p. 321–334.

BINDAHMAN, S.; ZAKARIA, N. Privacy in health information systems: a review. In: SPRINGER. **International Conference on Informatics Engineering and Information Science**. Berlin, Germany, 2011. p. 285–295.

BLUM, M.; FELDMAN, P.; MICALI, S. Non-interactive zero-knowledge and its applications. In: ACM. **Proceedings of the twentieth annual ACM symposium on Theory of computing**. Illinois, Chicago, USA, 1988. p. 103–112.

BOLFING, A. **Cryptographic Primitives in Blockchain Technology: A Mathematical Introduction**. United Kingdom: Oxford University Press, 2020. ISBN 9780198862840.

BOOTLE, J.; CERULLI, A.; CHAIDOS, P.; GROTH, J. Efficient zero-knowledge proof systems. In: **Foundations of security analysis and design VIII**. Bertinoro, Italy: Springer, 2016. p. 1–31.

BOS, J. W.; KAIHARA, M. E.; KLEINJUNG, T.; LENSTRA, A. K.; MONTGOMERY, P. L. **On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography**. Switzerland, 2009. https://eprint.iacr.org/2009/389.

CACHIN, C. **Architecture of the hyperledger blockchain fabric**. Zurich, Switzerland, 2016. https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf.

CAO, X.; KOU, W.; DANG, L.; ZHAO, B. Imbas: Identity-based multi-user broadcast authentication in wireless sensor networks. **Computer communications**, Elsevier, v. 31, n. 4, p. 659–667, 2008.

CARO, A. D.; IOVINO, V. jpbc: Java pairing based cryptography. In: **Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011**. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011. p. 850–855. Available at: http://gas.dia.unisa.it/projects/jpbc/.

CHAIDOS, P.; COUTEAU, G. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: SPRINGER. **Annual International Conference on the Theory and Applications of Cryptographic Techniques**. Tel Aviv, Israel, 2018. p. 193–221.

CHATZIGIANNAKIS, I.; PYRGELIS, A.; SPIRAKIS, P. G.; STAMATIOU, Y. C. Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices. In: IEEE. **2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems**. Valencia, Spain, 2011. p. 715–720.

CHOMSKY, N. Three models for the description of language. **IRE Transactions on information theory**, IEEE, v. 2, n. 3, p. 113–124, 1956.

CHOMSKY, N. On certain formal properties of grammars. **Information and control**, Elsevier, v. 2, n. 2, p. 137–167, 1959.

CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and smart contracts for the internet of things. **Ieee Access**, Ieee, v. 4, p. 2292–2303, 2016.

COOK, S. A. The complexity of theorem-proving procedures. In: **Proceedings of the Third Annual ACM Symposium on Theory of Computing**. New York, NY, USA: Association for Computing Machinery, 1971. (STOC '71), p. 151–158. ISBN 9781450374644.

CORMEN, T.; LEISERSON, C.; RIVEST, R.; STEIN, C. **Introduction to Algorithms**. London, England: MIT Press, 2009. (The MIT Press). ISBN 9780262258104.

COUTEAU, G. **Zero-knowledge proofs for secure computation**. Phd Thesis (Theses) — PSL Research University, Nov. 2017. Available at: https://tel.archives-ouvertes.fr/tel-01668125.

CRAMER, R.; DAMGÅRD, I. Secret-key zero-knowlegde and non-interactive verifiable exponentiation. In: NAOR, M. (Ed.). **Theory of Cryptography**. Berlin, Heidelberg: Springer, 2004. p. 223–237. ISBN 978-3-540-24638-1.

CREIGNOU, N.; KHANNA, S.; SUDAN, M. **Complexity Classifications of Boolean Constraint Satisfaction Problems**. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2001. (Discrete Mathematics and Applications). ISBN 9780898718546.

DAGHER, G. G.; MOHLER, J.; MILOJKOVIC, M.; MARELLA, P. B. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. **Sustainable cities and society**, Elsevier, v. 39, p. 283–297, 2018.

DAMGÅRD, I. B. A design principle for hash functions. In: BRASSARD, G. (Ed.). **Advances in Cryptology — CRYPTO' 89 Proceedings**. New York, NY: Springer New York, 1990. p. 416–427. ISBN 978-0-387-34805-6.

DAS, A. K.; WAZID, M.; KUMAR, N.; KHAN, M. K.; CHOO, K. R.; PARK, Y. Design of secure and lightweight authentication protocol for wearable devices environment. **IEEE Journal of Biomedical and Health Informatics**, v. 22, n. 4, p. 1310–1322, 2018.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE transactions on Information Theory**, IEEE, v. 22, n. 6, p. 644–654, 1976.

DIFFIE, W.; HELLMAN, M. E. Multiuser Cryptographic Techniques. In: NATIONAL COMPUTER CONFERENCE AND EXPOSITION, 1976, New York. **Anais...** New York: ACM, 1976. (AFIPS '76), p. 109–112.

DWIVEDI, A. D.; SRIVASTAVA, G.; DHAR, S.; SINGH, R. A decentralized privacy-preserving healthcare blockchain for iot. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 2, p. 326, 2019.

EDMONDS, J. Minimum partition of a matroid into independent subsets. **J. Res. Nat. Bur. Standards Sect. B**, v. 69, p. 67–72, 1965.

EYAL, I.; SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. In: **Financial Cryptography and Data Security**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 436–454. ISBN 978-3-662-45472-5.

FARAHANI, B.; FIROUZI, F.; CHANG, V.; BADAROGLU, M.; CONSTANT, N.; MANKODIYA, K. Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and healthcare. **Future Generation Computer Systems**, v. 78, p. 659 – 676, 2018. ISSN 0167-739X.

FIAT, A.; SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In: **Advances in Cryptology — CRYPTO' 86**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987. p. 186–194.

FILIPPI, P. D.; MANNAN, M.; REIJERS, W. Blockchain as a confidence machine: The problem of trust & challenges of governance. **Technology in Society**, Elsevier, v. 62, p. 101284, 2020.

FRANCO, P. **Understanding Bitcoin: Cryptography, Engineering and Economics**. United Kingdom: Wiley, 2014. (The Wiley Finance Series). ISBN 9781119019145.

GAO, W.; HATCHER, W. G.; YU, W. A survey of blockchain: Techniques, applications, and challenges. In: IEEE. **2018 27th international conference on computer communication and networks (ICCCN)**. Hangzhou, China, 2018. p. 1–11.

GAREY, M.; JOHNSON, D. **Computers and Intractability: A Guide to the Theory of NP-completeness**. United States: W. H. Freeman, 1979. ISBN 9780716710448.

GENESTIER, P.; ZOUARHI, S.; LIMEUX, P.; EXCOFFIER, D.; PROLA, A.; SANDON, S.; TEMERSON, J.-M. Blockchain for consent management in the ehealth environment: A nugget for privacy and security challenges. **Journal of the International Society for Telemedicine and eHealth**, v. 5, p. GKR–e24, 2017.

GHORI, M. R.; WAN, T.-C.; ANBAR, M.; SODHY, G. C.; RIZWAN, A. Review on security in bluetooth low energy mesh network in correlation with wireless mesh network security. In: IEEE. **2019 IEEE Student Conference on Research and Development (SCOReD)**. Perak, Malaysia, 2019. p. 219–224.

GIA, T. N.; JIANG, M.; SARKER, V. K.; RAHMANI, A. M.; WESTERLUND, T.; LILJEBERG, P.; TENHUNEN, H. Low-cost fog-assisted health-care iot system with energy-efficient sensor nodes. In: **2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)**. Valencia, Spain: IEEE, 2017. p. 1765–1770.

GOLDREICH, O. **Foundations of Cryptography: Volume 1, Basic Tools**. Cambridge, UK: Cambridge University Press, 2007. ISBN 9781139430234.

GOLDREICH, O. **Computational Complexity: A Conceptual Perspective**. New York, United States: Cambridge University Press, 2008. ISBN 9781139472746.

GOLDWASSER, S.; MICALI, S.; RACKOFF, C. The knowledge complexity of interactive proof systems. **SIAM Journal on computing**, SIAM, v. 18, n. 1, p. 186–208, 1989.

GOYAL, V.; PANDEY, O.; SAHAI, A.; WATERS, B. Attribute-based encryption for fine-grained access control of encrypted data. In: ACM. **Proceedings of the 13th ACM conference on Computer and communications security**. Virginia, USA, 2006. p. 89–98.

GUO, F.; MU, Y.; SUSILO, W.; WONG, D. S.; VARADHARAJAN, V. Cp-abe with constant-size keys for lightweight devices. **IEEE transactions on information forensics and security**, IEEE, v. 9, n. 5, p. 763–771, 2014.

HAMMI, M. T.; HAMMI, B.; BELLOT, P.; SERHROUCHNI, A. Bubbles of trust: A decentralized blockchain-based authentication system for iot. **Computers & Security**, Elsevier, v. 78, p. 126–142, 2018.

HAO, F. **Schnorr Non-interactive Zero-Knowledge Proof**. RFC8235, 2017. Available at: https://tools.ietf.org/html/rfc8235. Accessed on: 14.02.2021.

HASSAN, M. U.; REHMANI, M. H.; CHEN, J. Privacy preservation in blockchain based iot systems: Integration issues, prospects, challenges, and future research directions. **Future Generation Computer Systems**, v. 97, p. 512 – 529, 2019. ISSN 0167-739X.

HATHALIYA, J. J.; TANWAR, S. An exhaustive survey on security and privacy issues in healthcare 4.0. **Computer Communications**, v. 153, p. 311 – 335, 2020. ISSN 0140-3664.

HOUTAN, B.; HAFID, A. S.; MAKRAKIS, D. A survey on blockchain-based self-sovereign patient identity in healthcare. **IEEE Access**, IEEE, v. 8, p. 90478–90494, 2020.

HSS. **The HIPAA Privacy Rule**. U.S. Department of Health & Human Services, 2020. Available at: https://www.hhs.gov/hipaa/for-professionals/privacy/index.html. Accessed on: 23.12.2020.

HUANG, C.; YAN, K.; WEI, S.; LEE, D. H. A privacy-preserving data sharing solution for mobile healthcare. In: IEEE. **2017 International Conference on Progress in Informatics and Computing (PIC)**. Nanjing, China, 2017. p. 260–265.

IOVINO, V.; VISCONTI, I. Non-interactive zero knowledge proofs in the random oracle model. In: SPRINGER. **International Conference on Codes, Cryptology, and Information Security**. Switzerland, 2019. p. 118–141.

JESUS, E. F.; CHICARINO, V. R.; ALBUQUERQUE, C. V. de; ROCHA, A. A. d. A. A survey of how to use blockchain to secure internet of things and the stalker attack. **Security and Communication Networks**, Hindawi, v. 2018, 2018.

JIANG, W.; LI, H.; XU, G.; WEN, M.; DONG, G.; LIN, X. Ptas: Privacy-preserving thin-client authentication scheme in blockchain-based pki. **Future Generation Computer Systems**, v. 96, p. 185 – 195, 2019. ISSN 0167-739X.

KALAI, Y. T.; ROTHBLUM, G. N.; ROTHBLUM, R. D. From obfuscation to the security of fiat-shamir for proofs. In: **Advances in Cryptology – CRYPTO 2017**. Santa Barbara, CA, USA: Springer International Publishing, 2017. p. 224–251. ISBN 978-3-319-63715-0.

KIM, S.; DEKA, G.; ZHANG, P. **Role of Blockchain Technology in IoT Applications**. United States: Elsevier Science & Technology, 2019. (Advances in Computers Series). ISBN 9780128171899.

KING, S.; NADAL, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. **self-published paper, August**, v. 19, p. 1, 2012.

KOBLITZ, N. Elliptic curve cryptosystems. **Mathematics of computation**, v. 48, n. 177, p. 203–209, 1987.

KOTZ, D.; GUNTER, C. A.; KUMAR, S.; WEINER, J. P. Privacy and security in mobile health: a research agenda. **Computer**, IEEE, v. 49, n. 6, p. 22–30, 2016.

KUMAR, A.; KUMAR, R. Privacy preservation of electronic health record: Current status and future direction. In: **Handbook of Computer Networks and Cyber Security**. Cham, Switzerland: Springer, 2020. p. 715–739.

LATRÉ, B.; BRAEM, B.; MOERMAN, I.; BLONDIA, C.; DEMEESTER, P. A survey on wireless body area networks. **Wireless Networks**, Springer-Verlag New York, Inc., v. 17, n. 1, p. 1–18, 2011.

LE, X. H.; KHALID, M.; SANKAR, R.; LEE, S. An efficient mutual authentication and access control scheme for wireless sensor networks in healthcare. **Journal of Networks**, Academy Publisher, v. 6, n. 3, p. 355–364, 2011.

LEE, G. **Abstract Algebra: An Introductory Course**. Canada: Springer International Publishing, 2018. (Springer Undergraduate Mathematics Series). ISBN 9783319776491.

LI, F.; HONG, J.; OMALA, A. A. Efficient certificateless access control for industrial internet of things. **Future Generation Computer Systems**, Elsevier, v. 76, p. 285–292, 2017.

LI, Q.; ZHU, H.; XIONG, J.; MO, R.; YING, Z.; WANG, H. Fine-grained multi-authority access control in iot-enabled mhealth. **Annals of Telecommunications**, Springer, v. 74, n. 7-8, p. 389–400, 2019.

LI, X.; IBRAHIM, M. H.; KUMARI, S.; SANGAIAH, A. K.; GUPTA, V.; CHOO, K.-K. R. Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks. **Computer Networks**, Elsevier, v. 129, p. 429–443, 2017.

LI, X.; JIANG, P.; CHEN, T.; LUO, X.; WEN, Q. A survey on the security of blockchain systems. **Future Generation Computer Systems**, aug 2017.

LIANG, X.; ZHAO, J.; SHETTY, S.; LIU, J.; LI, D. Integrating blockchain for data sharing and collaboration in mobile healthcare applications. In: **2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)**. Montreal, QC, Canada: [s.n.], 2017. p. 1–5.

LINZ, P. **An Introduction to Formal Languages and Automata**. United States: Jones and Bartlett Publishers, 2006. ISBN 9780763737986.

LIU, S.; HU, S.; WENG, J.; ZHU, S.; CHEN, Z. A novel asymmetric three-party based authentication scheme in wearable devices environment. **Journal of Network and Computer Applications**, Elsevier, v. 60, p. 144–154, 2016.

LIU, W.; LIU, H.; WAN, Y.; KONG, H.; NING, H. The yoking-proof-based authentication protocol for cloud-assisted wearable devices. **Personal and Ubiquitous Computing**, Springer, v. 20, n. 3, p. 469–479, 2016.

LUNARDI, R. C.; MICHELIN, R. A.; NEU, C. V.; ZORZO, A. F. Distributed access control on iot ledger-based architecture. In: IEEE. **NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium**. Taipei, Taiwan, 2018. p. 1–7.

MA, C.; XUE, K.; HONG, P. Distributed access control with adaptive privacy preserving property for wireless sensor networks. **Security and Communication Networks**, Wiley Online Library, v. 7, n. 4, p. 759–773, 2014.

MACKAY, K. **micro-ecc**. Arduino Libraries, 2017. Available at: https://www.arduinolibraries.info/libraries/micro-ecc. Accessed on: 05.06.2019.

MCGHIN, T.; CHOO, K.-K. R.; LIU, C. Z.; HE, D. Blockchain in healthcare applications: Research challenges and opportunities. **Journal of Network and Computer Applications**, Elsevier, v. 135, p. 62–75, 2019.

MCGHIN, T.; CHOO, K.-K. R.; LIU, C. Z.; HE, D. Blockchain in healthcare applications: Research challenges and opportunities. **Journal of Network and Computer Applications**, Elsevier, v. 135, p. 62–75, 2019.

MENEZES, A.; OORSCHOT, P. van; VANSTONE, S. **Handbook of Applied Cryptography**. Boca Raton, Florida, USA: CRC Press, 2018. (Discrete Mathematics and Its Applications). ISBN 9780429881329.

MERKLE, R. C. A digital signature based on a conventional encryption function. In: POMERANCE, C. (Ed.). **Advances in Cryptology — CRYPTO '87**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. p. 369–378. ISBN 978-3-540-48184-3.

MERKLE, R. C. One way hash functions and des. In: BRASSARD, G. (Ed.). **Advances in Cryptology — CRYPTO' 89 Proceedings**. New York, NY: Springer New York, 1990. p. 428–446. ISBN 978-0-387-34805-6.

Microchip. **Wearable Heart Rate Monitor**. 2020. Available at: https://www.microchip.com/design-centers/medical/applications/wearable-activity-monitors/design-files-demo-boards/wearable-heart-rate-monitor-demo.

MILLER, V. S. Use of elliptic curves in cryptography. In: **Advances in Cryptology — CRYPTO '85 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986. p. 417–426. ISBN 978-3-540-39799-1.

MOOSAVI, S. R.; GIA, T. N.; NIGUSSIE, E.; RAHMANI, A. M.; VIRTANEN, S.; TENHUNEN, H.; ISOAHO, J. End-to-end security scheme for mobility enabled healthcare internet of things. **Future Generation Computer Systems**, Elsevier, v. 64, p. 108–124, 2016.

MOOSAVI, S. R.; GIA, T. N.; RAHMANI, A.-M.; NIGUSSIE, E.; VIRTANEN, S.; ISOAHO, J.; TENHUNEN, H. Sea: A secure and efficient authentication and authorization architecture for iot-based healthcare using smart gateways. **Procedia Computer Science**, v. 52, p. 452 – 459, 2015. ISSN 1877-0509.

MUTLAG, A. A.; Abd Ghani, M. K.; ARUNKUMAR, N.; MOHAMMED, M. A.; MOHD, O. Enabling technologies for fog computing in healthcare iot systems. **Future Generation Computer Systems**, v. 90, p. 62 – 78, 2019. ISSN 0167-739X.

NAKAMOTO, S. **Bitcoin: A peer-to-peer electronic cash system**. Working Paper, 2008. Available at: https://bitcoin.org/bitcoin.pdf. Accessed on: 16.01.2020.

NAVEED, M.; ZHOU, X.; DEMETRIOU, S.; WANG, X.; GUNTER, C. A. Inside Job: Understanding and Mitigating the Threat of External Device Mis-Bonding on Android. In: INTERNET SOCIETY. **Network and Distributed System Security Symposium**. San Diego, California, USA, 2014. ISBN 1891562355.

ODELU, V.; DAS, A. K.; RAO, Y. S.; KUMARI, S.; KHAN, M. K.; CHOO, K.-K. R. Pairing-based cp-abe with constant-size ciphertexts and secret keys for cloud environment. **Computer Standards & Interfaces**, Elsevier, v. 54, p. 3–9, 2017.

PAAR, C.; PELZL, J. **Understanding Cryptography: A Textbook for Students and Practitioners**. Germany: Springer Berlin Heidelberg, 2009. ISBN 9783642041013.

RAHULAMATHAVAN, Y.; PHAN, R. C. .; RAJARAJAN, M.; MISRA, S.; KONDOZ, A. Privacy-preserving blockchain based iot ecosystem using attribute-based encryption. In: **2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)**. Bhubaneswar, Odisha, India: IEEE, 2017. p. 1–6.

RAMLI, S. N.; AHMAD, R.; ABDOLLAH, M. F.; DUTKIEWICZ, E. A biometric-based security for data authentication in wireless body area network (wban). In: IEEE. **2013 15th International Conference on Advanced Communications Technology (ICACT)**. PyeongChang, South Korea, 2013. p. 998–1001.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. **Commun. ACM**, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, Feb. 1978.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. **Commun. ACM**, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, Feb. 1978.

ROEHRS, A.; COSTA, C. A. da; RIGHI, R. da R. Omniphr: A distributed architecture model to integrate personal health records. **Journal of biomedical informatics**, Elsevier, v. 71, p. 70–81, 2017.

ROSEN, A. **Concurrent Zero-Knowledge: With Additional Background by Oded Goldreich**. New York, United States: Springer Berlin Heidelberg, 2007. (Information Security and Cryptography). ISBN 9783540329398.

RUBINSTEIN-SALZEDO, S. **Cryptography**. Cham, Switzerland: Springer International Publishing, 2018. (Springer Undergraduate Mathematics Series). ISBN 9783319948188.

SAHAI, A.; WATERS, B. Fuzzy identity-based encryption. In: **Advances in Cryptology – EUROCRYPT 2005**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 457–473.

SAITO, K.; IWAMURA, M. How to make a digital currency on a blockchain stable. **Future Generation Computer Systems**, v. 100, p. 58–69, 2019. ISSN 0167-739X.

SANTIS, A. D.; MICALI, S.; PERSIANO, G. Non-interactive zero-knowledge with preprocessing. In: **Advances in Cryptology — CRYPTO' 88**. New York, NY: Springer New York, 1990. p. 269–282. ISBN 978-0-387-34799-8.

SATHYA, D.; KUMAR, P. G. Secured remote health monitoring system. **Healthcare Technology Letters**, v. 4, n. 6, p. 228–232, 2017.

SCHNORR, C.-P. Efficient signature generation by smart cards. **Journal of cryptology**, Springer, v. 4, n. 3, p. 161–174, 1991.

SCHOEMAN, F. D. **Philosophical dimensions of privacy: An anthology**. New York, United States: Cambridge University Press, 1984.

SECP256K1. **Bitcoin Wiki**. Bitcoin Wiki, 2019. Available at: https://en.bitcoin.it/wiki/Secp256k1. Accessed on: 05.03.2020.

SEN, J. **Cryptography and Security in Computing**. Rijeka, Croatia: IntechOpen, 2012. ISBN 9789535101796.

SHEN, B.; GUO, J.; YANG, Y. Medchain: efficient healthcare data sharing via blockchain. **Applied sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 6, p. 1207, 2019.

SHIM, K.-A. S2drp: Secure implementations of distributed reprogramming protocol for wireless sensor networks. **Ad Hoc Networks**, Elsevier, v. 19, p. 1–8, 2014.

SHRIVASTAVA, G.; LE, D.; SHARMA, K. **Cryptocurrencies and Blockchain Technology Applications**. Beverly, MA, USA: Wiley, 2020. ISBN 9781119621164.

SILVA, B. M.; RODRIGUES, J. J.; CANELO, F.; LOPES, I. M.; LLORET, J. Towards a cooperative security system for mobile-health applications. **Electronic Commerce Research**, Springer, v. 19, n. 3, p. 629–654, 2019.

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. Building an ethereum dapp. In: **Beginning Blockchain**. Bangalore, India: Springer, 2018. p. 319–375.

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. How bitcoin works. In: **Beginning Blockchain**. Bangalore, India: Springer, 2018. p. 149–217.

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. How blockchain works. In: **Beginning Blockchain**. Bangalore, India: Springer, 2018. p. 31–148.

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. How ethereum works. In: **Beginning Blockchain**. Bangalore, India: Springer, 2018. p. 219–266.

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. Introduction to blockchain. In: **Beginning Blockchain**. Bangalore, India: Springer, 2018. p. 1–29.

SIPSER, M. **Introduction to the Theory of Computation**. Boston, USA: Cengage Learning, 2012. ISBN 9781285401065.

SMART, N. **Cryptography Made Simple**. Switzerland: Springer International Publishing, 2015. (Information Security and Cryptography). ISBN 9783319219363.

SONG, Y.; WANG, H.; WEI, X.; WU, L. Efficient attribute-based encryption with privacy-preserving key generation and its application in industrial cloud. **Security and Communication Networks**, Hindawi, v. 2019, 2019.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice, International Edition: Principles and Practice**. 6. ed. United States: Pearson Education Limited, 2014.

Standards for Efficient Cryptography Group. **Recommended Elliptic Curve Domain Parameters**. Standards for Efficient Cryptography Group, 2010. Available at: https://www.secg.org/sec2-v2.pdf. Accessed on: 05.06.2019.

TAPSCOTT, D.; TAPSCOTT, A. **Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World**. New York, USA: Penguin Publishing Group, 2016. ISBN 9781101980156.

Texas Instruments. **Blood pressure monitor**. 2020. Available at: https://www.ti.com/solution/blood-pressure-monitor?variantid=33745&subsystemid=14125.

Texas Instruments. **Electronic thermometer**. 2020. Available at: https://www.ti.com/solution/electronic-thermometer?variantid=34261&subsystemid=25456. Accessed on: 20.01.2020.

THWIN, T. T.; VASUPONGAYYA, S. Blockchain-Based Access Control Model to Preserve Privacy for Personal Health Record Systems. **Security and Communication Networks**, v. 2019, p. 1–15, jun 2019. ISSN 1939-0114.

TOMAZ, A. E. B.; NASCIMENTO, J. C. D.; HAFID, A. S.; SOUZA, J. N. D. Preserving privacy in mobile health systems using non-interactive zero-knowledge proof and blockchain. **IEEE Access**, v. 8, p. 204441–204458, 2020.

TSCHORSCH, F.; SCHEUERMANN, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. **IEEE Communications Surveys & Tutorials**, IEEE, v. 18, n. 3, p. 2084–2123, 2016.

TURING, A. M. On computable numbers, with an application to the entscheidungsproblem. **Proceedings of the London Mathematical Society**, s2-42, n. 1, p. 230–265, 1936.

UHER, J.; MENNECKE, R. G.; FARROHA, B. S. Denial of sleep attacks in bluetooth low energy wireless sensor networks. In: IEEE. **MILCOM 2016-2016 IEEE Military Communications Conference**. Baltimore, USA, 2016. p. 1231–1236.

VITHANWATTANA, N.; MAPP, G.; GEORGE, C. mhealth-investigating an information security framework for mhealth data: Challenges and possible solutions. In: IEEE. **2016 12th International Conference on Intelligent Environments (IE)**. London, UK, 2016. p. 258–261.

VORA, J.; DEVMURARI, P.; TANWAR, S.; TYAGI, S.; KUMAR, N.; OBAIDAT, M. S. Blind signatures based secured e-healthcare system. In: **2018 International Conference on Computer, Information and Telecommunication Systems (CITS)**. Alsace, France: IEEE, 2018.

WANG, H.; SHENG, B.; LI, Q. Elliptic curve cryptography-based access control in sensor networks. **International Journal of Security and Networks**, Inderscience Publishers, v. 1, n. 3-4, p. 127–137, 2006.

WANG, J. **Java Realization for Ciphertext-Policy Attribute-Based Encryption**. Github, 2012. Available at: https://github.com/junwei-wang/cpabe/.

WANG, S.; ZHANG, Y.; ZHANG, Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. **Ieee Access**, IEEE, v. 6, p. 38437–38450, 2018.

WANG, W.; HOANG, D. T.; HU, P.; XIONG, Z.; NIYATO, D.; WANG, P.; WEN, Y.; KIM, D. I. A survey on consensus mechanisms and mining strategy management in blockchain networks. **IEEE Access**, IEEE, v. 7, p. 22328–22370, 2019.

WAZID, M.; ZEADALLY, S.; DAS, A. K.; ODELU, V. Analysis of security protocols for mobile healthcare. **Journal of medical systems**, Springer, v. 40, n. 11, p. 229, 2016.

WEE, H. Zero knowledge in the random oracle model, revisited. In: **Advances in Cryptology – ASIACRYPT 2009**. Berlin, Heidelberg: Springer, 2009. p. 417–434. ISBN 978-3-642-10366-7.

WERBACH, K. **The Blockchain and the New Architecture of Trust**. United States: MIT Press, 2018. (Information Policy). ISBN 9780262038935.

WOOD, G. *et al.* Ethereum: A secure decentralised generalised transaction ledger. **Ethereum project yellow paper**, v. 151, n. 2014, p. 1–32, 2014.

XU, X.; PAUTASSO, C.; ZHU, L.; GRAMOLI, V.; PONOMAREV, A.; TRAN, A. B.; CHEN, S. The blockchain as a software connector. In: IEEE. **2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)**. Venice, Italy, 2016. p. 182–191.

YAN, S. Y. Computational/mathematical preliminaries. In: ____. **Cryptanalytic Attacks on RSA**. Boston, MA: Springer US, 2008. p. 1–54. ISBN 978-0-387-48742-7.

YÁNEZ, W.; MAHMUD, R.; BAHSOON, R.; ZHANG, Y.; BUYYA, R. Data allocation mechanism for internet-of-things systems with blockchain. **IEEE Internet of Things Journal**, IEEE, v. 7, n. 4, p. 3509–3522, 2020.

YANG, L.; ZHENG, Q.; FAN, X. Rspp: A reliable, searchable and privacy-preserving e-healthcare system for cloud-assisted body area networks. In: **IEEE INFOCOM 2017 - IEEE Conference on Computer Communications**. Atlanta, USA: IEEE, 2017. p. 1–9.

YUE, X.; WANG, H.; JIN, D.; LI, M.; JIANG, W. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. **Journal of medical systems**, Springer, v. 40, n. 10, p. 218, 2016.

ZHANG, J.; WANG, X. A.; MA, J. Data owner based attribute based encryption. In: **2015 International Conference on Intelligent Networking and Collaborative Systems**. Taipei, Taiwan: IEEE, 2015. p. 144–148.

ZHANG, K.; YANG, K.; LIANG, X.; SU, Z.; SHEN, X.; LUO, H. H. Security and privacy for mobile healthcare networks: from a quality of protection perspective. **IEEE Wireless Communications**, v. 22, n. 4, p. 104–112, 2015.

ZHANG, R.; XUE, R.; LIU, L. Security and privacy on blockchain. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 3, Jul. 2019. ISSN 0360-0300.

ZHENG, Z.; XIE, S.; DAI, H.; CHEN, X.; WANG, H. An overview of blockchain technology: Architecture, consensus, and future trends. In: IEEE. **2017 IEEE international congress on big data (BigData congress)**. Honolulu, USA, 2017. p. 557–564.

ZUBAYDI, F.; SALEH, A.; ALOUL, F.; SAGAHYROON, A. Security of mobile health (mhealth) systems. In: IEEE. **2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE)**. Belgrade, Serbia, 2015. p. 1–5.