TÉCNICO LISBOA

Addressing Disinformation With Open-Ended Internet Voting Using Ring Signatures

Pedro Miguel Forjaz de Sampaio Narra de Figueiredo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Pedro Miguel dos Santos Alves Madeira Adão
Eng. Nelson Nobre Escravana

Examination Committee

Chairperson: Prof. Francisco António Chaves Saraiva de Melo
Supervisor: Prof. Pedro Miguel dos Santos Alves Madeira Adão
Member of the Committee: Prof. Carlos Nuno da Cruz Ribeiro

January 2021

Acknowledgements

First, I would like to thank Eng. Nelson for the unconditional support, smart and relevant tips and advice and for always leading me in the right way, particularly when I needed guidance. I would also like to express my gratitude to Prof. Pedro for sharing his expertise, which was essential throughout the development of this work, and for raising important issues which lead to a significantly better dissertation.

Additionally, I would like to express my appreciation for everyone at INOV, in particular to João, whose suggestions were remarkably helpful and, without them, this work would have been notably harder to develop.

Finally, and as important as everything and everyone else, I would like to thank my father, Pedro, and my mother, Paula, for the encouragement to always do better, and my sister, Teresa, who endlessly believed in me. Furthermore, to all my friends, with whom I could share a lot of moments of joy and fun.

To each and every one of you, my most sincere Thank You.

Resumo

O problema da desinformação tem criado preocupações relacionadas com fiabilidade do conteúdo em redes sociais. As chamadas *fake news*, ou notícias falsas, espalham-se rapidamente e enganam os utilizadores. Como tal, conclui-se facilmente que é necessário desenvolver uma solução que ajuda a avaliar a confiabilidade da informação que é publicamente distribuída. Nesta dissertação, é proposto um sistema de votação eletrónica totalmente descentralizado, que permite a votação anónima em características de confiabilidade em informação publicada em redes sociais. Nas últimas décadas, muito trabalho tem vindo a ser direcionado para o desenho de soluções de votação eletrónica que promovem privacidade e integridade. Contudo, tanto quanto sabemos, nenhuma destas soluções aborda votação aberta (sem fim definido) e descentralizada pela internet, com contagem de votos contínua e garantias de privacidade. A nossa proposta utiliza assinaturas em grupo e pseudónimos para garantir elegibilidade sem revelar a identidade do votante, e implementa *mix networks* para assegurar anonimato. Esta solução suporta vários processos de votação simultâneos, com boletins de votos flexíveis, e apenas necessita de uma volta de comunicação na fase de votação. Para além disto, também proporciona outras propriedades desejáveis em votação eletrónica, tais como precisão e verificação.

Palavras-chave: Votação Eletrónica, Notícias Falsas, Segurança, Privacidade, Confiança

Abstract

The problem of disinformation has raised concerns regarding the reliability of content on social media. The so-called fake news spread rapidly and mislead people. Therefore, it is clear how paramount it is to develop a solution which helps to assess the credibility of information that is publicly distributed. In this dissertation, a fully decentralized voting system is proposed, which allows people to vote anonymously on trustworthiness characteristics of posts on social media. In the past decades, significant effort has been directed into designing electronic voting solutions which promote privacy and integrity. Nevertheless, to the best of our knowledge, none of these solutions approach open-ended, continuous-tallying decentralized internet voting with privacy guarantees. The proposed scheme relies on ring signatures and pseudonyms to grant eligibility without revealing the identity of the voter, and implements mix networks to achieve voter anonymity. It supports multiple concurrent voting processes with flexible ballots, and only requires one round for the voting phase. Additionally, this proposal provides other desired voting properties, such as accuracy and verifiability.

Keywords: Electronic Voting, Fake News, Privacy, Security, Trustworthiness

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Voting is a foundational principle of democracy. It emerged as a way of expressing disapproval. Specifically, back in Ancient Greece, votes were registered on pieces of porcelain, which determined whether politicians were to be exiled. It was not until the 1800s that paper voting was introduced. A few decades later, Thomas Edison was granted a patent for one of his first inventions: an "electronic voting device", which was meant to be used by the United States Congress. However, this vote recorder was never implemented and paper ballots have remained around ever since.

In the 1980s, new efforts started aiming at developing alternatives to the conventional paper ballots and physical voting, which revived the notion of electronic voting. Electronic voting (or *e-voting*) is a broad concept. Generally, any voting process which involves a machine to cast ballots or count votes is electronic. This includes voting through the internet (also called *i-voting*), which can be done anywhere, and voting in the presence of authorities, using specific devices. Examples of these devices are scanners, which read paper ballots, and buttons or a touchscreen to vote.

Since that decade, an extensive amount of electronic voting protocols has unfolded and changed drastically the voting process by lowering costs, tallying faster and providing more flexibility to the voters. In fact, electronic voting may nowadays be applied efficiently to the majority of situations, ranging from small electoral rolls to country-sized ones [77].

In this work, we use voting to address the problem of disinformation. The so-called *fake news* is distorted or untrue information presented to readers or viewers. Relevantly, due to the growth of computer-mediated communication, this phenomenon has proliferated, which has raised some concerns regarding the trustworthiness of content on social media. The most significant event in which this problem gained public awareness was in the 2016 United States Presidential Election [10]. This led not only to scandals related with the impact on the results of the election, but also to privacy concerns inherent to the collection of user data on the internet.

The next two sections explain how electronic voting can be demanding to implement and how it can be used to support the evaluation of trustworthiness of information on social media.

This work was developed at INOV - Instituto de Engenharia de Sistemas e Computadores Inovação under the supervision of Eng. Nelson Escravana, in collaboration with Instituto Superior Técnico under the advisory of Prof. Pedro Adão. It is integrated in the ongoing EUNOMIA Project [2], supported by the European Union H2020 Research and Innovation Programme, with Grant Agreement number 825171.

1.1 Motivation

Electronic voting can be challenging. Some trust assumptions, such as untappable channels[1], are unrealistic for fully electronic voting, the reliability of each component of a system must be disputed and earning the trust of a voter is hard. Additionally, it has many requirements that a system should comply with, which can even sometimes contradict themselves.

To put things into perspective, we compare electronic voting with banking, as did Adida [7]. Banks process millions of transactions daily, recording all the money that goes in and out of every account, and keeping the receipts which customers can later use to audit their accounts. Similarly, one should be able to let voters cast millions of votes, save them and later count them.

This analogy, however, faces two substantial misconceptions [7]. The first one is that, unlike in banking, where there is a trusted administration, any entity participating in a voting process, including authorities, can be dishonest. The second one relates to failure detection and recovery, which is well defined in banking since there are records of every operation. In contrast, voting failures can go undetected as some records are often purposely erased at the expense of privacy.

In fact, for this analogy to be accurate, some requirements have to be added. First, the bank cannot know the balances of the customers. Second, a customer cannot prove their balance to anyone, while still being sure that the money is safe. These new restrictions clarify the difficulty of designing a voting system.

We shift our attention to fake news. This is a very delicate affair in the sense that fighting it is extremely difficult, which is due to a few reasons. First, content on the internet spreads rapidly and almost uncontrollably. Second, social networking platforms present users with specifically targeted information and advertisements. That is, it is not a single problem, instead every user is affected differently. Finally, and most pertinent, users eventually end up misinterpreting the information or being persuaded into changing their beliefs or behavior.

Taking these reasons into account, it is clear how paramount it is to develop a solution which helps users to assess the reliability of information that is publicly distributed.

The EUNOMIA Project [2] focuses on fighting disinformation by providing social media users with information cascades[2], sentiment analysis and trustworthiness indicators for posts on social networks. It introduces a concept called *Human-as-Trust-Sensor*, which places the user at the center of the reliability evaluation process. Along these lines, users are helped to identify the provenance of information and to understand how information has been modified and spread. Moreover, they are able to attribute trustworthiness properties to that data.

EUNOMIA endorses a design philosophy which fosters openness and transparency. It is a *fully*

[1]A channel can be considered *untappable* if it uses a one-time pad to hide data because one-time pads are mathematically unbreakable, thus providing perfect secrecy [70].

[2]An information cascade tracks changes made to data since its original source until it is presented to the user.

decentralized peer-to-peer platform, delegating the decision-making process and overall system management to a set of independent nodes. In addition, it is a *distributed system*, meaning that there is no single entity responsible for doing all the work, instead tasks are carried out by a set of nodes working together in a distributed manner.

Since EUNOMIA nodes are operated by independent and unrelated entities, trust is a feature of the overall system as we cannot guarantee that each of the entities is not malicious. The security model states that the system as a whole can be trusted, as long as a subset of nodes remains honest. Furthermore, storage in EUNOMIA is supported by a *potentially hostile* peer-to-peer network, which can be accessed by the nodes, providing them with a highly distributed, decentralized storage solution. Finally, all information that circulates outside of each node is presumed to be in a malicious environment and should be adequately protected.

1.2 Contributions

This work focuses solely on the third part of the EUNOMIA Project, which is providing trustworthiness indicators for online information. We propose a voting system which entitles users to assign trust properties to social media posts, and choose whether their set of attributes, such as followers or political ideology, is shown in the tally, if they wish to have their identity shown, or if they would like to stay anonymous.

This approach focuses on gathering people's input rather than delegating the task of information classification to a third party. Still, the system is only responsible for showing the users the voting tally, so it is up to them to make the decision of whether the information is trustworthy or not.

Currently, while there are no voting approaches tailored for this goal, many voting implementations exist, which feature the most varied properties. All of these differ mainly on the methods used to achieve ballot anonymization and validation, how voter registration is carried out (if it exists) and how tallying is performed. Consequently, the cost and efficiency of every system depend on the underlying properties and objectives of each one.

The *voting server*, integrated in each of the EUNOMIA nodes, and thus respecting the underlying design requirements, is responsible for handling ballots, anonymizing them (if necessary) and counting votes. All relevant public information is posted to a storage service, supported by the previously mentioned peer-to-peer storage network.

Considering the previous section, we emphasize how challenging it is to design a voting system under the security model of EUNOMIA. This is because the system is fully decentralized and distributed, and has no trusted third party, meaning that no entity can be trusted individually.

Furthermore, regarding the voting component, some more challenges arise. First, voting processes are open-ended, which raises many privacy complications. Second, the system must support multiple simultaneous voting processes. Third, the system has to avoid double voting without knowing if users have voted before. And fourth, ballots should be totally flexible and adaptable to different posts.

To summarize, the main contribution of this work is the design and implementation of a voting scheme which, besides tackling these challenges, promotes the following properties:

- **Accuracy**: All valid votes are counted (completeness) and all invalid votes are not counted (soundness).
- **Privacy**: Choices of the voters are secret.
- **Uniqueness**: No voter can cast a ballot twice on the same post.
- **Individual Verifiability**: Voters can verify if their cast ballot represents correctly their voting intentions (cast as intended) and that it has been properly submitted (recorded as cast).
- **Universal Verifiability**: Ability to confirm that all stored valid votes have been included in the tally (counted as recorded) and that only eligible voters voted (eligibility check).
- **Robustness**: Resistance to partial failures, to malicious behavior by the authorities or the voters and to some level of collusion.
- **Transparency**: All information relevant to the voting protocol is publicly available and able to be verified.

These properties make up the adequate voting solution for the designated purpose, offering anonymity and integrity.

1.3 Dissertation Outline

The remainder of the dissertation is outlined as follows:

- **Chapter 2** provides a background on the concepts presented throughout the work.
- **Chapter 3** develops on the main ideas of electronic voting and explains the evolution of the work attained in the area so far.
- **Chapter 4** proposes an electronic voting solution for the given problem.
- **Chapter 5** explains how the solution was implemented.
- **Chapter 6** reports the evaluation of the proposed solution.
- **Chapter 7** concludes this dissertation.

Chapter 2

Background

In this chapter, we present the background which is the foundation for the remainder of the topics addressed in this work. It is assumed the reader already has some knowledge of mathematics and public key cryptography.

We start by defining groups in Section 2.1, followed by cryptographic assumptions in Section 2.2. Next, elliptic curves are tackled in Section 2.3. Finally, Section 2.4 provides an overview of zero knowledge proofs.

2.1 Groups

This section introduces the notion of group. The group is the starting point for many of the concepts which are presented later in this chapter. We refer to Katz and Lindell [70] for a more detailed explanation.

Definition 2.1.1. A *group* is a pair ($\mathbb{G}$, $\circ$), which defines $\mathbb{G}$ as a set and $\circ$ as binary operation for which the subsequent conditions hold:

- **Closure**: For all $g, h \in \mathbb{G}$, $g \circ h \in \mathbb{G}$.
- **Existence of an identity**: There exists an element $e \in \mathbb{G}$, called *identity*, such that for all $g \in \mathbb{G}$, $e \circ g = g \circ e = g$.
- **Existence of inverses**: There exists an element $h \in \mathbb{G}$, called *inverse*, such that for all $g \in \mathbb{G}$, $h \circ g = g \circ h = e$.
- **Associativity**: For all $a, b, c \in \mathbb{G}$, $(a \circ b) \circ c = a \circ (b \circ c)$.

Moreover, a group is *abelian* if the following additional property holds:

- **Commutativity**: For all $g, h \in \mathbb{G}$, $g \circ h = h \circ g$.

For a question of simplicity, and when the binary operation $\circ$ is well understood, we refer to the group simply as $\mathbb{G}$. Also, the *order* (number of elements) of a group $\mathbb{G}$ is denoted by $|\mathbb{G}|$ and $\mathbb{G}$ is called *finite* when it has a finite number of elements.

Definition 2.1.2. A group $\mathbb{G}$ of order p is *cyclic* if there exists $g \in \mathbb{G}$ such that all elements of $\mathbb{G}$ can be written as g^n, for any $n \in \mathbb{Z}$. The element g is then called the *generator* and group $\mathbb{G}$ can be specified as $\mathbb{G} = \langle g \rangle$.

Groups can be of *prime order* p, which are the only ones employed throughout this dissertation. One intelligible example of a prime order group is $\mathbb{Z}_p^*$, which corresponds to all positive integers *modulo* p.

2.1.1 Fields

We now briefly define fields. Informally, a field can be specified as two abelian groups joined together, with each of them supporting a distinct binary operation. The definition below formalizes the properties of a field.

Definition 2.1.3. A *field* $\mathbb{F}$ is a set which defines two binary operations, usually called $+$ and $\cdot$, respectively *addition* and *multiplication*, for which the following conditions hold:

- $\mathbb{F}$ is an abelian group which designates $+$ as its binary operation and has *identity* element 0.
- $\mathbb{F} \setminus \{0\}$ is an abelian group which designates $\cdot$ as its binary operation and has *identity* element 1.
- **Distributivity**: For all $a, b, c \in \mathbb{F}$, $a \cdot (b + c) = a \cdot b + a \cdot c$.

The order q of a field $\mathbb{F}$ can be p^k, where p is a prime and $k \in \mathbb{N}$. In this particular case, we say that field $\mathbb{F}$ has *characteristic* p, it is called *finite* field or *Galois* field and represented by $\mathbb{F}_q$ or $\mathbb{F}_{p^k}$. Once again, $\mathbb{Z}_p^*$ can be an example of a prime field.

2.2 Cryptographic Assumptions

Computational hardness concerns the inability of a computer to solve a mathematical problem in reasonable time, and cryptography usually relies on the intractability of these problems to build provably secure systems. The assumptions rely on these problems because it is assumed that they cannot be broken efficiently, or in *polynomial time*, with the existing computing power and algorithms. Therefore, systems are secure as long as the assumptions hold.

In this section, we present the cryptographic assumptions upon which some of the propositions of this work fall back on. We start by defining *negligible* functions, which serve as basis for the subsequent definitions. We refer to Katz and Lindell [70] for more details.

Definition 2.2.1. A function $f : \mathbb{N} \rightarrow \mathbb{R}_0^+$ is *negligible* if, for every positive polynomial p, the condition $f(n) < \frac{1}{p(n)}$ holds for a sufficiently large n.

Definition 2.2.2. The *discrete logarithm* (DL) problem states that, given a cyclic group $\mathbb{G} = \langle g \rangle$ of order q and $g, h \in \mathbb{G}$, finding $x \in \mathbb{Z}_q$ such that $g^x = h$ is *hard*.

We now consider a group generation algorithm $\mathcal{G}$, which accepts a security parameter n and generates a tuple $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q and g is the generator of group $\mathbb{G}$. Also, we define an algorithm $\mathcal{A}$, which receives the tuple $(\mathbb{G}, q, g, h)$ and outputs $x \in \mathbb{Z}_q$ such that $g^x = h$.

Definition 2.2.3. The DL problem is *hard* relative to $\mathcal{G}$ if, for all *probabilistic polynomial-time algorithms* $\mathcal{A}$, there exists a negligible function *negl* such that

$$Pr[\mathcal{A}(\mathbb{G}, q, g, h) = x] \leq negl(n).$$

Taking the previous definition into consideration, we infer that the DL assumption holds as long as there exists a $\mathcal{G}$ where the DL problem is hard.

We now explain the Diffie-Hellman problem, which can be split into two distinct problems: the weaker *computational Diffie-Hellman* (CDH) and the *decisional Diffie-Hellman* (DDH) [25].

Definition 2.2.4. The *decisional Diffie-Hellman* (DDH) problem states that, given a cyclic group $\mathbb{G} = \langle g \rangle$ of order q and $g, h_1, h_2 \in \mathbb{G}$, where $h_1 = g^{x_1}, h_2 = g^{x_2}$ and $x_1, x_2 \in \mathbb{Z}_q$, it is *hard* to determine whether an element $h = g^{x_1 x_2}$ or if h was chosen uniformly from $\mathbb{G}$.

Intuitively, the CDH problem is based on the difficulty of *calculating* $h = g^{x_1 x_2}$, which relies strongly on the DL problem.

We now consider an algorithm $\mathcal{B}$, which receives the tuple $(\mathbb{G}, q, g, g^a, g^b, g^c)$, where $a, b, c \in \mathbb{Z}_q$, and outputs 1 if $g^c = g^{ab}$, and 0 otherwise.

Definition 2.2.5. The DDH problem is *hard* with respect to $\mathcal{G}$ if, for all *probabilistic polynomial-time algorithms* $\mathcal{B}$, there exists a negligible function *negl* such that

$$\big|Pr[\mathcal{B}(\mathbb{G}, q, g, g^a, g^b, g^c) = 1] - Pr[\mathcal{B}(\mathbb{G}, q, g, g^a, g^b, g^{ab}) = 1]\big| \leq negl(n).$$

Then, we conclude that the DDH assumption sustains, as long as there exists a $\mathcal{G}$ where the DDH problem is hard.

2.3 Elliptic Curves

The application of elliptic curves in cryptography was initially proposed by Miller [79] and Koblitz [71] in 1985. While it poses many advantages, such as reduced key sizes and stronger security, when compared to the well-known, widely embraced RSA cryptosystem [96], its adoption only began roughly a decade ago and it has been growing slowly.

Elliptic Curve Cryptography (ECC) leans on the definition of finite fields over elliptic curves. The attractive feature of elliptic curves is that points can be arranged into (cyclic) groups where the discrete logarithm problem is believed to be hard. So, when they are chosen adequately, the DL assumption holds because there exist no algorithms to solve it efficiently.

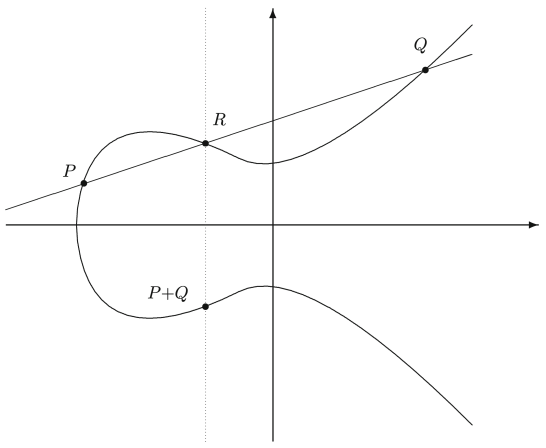
In this section, we succinctly explain how elliptic curves operate and clarify how they are used in the context of public key cryptography for this work. We mention Katz and Lindell [70] and Smart [103] for a more thorough explanation.

We now consider a finite field $\mathbb{F}_p$, where $p > 3$ is a prime. The equation

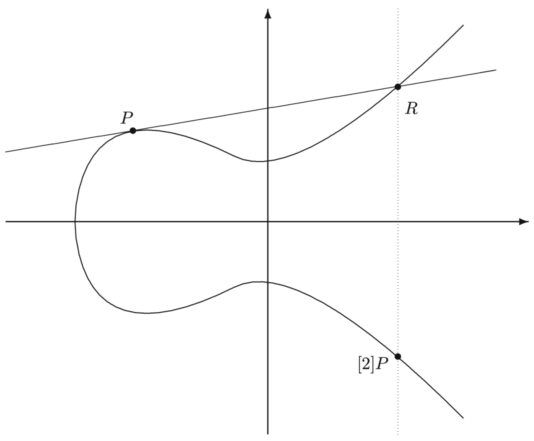
$$E : y^2 = x^3 + Ax + B \ (mod\, p)$$

generally defines an elliptic curve over the finite field $\mathbb{F}_p$, where $A, B \in \mathbb{Z}$ are constants. In this case, the *finite abelian* group of points on the curve is denoted by $E(\mathbb{F}_p)$, and there exists a special point $\mathcal{O}$, called *point at infinity*, which is considered the *additive identity*.

Subgroups of points from $E(\mathbb{F}_p)$ support addition and multiplication, as portrayed in Figure 2.1. In fact, the multiplication of a point $P \in E(\mathbb{F}_p)$ is only supported with a scalar value n, and comprises the addition of P with itself n times. This operation, $P + P + ... + P$ or nP, although easy to compute, is believed to be hard to reverse. In other words, it is hard to compute n from $Y = nP$, knowing both Y and P. This is called the *elliptic curve discrete logarithm problem* (ECDLP) and it is the foundation of ECC.



(a) Point Addition



(b) Point Multiplication

Figure 2.1: Operations on Elliptic Curves [103].

One important consideration about the ECDLP is that, unlike the DLP for linear algebraic groups, there exists no algorithm which can solve it in sub-exponential time, for the majority of cases. In fact, it is believed to be harder to solve than other well-known problems, such as the standard discrete logarithm and the integer factorization problems, and thus, yielding greater key size efficiency.

Another pertinent aspect about elliptic curves is the notation used to represent the operations between group elements or, in this case, points. As a convention, standard operations in a linear algebraic group $\mathbb{G}$, namely, multiplication and exponentiation, are considered analogous to the addition and scalar multiplication in a group $E(\mathbb{F}_p)$ of elliptic curve points, respectively. For example, denoting g^n in linear algebraic groups can be regarded as equivalent to writing nG in ECC. Throughout this work, while we usually resort to the algebraic notation, we may still use the elliptic curve notation in some cases.

ECC can excel at a variety of protocols, ranging from key agreement to digital signatures. In this work, we resort to an elliptic curve to generate key pairs and zero-knowledge proofs. Concisely, key pairs in ECC are composed of a *private key* $s \in \mathbb{Z}_p$ and a *public key* $P = sG$, where G is a predefined *base point* of the curve. Taking advantage of the ECDLP, we ensure that it is hard to deduce the private key s from the public key P.

2.4 Zero-Knowledge Proofs

Zero-knowledge proofs are very powerful cryptographic techniques, which allow a prover $\mathcal{P}$ to prove a statement to a verifier $\mathcal{V}$, without revealing any information, except the validity of that statement. One very useful example is proving knowledge of a private key, which allows the verifier to know that the prover holds the private key but *nothing else*. Zero-knowledge proofs were initially proposed by Goldwasser et al. [62] in 1989.

When defining a zero-knowledge proof, the following conditions must hold:

- **Completeness**: If both the prover and the verifier are honest and the protocol is followed, the verifier should accept the proof with probability one.
- **Soundness**: If the verifier is honest but the prover is dishonest, and the statement is false, the verifier should only accept the proof with very low probability.
- **Zero-Knowledge**: The verifier should not learn anything aside from the validity of the statement.

We refer to Smart [103] for more details on this topic.

2.4.1 Proofs of Knowledge

Proofs of knowledge are a category of zero-knowledge proofs. As expected, the goal is to prove knowledge of a secret value without revealing it. We can achieve this using an interactive protocol called sigma (Σ), which comprises three moves: *commitment*, *challenge* and *response*. The Schnorr protocol [101] is a particular example of the sigma protocol, which allows one to prove knowledge of a discrete logarithm in a cyclic group where the DL problem is hard.

To define proofs of knowledge formally, we resort to the symbolic notation proposed by Camenisch and Stadler [32]. We then generally represent proofs of knowledge as

$$PK\{(x_1, ..., x_n) : \text{logical operations between statements about } x_1, ..., x_n\}$$

where $x_1, ..., x_n$ are only known to the prover, and everything else is common knowledge.

As an example, we provide the proof of knowledge $PK\{(x) : h = g^x\}$, where $\mathbb{G} = \langle g \rangle$ of prime order p and $h \in \mathbb{G}$ are known to all parties, and $x \in \mathbb{Z}_p$ is secret. The flow of the protocol is shown in Figure 2.2.

As anticipated, one can also join together multiple statements about the secrets which knowledge is being proved. Figure 2.3 depicts the course of the sigma protocol for $PK\{(x, y) : a = g^x \wedge b = h^x \cdot g^y\}$, which has a conjunction of statements. Once again, everything except the secrets $x, y \in \mathbb{Z}_p$ is well known by all entities.

Prover $\mathcal{P}$		*Verifier* $\mathcal{V}$
pick random $k \in \mathbb{Z}_p$		
$r := g^k$	$\xrightarrow{r}$	
		pick random $c \in \mathbb{Z}_p$
	$\xleftarrow{c}$	
$s := k + cx \ (mod\ p)$	$\xrightarrow{s}$	confirm:
		$g^s = h^c \cdot r$

Figure 2.2: Sigma protocol to prove knowledge of x.

Prover $\mathcal{P}$		*Verifier* $\mathcal{V}$
pick random $k_x, k_y \in \mathbb{Z}_p$		
$r_a := g^{k_x}$		
$r_b := h^{k_x} \cdot g^{k_y}$	$\xrightarrow{r_a, r_b}$	
		pick random $c \in \mathbb{Z}_p$
	$\xleftarrow{c}$	
$s_x := k_x + cx \ (mod\ p)$		
$s_y := k_y + cy \ (mod\ p)$	$\xrightarrow{s_x, s_y}$	confirm:
		$g^{s_x} = a^c \cdot r_a$
		$h^{s_x} \cdot g^{s_y} = b^c \cdot r_b$

Figure 2.3: Sigma protocol to prove knowledge of x AND y.

Additionally, we point out a scenario where we would like to prove knowledge of two discrete logarithms with a single challenge value, $PK\{(x) : a = g^x \wedge b = h^x\}$. We call this special case the Chaum-Pedersen protocol [34], which flows exactly like the Sigma protocol.

We may also want to prove knowledge of n out of m statements. These proofs of partial knowledge, based on the sigma protocol, were first suggested by Cramer et al. [40]. To exemplify, we demonstrate the proof $PK\{(x,y) : a = g^x \vee b = h^y\}$, where everything aside from $x, y \in \mathbb{Z}_p$ is common knowledge. Because this is a disjunction of statements, the prover has to prove, at least, one of the two statements. In this case, we assume the prover only knows x. The flow is illustrated in the Figure 2.4.

One relevant aspect about the interactive sigma protocol is that its results are only verifiable by the participating entities. Also, this proof system might not be secure against a cheating verifier, which may have an unknown efficient strategy to recover the secret value, after many executions of the protocol. We call this property *honest-verifier zero-knowledge* (HVZK).

In 1986, Fiat and Shamir [54] proposed a solution to transform the three-move HVZK interactive protocol into a non-interactive general proof of knowledge. In this technique, called *Fiat-Shamir heuristic*, the challenge is generated from the common knowledge, which eliminates the need of calculating a random challenge. Figure 2.5 shows a non-interactive protocol for $PK\{(x) : h = g^x\}$. We define

Prover $\mathcal{P}$		*Verifier* $\mathcal{V}$
pick random $k_x, c_y, s_y \in \mathbb{Z}_p$		
$r_a := g^{k_x}$		
$r_b := h^{s_y} \cdot b^{-c_y}$	$\xrightarrow{r_a, r_b}$	
		pick random $c \in \mathbb{Z}_p$
	$\xleftarrow{c}$	
$c_x := c - c_y \ (mod\ p)$		
$s_x := k_x + c_x x \ (mod\ p)$		
	$\xrightarrow{c_x, c_y, s_x, s_y}$	confirm:
		$c = c_x + c_y \ (mod\ p)$
		$g^{s_x} = a^{c_x} \cdot r_a$
		$h^{s_y} = b^{c_y} \cdot r_b$

Figure 2.4: Sigma protocol to prove knowledge of x OR y.

$H : \{0,1\}^* \to \{0,1\}^*$ as a cryptographic hash function and denote $a \| b$ as a concatenation of strings a and b.

Prover $\mathcal{P}$		*Verifier* $\mathcal{V}$
pick random $k \in \mathbb{Z}_p$		
$r := g^k$		
$c := H(h \| g \| r)$		
$s := k + cx \ (mod\ p)$	$\xrightarrow{s, r}$	calculate: $c := H(h \| g \| r)$
		confirm: $g^s = h^c \cdot r$

Figure 2.5: Non-interactive protocol to prove knowledge of x.

The transcript of the non-interactive version of the sigma protocol can be saved and verified later by everyone. Relevantly, this non-interactive protocol can be used to create a signature scheme, which is called a *signature proof of knowledge* (SPK). Such signature is represented as

$$SPK\{(x) : y = g^x\}(m).$$

This example basically constitutes a Schnorr signature [101], where message m is signed using the private key x. This works exactly like the non-interactive protocol shown in Figure 2.5, except that message m is added to the challenge, such that $c := H(h \| g \| r \| m)$. One advantage of this scheme is the possibility of adding other conjunctive or disjunctive statements, which allows one to build a signature, while proving knowledge of complex statements.

Chapter 3

Related Work

The key purpose of this chapter is to give a better perception of the electronic voting field, the challenges it entails and how they have been solved throughout the past decades. Furthermore, this chapter provides a realization of how the existing work contributed to the solution proposed in this dissertation.

We start by reviewing the essentials of electronic voting in Section 3.1. We present the implications of voting electronically, followed by an overview of the entities and phases of a general voting process. This overview wraps up with the assessment of the properties desired in an electronic voting system. Afterwards, Section 3.2 discusses the current protocols, grouping them by cryptographic technique and presenting them chronologically. Still in Section 3.2, we study the differences between the cryptographic approaches, comparing them by efficiency, flexibility and security.

Finally, Section 3.3 presents a concept called *ring signature*, which has many use cases, ranging from e-voting to e-cash. We explain this cryptographic notion, clarify its relevance to voting schemes and discuss current work.

3.1 Electronic Voting

In the past decades, electronic voting systems have been getting enhancements, making them a reasonable alternative to traditional physical voting protocols. They can yield significantly better cost and time efficiency, which makes them even more attractive.

This applies to internet voting, which, aside from the aforementioned advantages, can also improve flexibility for the voters, specifically by allowing them to vote in a more convenient location, which may even increase turnout. Although, electronic voting systems are not limited to internet-based, fully electronic systems, instead they also encompass a few other types. The two most relevant ones are explained below.

- **Direct-recorded electronic** (DRE) voting machines are devices which present voters with a digital ballot and allow them to vote through an interface, usually buttons or a touchscreen. DRE machines replace paper ballots in the sense that they are placed in conventional voting stations, promoting uncoercibility.

- **Optical Scan** voting machines combine paper ballots with electronic devices, such that voters are given special ballots, which are then read by this machine. While they do not replace the paper ballots, they allow for faster tallying.

Voting systems are composed of entities which follow steps to achieve a result. While some of them may resort to different techniques, the goal is the same: a process which delivers security, privacy and integrity. The following two subsections develop on the entities and phases of a generalized voting scheme, and the last subsection states the core requirements to accomplish the main objective.

3.1.1 Entities

This section elucidates on the components of a voting scheme. Since this is just an abstract definition, a true voting implementation may have more entities than the ones mentioned.

- **Voter**: This entity has the right to choose between a number of possibilities, or to write up their choice[1], and then submit their decision.
- **Registration Authority**: It is responsible for assuring that only eligible voter exercise their right at most once. Therefore, it needs to check the identity of a voter and verify if such elector is registered to vote before they cast a ballot.
- **Tallying Authority**: This authority is in charge of collecting, counting the votes and publicly sharing the election results. It should also confirm the authenticity and uniqueness of each ballot, however this can be done by a separate entity called *validator*.
- **Supervisor**: Manages the entire voting process, compiles a list of eligible voters and is able to start, suspend and terminate elections.

Finally, there exists a structure commonly called *bulletin board*, which works as an append-only log used by voting protocols to save relevant voter information and ballots. A bulletin board can be distributed among multiple machines to improve availability, and usually ensures the integrity of data, for example, by using digital signatures.

3.1.2 Phases

To better understand how the voting process develops, this section states and describes all the steps necessary to conduct a correct electoral process. Once again, this is an abstraction so, for that reason, a real solution might have more or less phases than the ones referred.

1. **Preparation**: This consists of compiling a list of eligible voters and preparing the ballots.
2. **Authorization**: Verify if a voter who is trying to cast a ballot is indeed authorized to do so.
3. **Voting**: A voter makes a choice and submits their decision.

[1]This is called a write-in ballot.

4. **Claiming**: The voters are able to check if their vote is set for counting, and complain if a cast vote is not listed for tallying.
5. **Tallying**: The tallier checks that all votes are well formed, counts all of them and publishes the election outcome.

3.1.3 Properties

Most of the properties of a secure voting approach are already listed and explained in Section 1.2. Even though, there are still a few which will not be implemented in this work and were not previously mentioned. These properties are clarified below.

- **Fairness**: In a fair system, election authorities cannot know voting results until the tallying phase.
- **Receipt-Freeness**: This notion refers to refraining the system from providing voters with a proof of how they voted, after the ballot was cast [20, 64, 86, 99].
- **Coercion-Resistance**: This is the strongest privacy property, which, besides offering receipt-freeness, tries to fight an active coercer by making infeasible for the adversary to confirm if the coerced voter complied with the demands [38, 69].

The last two properties are improvements to the privacy property, and thus, much stronger. They aim at preventing *vote-buying* and *vote-coercion*, and that is why many electronic voting proposals try to incorporate them.

3.2 Voting Protocols

This section focuses on the evolution of existing voting solutions. Besides describing the work accomplished by each proposal, such as goals, contributions and relevance, we also identify strengths and limitations of each one. In addition, we point out the aspects which can be improved or extended and any features which are planned to be refined in the future.

Generally, voting protocols are categorized by the cryptographic procedure used to anonymize votes. The first three subsections present the voting proposals based on the most frequently employed cryptographic approaches. While these three are the most common, there exist other techniques, namely, *threshold cryptography* [46, 47, 90], which finds its roots in the secret sharing scheme by Shamir [102], and zero-knowledge proofs [62]. They are usually used together with one of the leading approaches. For example, threshold cryptography is usual in multi-authority voting protocols, in which entities share a private key.

Schemes in each subsection are sorted out with respect to publishing date and correlation with other work, meaning that subsequent approaches are newer and possibly extend previous solutions. The last subsection discusses and compares all the solutions and how they fit and adapt to each particular situation.

3.2.1 Mix Networks

The concept of mix network was first presented by Chaum [37] in 1981 and it set out to solve the problem of anonymization of communication between two parties and their identities, both in email and in electronic voting. In this approach, an entity called *mix* receives a list of encrypted inputs, and decrypts and shuffles it in a random way, such that the outputs are permuted and, thus, unlinkable to the inputs. Usually, to achieve greater anonymity, a few mixes are set up sequentially, creating a *mix network* (abbreviated, mixnet) or *shuffle network*. This technique is particularly useful when the underlying communication system does not provide confidentiality or, even more important, anonymity.

Mix networks can be implemented in two ways. The first one is called a *re-encryption mixnet* and, in this case, the information is encrypted initially, and every time it goes through a shuffle agent, it is re-encrypted and permuted. The second one is the *decryption mixnet* proposed by Chaum [37], and it works by first encrypting the inputs successively with the public key of each mix (like an onion), and then sending them through the mix network in the inverse order of the encryption, for each mix to decrypt its messages, shuffle them and send them to the next mix. The latter is depicted in Figure 3.1.

In the context of voting, the inputs to the mix network are the ballots. Succinctly, the encrypted ballots, along with the voter identity, are placed in the bulletin board. Throughout the voting phase, the voters can access the bulletin board and confirm that their encrypted ballot is, in fact, there. When the election finishes, the identities of the voters are discarded, and the ballots are shuffled and decrypted by the mix network, in a single batch. In the end, the decrypted ballots can be counted and no one is able to correlate them with the voters, or even with the initial ciphertexts.

When used in voting, it is desired that a mixnet provides *correct operation*, *privacy* by creating unlinkability between inputs and outputs, and *robustness* by generating strong proof that the shuffling and the cryptographic operations were carried out correctly.

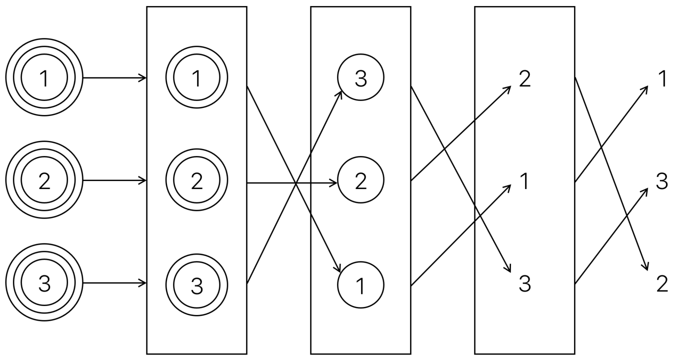


Figure 3.1: Operation of a Decryption Mix Network.

Chaum [37] is considered the pioneer of electronic voting because his proposition includes an election scheme which ensures privacy, providing unlinkability between voters and ballots. This work has the greatest relevance in this field and its contributions are present in many subsequent voting solutions. Still, this primordial proposition lacks efficiency as the length of the ciphertext increases with the number of mixes used. Also, it focuses only on the operation of the mixnet and not on proving that shuffling was done correctly, which is important in a voting context, since it usually translates into verifiability.

Following this approach, Park et al. [88] proposed a new anonymous channel, which improved the efficiency of the Chaum mix network, and developed one of the first election schemes based on mixnets. The first contribution solved the aforementioned ciphertext length expansion problem present in the work by Chaum [37]. This was achieved by using ElGamal [49, 52] instead of RSA [96], making the ciphertext length, and corresponding voter work, independent of the number of mixes. Second, they presented a voting protocol which resorts to their improved mix network, automatically enhancing efficiency. It also satisfied fairness by guaranteeing that votes disrupted due to the incorrect operation of the mix network were detected, and, thus, ensuring that a partial tally is not released. This will not influence an eventual follow-up re-election process, an issue which could happen in the proposition by Chaum [37].

Notwithstanding, this protocol [88] still has a few weaknesses. Pfitzmann [92] showed that the anonymity of the proposed channel can be compromised, either by taking advantage of the cryptosystem, or by actively injecting specific ciphertexts into the mixes. Regarding the election scheme, ballots have to be decrypted individually, and malformed votes are only detected during the tallying phase.

The next paper presented one of the first voting solutions featuring receipt-freeness, a property previously introduced by Benaloh and Tuinstra [20], and pioneered the concept of universal verifiability, nowadays existing in the majority of voting systems, and therefore the relevance of this work. Sako and Kilian [99] proposed said voting solution, which resorts to anonymous channels [37, 88] to ensure voter privacy. Aside from this, the proposition reduces the unwieldy necessity of a physical voting booth required by the Benaloh-Tuinstra [20] and Niemi-Renvall [83] protocols to the existence of a one-way untappable channel from the voting authorities to the voters. In this work, universal verifiability is achieved by requesting each mix center to prove interactively[2] that their messages were processed correctly.

Despite these improvements, this approach by Sako and Kilian [99] still has a few setbacks. The first one is shared between all solutions of this category, which is the heavy processing load that mix networks imply. Second, while the voter effort is independent of the number of mix servers, the work of a verifier is still proportional to it, since each mix has to be checked. Third, the abovementioned one-way untappable channel cannot be satisfied by cryptography, and hence the channel must be physical. Finally, Michels and Horster [78] pointed out two more serious issues, namely, a coercer cannot collude with a mix, otherwise it might compromise the correctness of the tally, and one honest mix may not be enough to ensure privacy. As a matter of fact, none of the presented solutions so far are robust, that is, if a mix refuses to cooperate, the system is incapable of producing a correct tally.

Abe [3] came up with a robust, threshold mix network which holds the universal verifiability property resorting to *cut-and-choose*[3] methods. This proposition is different than the previous ones because it implements threshold cryptography [47], so, instead of having a mix server own an entire private key, this secret key is generated in a distributed manner, such that t mixes are necessary to decrypt some

[2]The proof can be made non-interactive by employing the Fiat-Shamir Heuristic [54], allowing everyone to check the correctness without having to interact with the mix.

[3]This class of protocols was first introduced by Rabin [94], and popularized as cut-and-choose by Brassard et al. [27], one decade later. They allow an entity to prove to another that some information respects the convention agreed upon by them, usually without revealing any part of the information itself. Examples include zero-knowledge proofs [62] and witness hiding protocols [53].

information. As a result, this protocol attains privacy even if $t-1$ servers are malicious, and robustness as long as, at least, t shufflers are cooperative. Regarding efficiency, the work done by each verifier is uncorrelated with the number of mix servers, unlike in the work by Sako and Kilian [99], and the tasks required by each mix are independent of the number of mixes as well.

Still, in terms of computational effort, efficiency can be improved and the protocol is impractical for large-scale elections. Also, when the inputs to the mix network are very long, it is usually a best practice to divide them into smaller ones, otherwise performance might decrease severely, which is a problem affecting the next proposal as well.

In the same year, Jakobsson [65] presented a more efficient and robust mix network. Similarly to the previous work [3], this proposition consists of a mix network based on re-encryption and threshold decryption. It avoids the use of cut-and-choose methods, replacing them with a scheme of *undeniable signatures*[4], and allows the work and interactions for each voter to be significantly low, thus conferring much better efficiency to this protocol. However, this approach lacks a crucial integrity property, which is universal verifiability, making it unsuitable for the majority of voting scenarios. In addition, it has been demonstrated by Desmedt and Kurosawa [48] that this solution is not as robust as claimed by presenting an attack which compromises one mix server, consecutively preventing the system from computing the correct output.

Since proofs of correct operation in mixnet-based schemes are complex and computationally demanding, Neff [82] introduced a new protocol to obtain shuffle verifiability, in which the complexity is linear to the number of ciphertexts, and applied this solution to current voting systems, achieving greater efficiency. Consequently, this protocol accomplished universal verifiability at a much lower cost than the one presented by Sako and Kilian [99]. Despite being a strong approach to achieve verifiability, the author focused only on the efficiency of proving shuffles of ElGamal ciphertexts. Also, while this proposal is efficient and adequate for large-scale voting, versatility and efficiency can still be improved.

In 2002, a novel technique to prove the correctness of a mix network was put forward by Jakobsson et al. [67]. This procedure, named *randomized partial checking*, provides strong evidence of correctness instead of a full proof of accurate operation, which can be even faster than the operations carried out by the mixes (such as permutations). Hence, it is more efficient when compared to other systems which use other cut-and-choose protocols, such as zero-knowledge proofs. It works by requiring each mix to randomly reveal half of its input/output pairs, and allowing verifiers to confirm that they match. In a voting context, this translates into a much simpler and less costly option.

Nevertheless, probabilistic verification faces a somewhat considerable tradeoff with privacy, since the mixes must constantly disclose random information to be verified. Therefore, voter privacy is provided by the mix network as a whole, rather than by each mix individually.

The next proposition introduced one of the first solutions which includes coercion-resistance, consequently addressing common problems in voting, namely, vote-coercion and vote-buying. Juels et al. [69] claimed that most receipt-free approaches comprise unrealistic assumptions and have to face some concerns, specifically *forced abstention*, *randomization*, wherein a voter is coerced into choosing a ran-

[4]Introduced by Chaum and van Antwerpen [35], undeniable signatures are similar to digital signatures, with the exception that they require the cooperation of the signer to be verified. Without help from the signer, the verifier cannot validate the signature.

dom candidate, neutralizing the vote, and *simulation*, in which a coercer forces a voter to divulge their private key and votes on their behalf. The contributions of this work encompass a definition of the properties of coercion-resistance and the development of a voting scheme based on the previously mentioned definitions, which was also able to weaken the assumptions of preceding receipt-free protocols. Yet, this solution requires a time-consuming preparation phase and it is only practical for small elections since the work of tallying authorities grows quadratically with the number of voters.

Then, Benaloh [17] proposed a lightweight and less sophisticated yet effective voting approach. This system is based in part on the mix network developed by Sako and Kilian [99], wherein each mix server appends a shuffle proof to the ballots, resorting to the Fiat-Shamir heuristic [54]. Ballots are then decrypted using a key shared among the trustees. The main disadvantage of this protocol is efficiency, which ends up as a tradeoff for simplicity. This is because the protocol is purposely not as optimized as the majority of the previous ones, enabling a simpler implementation and an easier adoption.

Finally, Bayer and Groth [15] designed a zero-knowledge shuffle correctness argument, in other words, a mathematical system to generate non-interactive proofs that a mix network operated correctly, which reveal no information about permutations. Introduced as improvement to Sako and Kilian [99], Abe [3] and Neff [82], this is one of the most efficient shuffle arguments to date, adaptable to homomorphic cryptosystems other than ElGamal, and achieving sublinear complexity in communication. While this paper does not propose a voting scheme, its contributions are pivotal to the development of efficient mix networks and, consecutively, mixnet-based voting protocols.

Examples of Mix-Network-Based Voting Systems

Helios. Devised by Adida [8] in 2008, Helios is a fully electronic voting system built upon the Sako-Kilian [99] and Benaloh [17] procotols, which focuses primarily on *auditability*, and therefore promoting *unconditional integrity*, even if all authorities are corrupted. As expected, anonymity is attained by relying on mix networks, which provide proofs of correct operation upon decryption. This system does not address coercion-resistance and provides few security guarantees if voters are submitting ballots from compromised environments. Still, Helios is capable of attenuating some of the problems of pure electronic voting, such as undermining vote integrity, by providing end-to-end verifiability.

Civitas. Based on the scheme proposed by Juels et al. [69], Civitas [38] is the first end-to-end verifiable and coercion-resistant voting system. It is able to achieve coercion-resistance by allowing a voter to generate fake credentials and vote with them in the presence of a coercer, later invalidating the vote. A mix network, which resorts to randomized partial checking [67] to enforce verification, is used to anonymize votes and credentials. Also, Civitas is able to mitigate the scalability problem of Juels et al. [69] by dividing the election into small blocks of votes, making it suitable, although costly, for large elections.

3.2.2 Homomorphic Encryption

Homomorphism in cryptography allows the application of specific computations to a ciphertext without having to decrypt it, and producing the same result as if they were applied to the plaintext. In particular, the *homomorphic property* is the ability to concatenate multiple ciphertexts together and to decrypt them as a whole. Some well known cryptosystems which support this property are Paillier [87] and a ElGamal [49, 52].

Regarding voting systems, this technique works by encrypting all votes separately with a public key which belongs to election authorities, then adding all of them together and, only after, decrypting the aggregated set of votes at once, which reveals no information about the votes individually. It is important to highlight that this approach only concerns the tallying process, so other requirements, such as verifying ballot validity, should be fulfilled through proofs of correctness.

The first homomorphic-encryption-based voting protocol was proposed by Cohen (now Benaloh) and Fischer [39] in 1985. In this scheme, besides the voter, there exists a centralized entity designated *government*, responsible for guaranteeing computational privacy, which relies on the intractability of the *residue problem*[5]. Even colluding with dishonest voters, said government can only release a false result with very low probability, thus ensuring integrity as well. While this solution focuses only on yes/no voting, it can be extended to multiple choices, although it would impact the efficiency. Still, the greatest downside of this approach is the ability of a dishonest government, as a single authority, to know any vote individually. The authors suggested solving this problem in the future by distributing the tallying work between a few authorities, which would only compromise privacy if these entities cooperated with each other.

The following year, Benaloh and Yung [21] picked up this suggestion and came up with a voting solution which distributed the powers of the government, therefore protecting the privacy of individual votes, even if just one authority is honest. Besides privacy, the authors were able to decrease the chance of the system producing an incorrect but credible tally, improving integrity. Nonetheless, this scheme features no robustness because if one tallier does not respect the protocol, no tally is produced and the election becomes void.

The next proposal was developed along the lines of the previous two [21, 39]. The approach by Benaloh [22] is described as a (J, K, L) election system, meaning that there are L voters and K tallying authorities. The election process finishes only if a threshold J of talliers post their voting results. Any entity can posteriorly verify the accuracy of the reported tally. Similarly to the preceding work, all information is shared publicly in a bulletin board, including encrypted votes. These are agglomerated using the homomorphic property and decrypted at once, not revealing individual votes.

Still, these three approaches [21, 22, 39] are not suitable for large-scale elections since the communication and computation overheads would increase considerably. They also share the possibility of a

[5]Given $q \in \mathbb{Z}_N^*$, we say q is an r^{th} *residue modulo N* (or *quadratic residue*, if $r = 2$), if there exists an x such that $q \equiv x^r \ (mod\ N)$. The residue problem states that it is hard to decide whether q is an r^{th} residue modulo N, given q and N, where $N = p_1 \cdot p_2$ and p_1, p_2 are primes. This problem, first discussed by Gauss [59], is weaker than the *integer factorization* problem since, if N is not a composite integer, the problem is easily solvable. We refer to Goldwasser and Micali [60] for a more detailed explanation.

voter to prove through a receipt how they voted after the ballot was cast. This characteristic is undesirable if there exists an adversary trying to influence the choice of the voter, raising concerns related to vote-buying and vote-coercion.

In 1994, Benaloh and Tuinstra [20] introduced the concept of *receipt-freeness*, which focuses on disallowing voters to take away a ballot receipt and later be able to prove how they voted, assuming they vote in a safe environment. The contributions of this work also include the development of two voting schemes, which promote not only receipt-freeness, but also privacy and correctness. The first one works with a centralized tallying authority, which jeopardizes privacy. The second one was an adaptation to support multiple talliers, guaranteeing secrecy and making sure the probability of generating a convincing fake tally is negligible. Despite all the effort to develop a receipt-free protocol, Hirt and Sako [64] proved later that it was possible to construct a receipt, and that this approach was not receipt-free after all.

In the same year, a new scheme was suggested by Sako and Kilian [98], which focused on improving performance. Previously proposed protocols, such as the ones by Benaloh [22] and Benaloh and Yung [21], lack communication efficiency and this solution increases said efficiency remarkably. The voting phase is simple and lightweight, and there exists the possibility of choosing whether to prioritize security or efficiency by adjusting the number of tallying centers, thus conferring flexibility. Instead of resorting to the residue problem as the underlying mathematical system of its zero-knowledge protocols, this scheme relies on the discrete logarithm assumption, which improves the performance since the computations are simpler.

Notwithstanding, this protocol does not encompass receipt-freeness. Moreover, a malicious voter can copy a vote by duplicating the ballot, which goes unnoticed. Finally, the greatest drawback of this proposal, which also affects the previous papers, is the impact on efficiency if it needs to support voting with more than two choices.

After, Cramer et al. [41] invented a voting solution which demanded a linear number of cryptographic operations from the authorities, instead of a quadratic one. Therefore, the main contribution of this work is an efficient voting scheme. Additionally, it offers the usual desired voting properties, such as privacy and robustness, and solves the vote duplication problem of the preceding proposition [98].

In this solution, multiple authorities substitute a centralized authority, which would be able to reveal individual votes. The authors also proposed a replacement of the zero-knowledge proofs used by the previous systems with a more efficient witness hiding protocol, set forth by Cramer et al. [40]. Voter effort is reduced and becomes linear, but still dependent on the number of authorities. Nonetheless, this approach is not receipt-free and the next proposal shows that efficiency can be improved even more.

The following year, a new solution was presented by Cramer et al. [42], which complements the previous work by making the voter's effort complexity independent of the number of authorities. In this protocol, voters post their encrypted vote directly to the bulletin board, along with a proof of valid ballot format, which prevents the disruption of the election process before the tallying phase. Once again, this scheme resorts to multiple authorities to strengthen privacy and robustness. These authorities fall back on the threshold cryptography protocol put forward by Pedersen [90] (and based on Shamir secret sharing [102]) to generate private keys distributedly, and later decrypt all votes joined together.

Nonetheless, this scheme [42] does not satisfy receipt-freeness, although the authors explain how it can be achieved, leaving it as future work. Also, if the election requires more than two candidates, the ballot and the proof increase in size and complexity, respectively, and thus, degrading efficiency.

Then, Hirt and Sako [64] advanced a new proposal, which combines the receipt-freeness of the mixnet-based solution by Sako and Kilian [99] and the efficiency of the previous work [42], making this proposition one of the most efficient receipt-free homomorphic voting systems to date.

In this protocol, before the election, a list of possible votes is compiled. Then, this list is randomly ordered and re-encrypted by all authorities sequentially (similar to a mix network), and each authority provides the voter with a designated verifier proof[6] of permutation to demonstrate how the shuffling was done. After the last shuffle, the list of successively re-encrypted votes is shown to the voter. Due to the designated verifier proofs, the voter is the only entity capable of deducing the equivalence between the ciphertexts and the actual votes, which makes this protocol receipt-free. Finally, the voter makes their choice by selecting one of the encrypted outputs.

While the main focus of this work is adapting the Cramer et al. [42] proposition to be an efficient 1-out-of-L[7] receipt-free voting solution, Hirt and Sako [64] also proposed a 1-out-of-2 scheme, which turned out to be an exceptionally efficient solution in yes/no elections. Although, this protocol still has a few drawbacks to point out. It assumes the existence of untappable channels which, if tapped by an attacker, undermine vote secrecy. Furthermore, the decryption complexity is proportional to the number of choices, so it might not have the desired efficiency if L is large.

In 2001, Baudron et al. [14] devised another very efficient homomorphic receipt-free scheme, with performance equivalent to the previous work [64]. This system is practical in the sense that it is flexible enough to support many voters and candidates, without compromising performance significantly. Authorities are structured hierarchically, specifically, national, regional and local. Local authorities share a private key, such that privacy is only compromised if a threshold of them collude. Tallying is performed in stages of hierarchy, that is, local authorities compute their tallies and give them to the regional authorities, which merge and forward them to the national authority, producing a final tally.

This proposition uses the homomorphic property of the Paillier cryptosystem [87] to concatenate votes. Although, the use of Paillier poses an efficiency disadvantage when compared to ElGamal [52], since its computations are more complex and less efficient, specially in distributed key generation. Also, this approach is limited to 1-out-of-L, not being able to support K-out-of-L voting.

Examples of Homomorphic-Encryption-Based Voting Systems

Helios 2.0. The year after the development of the mixnet-based system Helios [8], a new version was launched. Though simple and effective, the first version lacked scalability and vote decryption was delegated to a single trusted authority. The new end-to-end verifiable system by Adida et al. [9] made the choice of shifting from mix networks to homomorphic tallying based on Cramer et al. [42] because the

[6]Designated verifier proofs [66] are a type of cryptographic proof which is not verifiable by everyone. Instead, only specific entities are able to verify its correctness.

[7]Voting schemes are defined as K-out-of-L schemes when the voter chooses K options out of L choices.

tallying process becomes easier to implement, and thus, easier to verify. It also ensures confidentiality by encrypting the votes on the computer of the voter, and distributing the private key among a set of authorities. While still not addressing coercion-resistance, this system makes up an efficient and simple solution, which holds on to the core voting properties.

Votebox. This system was presented by Sandler et al. [100] in 2008. Like Helios 2.0, Votebox is also based on the approach by Cramer et al. [42] and provides end-to-end verifiability. It was designed to work primarily with DRE machines, which are resistant to data loss in case of failure, and require a lighter software stack. The system solves the problem of faulty or malicious DRE devices using a technique called *ballot challenge*, proposed by Benaloh [18], which allows the voters to audit the machine as many times as they want, before casting the ballot. Nevertheless, Votebox focuses mainly on integrity, leaving privacy exposed if a voting machine is compromised.

3.2.3 Blind Signatures

Blind signatures were introduced by Chaum [33] in 1983 as a solution to payment untraceability. They work like regular digital signatures, with one significant difference: the message is blinded, so the signer has no knowledge about the information being signed. Once the data is signed, the message author can unblind the information and request another authority to verify the signature, following the regular verification procedure. One important note is that the signing entity cannot relate the blinded message they signed, with the corresponding unblinded message which they may need to verify later. Some well-known cryptosystems for digital signatures, which are compatible with blind signatures, are RSA [96] and DSA [73].

In a voting environment, a blind signature authenticates the ballot of a voter before the tallying process, ensuring the eligibility of that voter and keeping the vote unknown to the signer. After getting the signed ballot, the voter can unblind it and either send it immediately to tallying authority, or wait as long as they want. In the tallying phase, the ballot is unlinkable to the voter. All in all, this process guarantees privacy and integrity.

In 1992, Fujioka et al. [56] introduced the first widely recognized voting scheme which resorts to this technique, and it is the foundation for the majority of blind-signature-based voting solutions presented so far. It aimed at creating a voting system practical enough to support large-scale elections, which also promotes privacy and fairness. The voters must communicate with the counters (or talliers) using anonymous channels to send cryptographic keys, which can be achieved using a mix network. The third entity is the election administrator, which is responsible for checking the eligibility of the voters, and therefore applying blind signatures to the ballots, if eligibility is confirmed. Before the signature, the voter commits[8] to their choice and the ballot is hidden behind a blinding factor.

Because this is just a primordial solution, receipt-freeness is not approached. Also, if the anonymous channel assumption is broken, the privacy of the voter is compromised. Another disadvantage of this

[8]A commitment scheme [27, 81, 89] is a cryptographic technique which allows an entity to choose a value, hidden from others, commit to it and reveal it later, if desired. It is designed to be binding, meaning that, after commitment, the value chosen cannot be changed.

approach is that voters must be active in all phases of the voting process, which is a considerable inconvenience.

The next proposition was presented by Okamoto [85] in 1996 and receipt-freeness was the main focus. Besides offering privacy and fairness, it was also able to reduce the effort required of a voter to two rounds of messages, one to obtain eligibility and another one to vote. Nevertheless, the succeeding work showed a security flaw in the receipt-freeness property of this scheme due to the lack of a formal definition of said property, leading to a proof that this solution was not receipt-free after all.

In the following year, Okamoto [86] proposed two new voting schemes, one with a stronger assumption, namely a physical booth, and another one with a weaker assumption, which is an untappable channel. The formal definition of receipt-freeness is presented and applied to both proposals, proving that the property is satisfied. Still, it has to be pointed out that a channel that is secret and anonymous (thus, untappable) is extremely difficult to devise. Additionally, a voter must take part in three of the four voting phases (authorization, voting and claiming), which is not practical.

The next protocol focuses on improving the pioneer approach by Fujioka et al. [56]. The contributions of this new approach by Ohkubo et al. [84] concentrate mainly on reducing the required active participation of the voters during the entire election process, which is demanded by the Fujioka et al. [56] protocol. The authors managed to minimize the involvement of the voters to the ballot casting phase, while still being able to satisfy the same properties. In this scheme, there is no single authority, instead a set of authorities work distributedly, meaning that disruption only happens if a threshold of authorities collude. Notwithstanding, this solution does not address receipt-freeness. Also, the required existence of an anonymous channel is still impractical.

Examples of Blind-Signature-Based Voting Systems

Sensus. In 1997, Cranor and Cytron [43] developed a voting system based on the proposal by Fujioka et al. [56]. Resorting to the RSA [96] cryptosystem as the underlying signature protocol, this system can conduct small elections and can be adapted to support large-scale ones. There exists a new entity besides the ones mentioned in Fujioka et al. [56], the registration authority, which is responsible for signing up voters for the election. However, Sensus faces a critical tradeoff between fairness and voter effort: either voters stay active until the end of election, providing their commitment keys only in the tallying phase, or cast the ballot and send the keys immediatly after, allowing the authorities to know intermediate results, and therefore compromising fairness.

EVOX. Proposed by Herschberg [63] in 1997, EVOX was a new voting system built upon the protocol developed by Fujioka et al. [56], which ensured that a voter could walk away after casting a ballot. In this proposal, there exists a new entity called *anonymizer*, which is responsible for collecting the ballots and forwarding them to the tallier, working as a mediator between the voter and the tallier, and guaranteeing that the tallier cannot link messages back to the sender, similar to a mix network. The fairness property is then secured with the use of this anonymizer. Still, this system faces some problems. Because the administrator generates the ballots, it can simply vote on behalf of voters who did not vote, posing a threat to election integrity. Additionally, the anonymizer must be trusted, otherwise anonymity is compromised. Finally, while the system is able to ensure security, efficiency is poor and scalability is difficult, since the single administrator constitutes a bottleneck.

EVOX-MA. Introduced as an improvement to the previous system, DuRette [51] came up with EVOX-MA in 1999, a voting solution which aimed at solving the main problem of the EVOX system, which is the possibility of an administrator to pretend to be an abstaining voter and vote on their behalf. This was solved by distributing the power of the administrator by several machines, and proposing a new entity called *manager*, which task is to distribute the ballots throughout voters, therefore preventing an individual administrator from forging a vote. Nevertheless, it is still possible for the manager, colluding with a set of administrators, to impersonate voters and cast counterfeit ballots. This situation can happen particularly when performance is prioritized, specifically, when the threshold of administrators required to blind sign ballots is small. Also, the system lacks robustness since the manager constitutes a single point of failure, which affects scalability as well.

REVS. The Robust Electronic Voting System (REVS) was presented by Joaquim et al. [68] in 2003 and focused on eliminating the drawbacks of EVOX-MA. One of the main focal points was improving robustness by avoiding single points of failure. Another useful feature of REVS is the possibility of running multiple elections simultaneously, without the threat of voters exchanging or stealing ballots and casting them in elections they are not eligible, a problem that could occur in both EVOX and EVOX-MA. This system was designed so that the voter would only need to trust the client application, and that it would be executed only on trusted hosts. Although, the voter module is vulnerable to interference, which can compromise privacy and, eventually, integrity. Besides, neither this system nor the previous ones address receipt-freeness, which can be relevant in some elections.

3.2.4 Discussion

In this subsection, we discuss and compare the cryptographic approaches with regard to three fundamental aspects which must be taken into consideration when developing an electronic voting system, namely, efficiency, flexibility and security. Also, because voting systems may have distinct requirements and goals, we point out any possible tradeoffs and explain how the techniques adapt to each situation. This section ends with some remarks, which compare the current work with the challenges our protocol faces.

Efficiency. This is a critical feature, especially in large-scale voting. Usually, we refer to efficiency as global property of the protocol. To evaluate it, we take into consideration the following parameters:

- The computational and time resources the protocol demands from a voter. We call this *voter effort*.
- The computational complexity, time spent and bandwidth required for a specific operation, including voter registration, voting and tallying, in the perspective of an authority.

- The monetary cost per voter, with regards to software, hardware and operational costs.

We sorted out these criteria by relevance, with respect to our context. Evidently, in other scenarios, other parameters may be prioritized. While voting system designers aim at maximizing efficiency, it can sometimes end up traded with other requirements, such as verifiability.

Regarding cryptographic approaches, blind signatures offer great efficiency since they only require digital signatures, which are fast and have low computational complexity. Nevertheless, this technique can be time-consuming as it requires, at least, two round trips (getting eligibility and voting).

Homomorphic-encryption-based voting schemes are exceptionally efficient and simple when used for 1-bit voting (yes/no). However, when the ballot size is expanded, efficiency drops severely, as the complexity to generate proofs of correctness increases. Accordingly, it could be the best option, in terms of performance, when a yes/no election is conducted.

Finally, mix networks require little effort from a voter and can achieve superior time performance, if the technique used to verify the correctness of the permutations is efficient. Even with the need to open ballots individually and to validate them in the tallying phase, mix networks provide greater efficiency than homomorphic encryption schemes, if the list of candidates is large. Note that one could also append proofs of correctness to the ballots to avoid possible election disruptions, due to invalid ballots. Though, this would impact efficiency considerably.

Flexibility. This concept concerns the possibility of changing the format of the ballot, and how it can affect efficiency. As one might conclude, the less flexible approach is homomorphic encryption because its complex mathematical structure requires proofs to show that the ballot is well-formed. Therefore, a bigger ballot would mean worse performance due to the increase in complexity of the proofs of ballot validity. This can imply ballot restrictions, and thus, poor flexibility.

On the other hand, blind signature approaches are ballot independent, that is, ballots can have any format of any size, without having to change the implementation or the components of the system. Lastly, mixnet-based solutions have a flexibility similar to blind signatures, since ballots are opened and validated during the tallying phase.

Security. This notion refers to the capability of the voting system to respect its well defined properties, mentioned in Sections 1.2 and 3.1.3. Besides, security also concerns the correlation between robustness and trust assumptions. As referred in Section 1.2, robustness is the resistance of the system to disruption when some entities are faulty or malicious. As for trust assumptions, they include what the system must assume as trustworthy in its environment, in order to function properly. As anticipated, the majority of voting solutions aim at boosting robustness and decreasing trust assumptions to a minimum. Still, there can be a tradeoff between both of these and efficiency, which may sometimes result in very efficient systems being vulnerable to disruption.

In regards to cryptographic techniques, starting by homomorphic encryption, it can attain superior robustness seeing that ballots always carry a proof of correctness, which means that any attempt to disrupt the election can be detected earlier than the tallying stage. Furthermore, talliers usually share a secret key, which decrypts the votes. This yields robustness since only a threshold of authorities is required to carry out the tallying operation. Nonetheless, this approach usually has cumbersome trust assumptions, such as physical voting booths or untappable channels, which if broken, compromise the election process.

In a mix network context, the shufflers can also share private keys to decrypt ballots, although, because they are opened individually, a malformed ballot will be detected only in the tallying phase, which can disrupt the entire voting process. Notwithstanding, the majority of mixnet-based systems are very robust in the sense that, often, only one or a threshold of mixes must be honest to ensure the voting can proceed correctly. Additionally, both of these voting solutions might still need to assume the existence of untappable channels, if they address coercion-resistance.

Finally, in blind signature approaches, while robustness can be solved by replicating each of the components of the system and guaranteeing that, at least, a set of them works properly, they have to ensure a few complicated trust assumptions, such as anonymous channels. Another security disadvantage is that malformed ballots are only detected in the tallying phase.

Final Remarks. First, all the work presented so far has a strict separation between the voting and the tallying phases, which concerns the fairness property. The usual way to achieve fairness is by saving the encrypted ballot in the bulletin board, and then decrypting it in the tallying phase. Another way to do so, without forcing the voter to be active throughout the entire voting process, is to use an intermediary, which saves the ballots until the tallying phase. The latter is a common procedure among voting protocols based on blind signatures.

In our case, the ballot needs to be posted to the bulletin board immediately after voting, in plain text, and able to counted. This is what we call open-ended voting, which, to the best of our knowledge, no systems have approached yet.

Second, to verify if a voter is eligible to cast a ballot, their identity is checked against the electoral roll. This can be done either in the beginning of the tallying phase or during the voting phase. Then, before counting the ballots, the identities are usually discarded to avoid compromising privacy. This is the most common procedure performed by current voting schemes.

In our scenario, though, such verification, using the plain identity of the voter, cannot be done, as it may jeopardize anonymity. Furthermore, it is infeasible to discard the list of voters, otherwise the system would not be able to detect double voting.

Finally, only a few schemes are designed to support multiple elections running simultaneously [51, 63, 68]. Generally, each election has a credential to be distinguished from others. While this could be a possible approach to solve the problem, we find a simpler and more efficient way to ensure the distinction between voting processes.

3.3 Ring Signatures

The concept of *ring signature* was first introduced by Rivest et al. [97]. It allows an entity to sign a message on behalf of a group, preserving the anonymity of the signer. This notion is associated with the idea of *group signature*, proposed a decade earlier by Chaum and van Heyst [36]. However, ring signatures comprise one considerable difference: there is no central authority, which manages the group and can revoke anonymity. Also, rings can be formed without previous setup and do not require the cooperation of the signers.

This section only reviews the work on ring signatures due to the fully decentralized nature of the EUNOMIA platform [2], which hinders the existence of group managers. Still, since the proposition by Chaum and van Heyst [36], several efficient protocols based on group signatures have been put forward [12, 26, 30, 32]. We now explain the general operation of a ring signature, followed by a brief assessment of the current work, and its correlation with election voting.

Ring signatures work by having the legitimate signer spontaneously select a set of valid and possible signers (including themselves), and then generate a signature, such that no one is able to link back said signature to the real signer, unless all the non-signers collude to reveal the signer. Rather, the verifier only learns that the signature was produced by one of the ring members. This anonymity feature can be useful in many cryptographic protocols, such as e-cash, e-voting and attestation.

While Rivest et al. [97] were the first to formalize the definition of ring signature, a few years earlier, one could build a ring signature based on the proofs of partial knowledge advanced by Cramer et al. [40], combined with the Fiat-Shamir heuristic [54]. This technique was not explicitly called a ring signature, although we refer to it because it carries substantial relevance to the remainder of this work.

The technique by Rivest et al. [97] is a 1-out-of-N ring signature, which means that there is only one valid signer in the ring. Abe et al. [6] also proposed a flexible and efficient 1-out-of-N signature, which was compatible with multiple types of keys. Eventually, new protocols unfolded, which allowed multiple signers in a ring. Bresson et al. [28] and Liu et al. [75] came up with such *threshold*, or K-out-of-N, ring signatures, which yielded time and size efficiency.

We now shift our focus to the voting setting. To vote, the voter appends one ring signature to the ballot, wherein the set of signers must be included in the set of eligible voters. This way, the voter proves to the authority that the vote came from a valid voter, while keeping their privacy. Notwithstanding, one serious problem arises when taking this approach: voters can take advantage of the anonymity property to vote multiple times, as long as all ring members are eligible voters.

Following this concern, a new category of ring signatures emerged. Linkable ring signatures allow a verifier to link two distinct signatures to the same ring member. This idea was first presented by Liu et al. [76], who also proposed a voting system with a registration and voting phase. One year later, Liu and Wong [74] redefined the linkability and anonymity properties of Liu et al. [76], improving the security model. Still, for both proposals, the signatures were only linkable if the same set of keys was used.

The primordial proposal by Liu et al. [76] also entailed a threshold signature, along with their linkable ring signature. However, it was not efficient. Thus, Tsang et al. [104] came up with an efficient threshold linkable ring signature scheme, which, additionally, was able to link signatures produced on behalf of different rings, solving the linkability issue of the two previous approaches [74, 76].

One could also be interested in revoking the anonymity of a signer, assuming they signed multiple times. This concept, known as *traceability*, was first formalized for ring signatures by Fujisaki and Suzuki [58]. The proposed traceable ring signature has a tag, which identifies the voter and the election and, therefore, provides traceability. A few years later, Fujisaki [57] improved the size efficiency and the security model of the previous ring signature [58].

Nevertheless, traceable ring signatures are not adequate for electronic voting, since the preservation of the anonymity of the voter is almost always desired. In fact, this type of ring signature is more relevant in electronic cash, where an authority might be interested in knowing the identity of a double spender.

Chapter 4

Voting System Design

In this chapter, a voting protocol to be integrated in the EUNOMIA system is proposed. Section 4.1 starts by recapping the objectives which we set out to solve. Subsequently, an overview of the EUNOMIA ecosystem is given, and its security model is clarified in Section 4.2. We then list the entities which take part in the solution in Section 4.3, along with the explanation of how their credentials are generated and managed. Afterwards, the architecture of the voting system is presented in Section 4.4, which covers all details related to the protocol. This chapter wraps up with a summary in Section 4.5.

4.1 Objectives

Recalling the overall goal of this work, we want to provide users with a way to vote on trustworthiness properties of posts on social networks, through the EUNOMIA [2] platform. Therefore, we suggest a voting protocol which endorses security, privacy and integrity. We refer to Section 1.2 for the desired properties of this voting protocol, wherein they are listed and briefly explained.

The proposed approach is rather challenging to design. We state the reasons to justify the previous assertion.

- Unlike traditional voting systems, the voting procedure in this context is *open-ended*, which implies that it does not end unless the post is deleted. This raises many complications, one of them being privacy, since ballots are counted immediately after they are cast.
- The system must support multiple voting processes running simultaneously. Usually, every election requires a different credential, so this could turn out to be a scalability issue: having millions of concurrent voting processes with millions of voters would not be practical.
- Every user can cast a ballot on every post at most once, so all users are eligible unless they have voted before. Additionally, no one can know which users have voted. To solve this problem, the majority of protocols discard the list of voters who cast a ballot, after the election. Although, we cannot do such a thing because the process is open-ended.

- The ballot can differ depending on the post the user is voting on. Hence, ballots should have a flexible structure, and support from the simpler, *trust/no-trust* scenario, to the more complex, write-in ballot scenario.

4.2 EUNOMIA Design

In this section, the fundamentals of the EUNOMIA infrastructure are reviewed. Subsection 4.2.1 describes the behavior of the entities which make up the network, the nodes, and subsection 4.2.2 points out the trust assumptions of the platform. This section is relevant because the voting protocol is built upon the EUNOMIA system, therefore relying on some of its services and assumptions.

4.2.1 Nodes

The EUNOMIA ecosystem is made up of a set of nodes called EUNOMIA Services Nodes (ESN). Each group of nodes working together is called a *federation*. In a federation, the nodes coordinate themselves and are responsible for all tasks, ranging from authentication to data management. The ESN is made up of smaller components, and each of them is in charge of a certain service. In addition, nodes are highly flexible and new services can be added with little integration effort. Figure 4.1 portrays a node, its services and other entities which connect to a node.

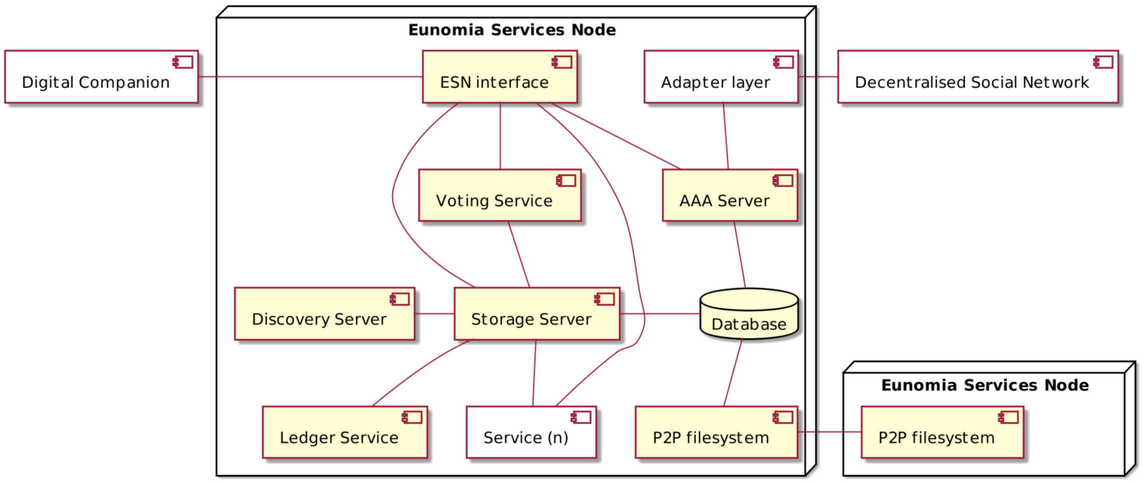


Figure 4.1: EUNOMIA Node Architecture [1].

Some of the parts of the ESN serve as basis for certain functionalities of the voting system. We list, and succinctly explain, the most relevant services in this context.

- **Digital Companion**: Despite not being part of the node, the Digital Companion (DC) is the interface between the users and the ESN, and can communicate with some services. In fact, it corresponds to the user interface, which provides information and collects inputs from users. Furthermore, the DC is crucial to the voting system as it performs relevant cryptographic operations.

- **AAA Server**: Responsible for authentication and authorization of users, this service provides a bridge between the EUNOMIA authentication and the voter registration process. Also, it works as an interface between EUNOMIA and the social network, allowing users to authenticate through their social media accounts.
- **Discovery Server**: This resource plays a significant role in the ecosystem, since it publishes the information of each service. Relevantly, it contains the credentials and metadata necessary for the voting protocol to execute properly.
- **Storage Server**: Securely stores all EUNOMIA data, in a distributed manner (through the P2P filesystem). It keeps all information regarding the voting processes, including ballots and voter credentials. The integrity of the data is ensured by the *Ledger Service*.
- **Voting Server**: Handles all operations related to the voting protocol, from voter registration to counting ballots.

Each ESN runs on top of another node, which belongs to the social networking platform, and, because of its abstraction, it is flexible enough to work with multiple social networks. When accessing their account, users can grant access to their data, which EUNOMIA will extract and process securely. Afterwards, EUNOMIA will be able to provide users with the means to assess the reliability of the content on their social media feed.

We relevantly add that users only communicate with one node. This node provides access to the services and handles their data, which may be synchronized, using appropriate security mechanisms, with other nodes.

4.2.2 Security Model

The security model and trust assumptions of EUNOMIA are now reviewed. We start by assessing the global infrastructure, followed by an analysis of the distribution of information. Lastly, we approach the communication in the whole EUNOMIA environment.

As pointed out in Section 1.1, nodes are controlled by independent and unrelated entities. While this design choice withdraws the need for a centralized, controlling entity, which could impact the trustworthiness evaluation process, it raises some concerns with regard to the security of the system. Generally, we say the overall platform is secure if a subset of nodes of predefined size is trustworthy.

This infrastructure yields significantly more robustness and resilience than a central authority since it eliminates single points of failure. Additionally, it promotes scalability, so, if necessary, the network of nodes can be scaled up to support more users.

As explained in the last section, the digital companion is a part of the EUNOMIA ecosystem, but totally dissociated from the network of nodes. In fact, the digital companion can be considered the device of a user. Because the user may be interested in deceiving the system, and they carry out their actions through the digital companion, we stipulate that the DC is not trustworthy, with respect to the EUNOMIA environment.

Concerning data, we assume the information exchanged between services of the same node is trustworthy. On the other hand, data placed in the distributed storage server, which relies on a potentially hostile peer-to-peer file system accessible by every node, is not secure, and each ESN is accountable for handling the confidentiality and integrity of its own data.

Furthermore, the synchronization of information through the distributed file system is eventually consistent, and therefore assumed to be fast enough for the system not to be vulnerable to race conditions.

Finally, we expect the underlying channels of all communications to ensure the confidentiality and integrity of the data exchanged. This assumption concerns mostly the connection between users and nodes, since node communication is secured by EUNOMIA.

4.3 Entities

In this section, we state the components which make up the voting system. Their functionality is also described, along with how they integrate in the EUNOMIA platform. Subsequently, we explain what type of credential every entity has and how they are generated, distributed, managed and utilized.

- **Voter**: Evaluates online posts by attributing characteristics of trustworthiness. Additionally, they may decide to append their features, such as number of followers or political ideology, to the ballot. In the EUNOMIA ecosystem, the voter is the social network user and carries out the actions through the digital companion (DC).
- **Manager**: Registers new voters, distributes ballots, extracts and verifies user features and manages all voting processes.
- **Tallier**: Verifies the eligibility of the voters and validates, signs and appends ballots to the bulletin board. Also, it issues tallies for each voting process.
- **Anonymizer**: Yields the unlinkability between a ballot and a voter. Multiple anonymizers linked together form a mix network.

The manager, tallier and anonymizer make up the *voting server* component of the EUNOMIA node. Furthermore, the bulletin board is supported by the storage server, but just as an abstraction, that is, the storage server has no knowledge of what belongs to the voting protocol. Also, unlike the usual bulletin boards, we support modification and deletion of data.

These components, along with some relevant EUNOMIA services, are depicted in a diagram shown in Figure A.1 of Appendix A.1.

4.3.1 Credentials

Every entity in the voting protocol has a credential in order to be properly identified. For all parties, we specify the type of credential, how it is generated and distributed, and its functionality.

- **Voter**: Generates a key pair locally, in the digital companion, during the registration process. The public key is sent to the manager, which is responsible for storing, managing and making it available to everyone. This key identifies the voter within the voting system, independently of EUNOMIA, and it is used to verify the eligibility when voting on a post, as described in Section 4.4.2. In the remainder of this work, we refer to the public and private keys of a voter as y and x, respectively.

The next three entities all have the same type of credential, a key pair, along with a public key certificate. Each of these parties generates a key pair and a certificate when the node is first brought up, specifically, when the voting server is initialized for the first time. The private key is saved locally and the certificate is placed in the discovery service, available to the other services and to the voters. The capabilities of these credentials are now stated, for each of the components.

- **Manager**: The key is used to sign web requests, namely, ballot and user features requests, and to sign features when their validation is required.
- **Tallier**: Signs tallies and valid ballots to be stored, and is used to verify the validity of the ballots when issuing a tally.
- **Anonymizer**: Encrypts and decrypts information to build up a mix network.

Additionally, each entity is capable of signing requests sent to another voting server, so the receiver can be sure that such requests are authentic.

Finally, to refer to each of these keys, we define the notation C_i^k, in which C is the first letter of the component, i is the identifier and k is either pk (public key) or sk (secret/private key). Furthermore, the notations $E(m)_k$ and $S(m)_{sk}$ designate the encryption with public or symmetric key k and signature with private key sk, respectively, of message m. For example, $S(m)_{T_1^{sk}}$ is the signature of message m using the private key of tallier 1.

4.4 Architecture

This section advances the architecture of the proposed voting system. We start by explaining how we reached this particular approach, followed by the phases which make up the protocol. Then, the subsequent subsections describe thoroughly the most relevant processes of the architecture.

When designing this system, we took into into account the challenges presented in Section 4.1. Initially, we prioritized the problem of ballot flexibility, as there might be the need to customize the ballot in conformity with the information being evaluated. For this reason, we excluded homomorphic encryption, since it has low flexibility and it would be costly to compute the proofs of correctness for the ballots.

The first proposed solution relied on blind signatures, combined with a mix network to anonymize ballots, after obtaining eligibility. Blind signatures provide the desired flexibility and have a step to verify the eligibility of the user. Furthermore, the technique is mathematically simple and its implementation is quite straightforward. Nonetheless, this approach was eventually dropped for the reasons below.

Integrity. Because the ballot is blinded before being signed, the voter can request eligibility for a certain post, in which they are eligible to vote, but the blinded ballot can contain a different post and still be signed by the manager. On that account, the voter would be able to break the uniqueness property. To solve this problem, there are two solutions: either we issue one credential for each post, which is extremely impractical, or we use partial blind signatures[1], which are more complex to implement.

Efficiency. The crux of EUNOMIA is to eliminate single points of control and failure. While the cryptographic operations might be fast, there would still be the need to distribute tasks among multiple nodes. Thus, the extra step to verify eligibility would constitute the majority of running time of the protocol, which could severely degrade user experience.

Privacy. To verify eligibility, the manager must know whether the user has voted before. However, as noted in the beginning of this chapter, the system can never know which users have voted. The best way to solve this problem would be to make sure the credentials of the voters are unlinkable to the social network users, which must be ensured during the registration phase. Nevertheless, because the voter registration process relies on the EUNOMIA platform, it would be hard to guarantee that one could not trace back the credentials to the user.

While we could have tried to solve these obstacles, the solution would have become outstandingly complex, so we decided to drop this approach and seek a simpler solution.

Afterwards, we started to turn our focus to the eligibility problem, while still keeping in mind the ballot flexibility requirement. To summarize, a solution was needed, which allowed the voter to prove the validity of the following statements together, without revealing absolutely nothing else:

- Their public key is valid and registered.
- They have the corresponding private key.
- They have not voted before on a certain post.

The solution which immediately comes to mind to attain the first two statements is a ring signature. It would allow the voter to prove that they know the corresponding private key of a public key, which is included in a set of valid keys. To meet the third requirement, we would need to add linkability to the ring signature. Nevertheless, this linkability property needs to be customized to link only voter and post together, and never either of them separately, that is, it must be unlinkable for the same user and different posts. Furthermore, it should work independently of the keys used in the ring.

Hence, we propose a voting system which relies on a ring signature to provide eligibility. The uniqueness property is achieved through a pseudonym, which is thoroughly explained in Section 4.4.2. We provide anonymity by implementing a mix network between the voter and the tallier, which is only optional when voting, and left at the discretion of the user.

[1]Partial blind signatures [4, 5] allow a signer to include relevant information in the blinded content being signed. One pertinent example is setting an expiration date for a signature, without knowing the message.

Before proceeding to the voting protocol, we highlight that it is mandatory for the user to register first as a voter. While the registration phase is not included in the voting protocol below, it is described in Section 4.4.1. We now enumerate the phases of the proposed solution:

1. **Preparation**: The user requests a ballot and, optionally, their features, in separate requests. Note that ballots could have already been requested by the user interface, automatically.
2. **Voting**: The user makes their choice and the digital companion generates the ring signature and appends it to the ballot. Additionally, they decide whether to attach some or all of their features to the ballot.
3. **Anonymization**: The user decides whether to vote privately or publicly. Should the user prefer to keep their anonymity, the ballot is sent through the mix network. Otherwise, it is sent directly to the tallier.
4. **Validation**: The tallier receives the ballot in plain text, with the ring signature inside, and gathers a threshold of signatures from other talliers, which are conditional on the validity of the ring signature, and on the uniqueness and wellformedness of the ballot. If the user has appended features, they are sent to a threshold of managers, which validate them and return them signed. Upon receiving all responses, the tallier verifies the signatures of the features, discards them, and posts the ballot, along with the features and the ballot signatures, to the bulletin board.

The next subsections explain comprehensively the procedures encompassed in this protocol, except the preparation phase, which is a simple request of a ballot and features. Section 4.4.2 tackles the details of the voting and the validation phases, describing the pseudonyms, the ring signature and the validation of user features. Then, the anonymization technique is discussed in Section 4.4.3.

Still in this section, the management of ballots, after the protocol, is addressed, namely, verifying and tallying, which are studied in Sections 4.4.4 and 4.4.5, respectively.

After careful review of each of the aforementioned subsections, the whole protocol is depicted in a sequence diagram shown in Figure A.2 of Appendix A.2. Note that, in this sequence diagram, the use of features is always optional.

4.4.1 Registration

The voter registration process is required for the user to vote. This process is carried out only once, preferably when the user is signing up with EUNOMIA. The main purpose of this procedure is to generate credentials for a voter. Despite only being part of the voting component, the registration relies heavily on EUNOMIA authentication. In any case, the rest of voting protocol runs independently of EUNOMIA authentication.

The registration process flows as follows:

1. The digital companion generates a key pair, which identifies the voter.
2. The public key, along with the EUNOMIA user identifier, are sent to the manager.

3. The manager verifies, through the EUNOMIA authentication and storage services, if the user has registered before as a voter.
4. If the user has not registered before, the manager saves the public key in the storage service, making it available to everyone, and unlinked to the user identifier.

With the help of EUNOMIA authentication, it is guaranteed that anyone who is not registered in the EUNOMIA platform, and that users who have already generated credentials for the voting component cannot issue new credentials, otherwise, the same user would be able to vote multiple times with distinct credentials. Moreover, no node, besides the one which handles the registration process, can correlate a public key with a social network user.

Finally, note that it is assumed that the registration process is performed honestly, as explained in Section 6.1.1.

4.4.2 Voting and Validation

This section approaches the voting and the validation steps of the protocol. After receiving a ballot, the user makes their choice, and the digital companion calculates the pseudonym and generates a ring signature. Additionally, the user can choose to append some of their features to the ballot. The tallier then validates the ring signature and confirms, through the pseudonym, that the ballot is unique. We explain all of these processes in the following subsections.

Voter Pseudonyms

We use the notion of *pseudonym*, which allows users to hide their identity from the system, when voting. Pseudonyms were first used in the context of attestation [29] and they provide the linkability we require. This linkability property solves the eligibility problem, avoiding frauds such as double voting.

Our pseudonym is determined using two parameters, namely, the private key of the voter and the identifier of the post. Relevantly, it is deterministic, which implies that, for the same voter and post, the pseudonym does not change. The definition of a pseudonym is now formalized. We denote $\mathbb{G}$ as a prime order group and define $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ as a cryptographic hash function. The pseudonym is calculated as

$$nym = H_1(postId)^x$$

where $postId$ is the identifier of the post and $x \in \mathbb{Z}$ is the private key of the user.

Note that no user can compute pseudonyms of other users unless they hold their private key. Furthermore, due to the discrete logarithm assumption, explained in Section 2.2, the pseudonym is hard to reverse, so the identity of the user is preserved.

Notwithstanding, one serious problem arises when using pseudonyms: the system does not know whether they were calculated using the adequate parameters, which, if not, may deceive the system into believing that a user has not voted before when, in fact, they did. Therefore, to prove that a pseudonym is correctly formed, we append it to the ring signature. This process is clarified in the next subsection.

Ring Signature

In this section, the ring signature used to ensure eligibility is tackled. We start by highlighting the logic behind it and, only then, explaining the signature generation and verification processes.

The ring signature we propose is, in fact, a signature proof of knowledge. We refer to it as ring signature because its structure is comparable to one and because it behaves similarly to a linkable ring signature [76, 104].

Our ring signature scheme is formalized as

$$SPK\left\{(x_1,...,x_n) : \bigvee_{i=1}^{n} \left(y_i = g^{x_i} \wedge nym = H_1(postId)^{x_i}\right)\right\}(B)$$

wherein ballot B, which is a bit string, is signed, while proving knowledge of values $x_1,...,x_n \in \mathbb{Z}$, which are private keys. The element $g \in \mathbb{G}$ is predefined and used to generate the public keys from the private keys. The public keys $y_1,...,y_n \in \mathbb{G}$, which form a set $\mathcal{Y}$, are randomly chosen from the set of keys of registered voters. Each of these public keys y_i has a corresponding private key x_i'. Because the voter only knows their private key x_j, where $j \in \{1,...,n\}$, the other private keys x_i are simulated by the protocol.

We say the ring signature has size n when the set of public keys $\mathcal{Y}$ contains n elements. Finally, everything in the protocol, including ballot B, is common knowledge, except the values being proved.

This construction uses the Chaum-Pedersen protocol [34], explained in Section 2.4.1, to ensure the validity of two statements, in a conjunction:

- The private key x_i generates a registered public key y_i.
- The pseudonym nym is computed with private key x_i.

The ring signature is a disjunction of n of these conjunctive statements. Due to the proofs of partial knowledge by Cramer et al. [40], clarified in Section 2.4.1, one can prove that, at least, one of the conjunctive statements is valid, revealing nothing else.

We now proceed to the generation and verification of the ring signature. For a matter of simplicity, we assign $h := H_1(postId)$, where H_1 is the cryptographic hash function referred in the previous section, which maps a string to an element of group $\mathbb{G}$. We also define $H : \{0,1\}^* \rightarrow \{0,1\}^*$ as a cryptographic hash function. Remind that set $\mathcal{Y} = \{y_1,...,y_n\}$ and elements g and nym all belong to the prime order group $\mathbb{G}$ of order l.

In addition, we chose to make this proof non-interactive, which not only does not require honest verifiers [45], a condition we could not guarantee, but is also, in our context, significantly less time consuming. The protocol works as follows:

1. The voter, through the digital companion, calculates their pseudonym $nym = h^x$ and randomly picks from $\mathbb{Z}_l^*$
   - k_i, for all $i = 1, 2, ..., n$
   - c_i, for all $i = 1, 2, ..., n, i \neq j$

2. In the next phase, the voter calculates the following commitments:

$$r_i^{key} := \begin{cases} g^{k_i} & , i = j \\ g^{k_i} \cdot y_i^{-c_i} & , i \neq j \end{cases} \qquad r_i^{nym} := \begin{cases} h^{k_i} & , i = j \\ h^{k_i} \cdot nym^{-c_i} & , i \neq j \end{cases}$$

   Note that the voter only knows their private key x_j and, therefore, fakes the protocol for all the private keys they don't know.

3. Then, resorting to the Fiat-Shamir Heuristic [54], the voter generates a challenge c based on the common knowledge. We emphasize that challenge c includes ballot B, which makes the scheme a ring signature, rather than a proof of knowledge. The challenge is defined as

$$c := H(r_1^{key} \parallel ... \parallel r_n^{key} \parallel r_1^{nym} \parallel ... \parallel r_n^{nym} \parallel y_1 \parallel ... \parallel y_n \parallel nym \parallel B).$$

   Because the voter only knows one of the private keys, only one of the conjunctive statements is true. While the challenges for the fake statements were all generated randomly in step 1, a valid challenge for the legitimate statement still needs to be computed:

$$c_j := c - \sum_{i=1,\ i \neq j}^{n} c_i \ (mod\ l)$$

4. The protocol proceeds to the last phase, the response. The following assignment determines the correct responses s_i, for all $i = 1, 2, ..., n$.

$$s_i := \begin{cases} k_i + c_i \cdot x_i \ (mod\ l) & , i = j \\ k_i & , i \neq j \end{cases}$$

5. The voter sends the commitments $\forall_{i=1}^{n} r_i^{key}, r_i^{nym}$, the challenges $\forall_{i=1}^{n} c_i$, the responses $\forall_{i=1}^{n} s_i$, the pseudonym nym and the set $\mathcal{Y}$ of public keys to the verifier, which is the tallier. All of these components make up the ring signature, which is sent inside the ballot.

6. Finally, the tallier confirms that the next conditions hold:

   (a) All public keys used in the ring signature belong to registered and valid voters.

   (b) The voter pseudonym nym has not been used before.

   (c) After computing the challenge c using the common knowledge, verify that

$$c = \sum_{i=1}^{n} c_i \ (mod\ l).$$

(d) For all $i = 1, 2, ..., n$, the following statements hold:

$$r_i^{key} = g^{s_i} \cdot y_i^{-c_i} \qquad\qquad r_i^{nym} = h^{s_i} \cdot nym^{-c_i}$$

If all of the previous requirements verify, the ring signature is valid and the voter is allowed to cast the ballot.

We further develop on the ring signature in Section 6.1.3, which formalizes the scheme and proves it holds on to its properties.

Validating Features

The possibility of appending user features, such as number of followers, to a ballot, when voting, can help users make more conscious conclusions regarding the trustworthiness of content. Initially, one might think that the features could be validated and signed when they are requested, in the preparation phase. However, this may lead to serious problems, such as the theft or purchase of features from other users, or even the loss of ballot privacy.

Therefore, it must be done in the validation phase. Nonetheless, validating features without compromising ballot secrecy is no trivial task. In fact, features cannot be validated by a node which is already verifying the ballot, as it would be possible to link a vote to a user.

To solve this problem, we took advantage of the decentralized setting wherein this project is implemented. That is, we resort to other nodes, which have no knowledge about the ballot, to validate user features. This implies that, when the ballot is received by the tallier, the features, and the respective user identifier, must be encrypted to ensure privacy. However, the voter must not be able to choose with which key to encrypt the features, since it may lead to collusion.

Therefore, we suggest a (t, n) threshold scheme [102], in which the public key belongs to a federation of n nodes, and the private key is distributed among the nodes. To perform any private key operation, a threshold $t = 2$ of managers (or nodes) must cooperate. Whenever a new node joins the federation, it is responsible for generating a new key pair in a distributed manner. We refer to the public key of the federation as the *validation key* (or V_k, as noted in the sequence diagram of Figure A.2).

Threshold schemes are not new to this work. In fact, some voting protocols [3, 14, 42, 65], presented in Section 3.2, resort to these schemes when decrypting votes in the tallying phase. Generally, we define two algorithms for a threshold system: *key generation* and *decryption*.

Key generation is a delicate process in the sense that there should not be a centralized node, which distributes the shares of the private key. Instead, all nodes must engage in a joint protocol, such that no single entity is capable of reconstructing the private key alone. To achieve this, we follow the distributed key generation protocol set forth by Pedersen [90]. Informally, each node computes a share x_i of the private key x, which can be reconstructed using t shares.

The decryption algorithm, described by Desmedt and Frankel [47], works by having entities compute their decryption share w_i, along with a zero-knowledge proof that this share is correctly computed. Then, a node receives all the required shares $w_1, ..., w_t$, and multiplies all shares together to fully decrypt the message, without revealing the private key.

Feature validation is part of the validation phase, and it is done concurrently with ballot verification. This procedure works as follows:

1. Still in the voting phase, the user encrypts their features, along with their user identifier, using the validation key V_k, places the resulting ciphertext in the ballot and votes.
2. The tallier receives the ballot and forwards the encrypted features to the manager, in the same node, which computes their decryption share w_1 of the features ciphertext.
3. The manager sends their share w_1 and the encrypted features to a threshold of other managers, each of which uses the received partial decryption w_1 and their decryption share w_i to fully decrypt the features.
4. Once the features are decrypted, each manager validates them, according to the user identifier.
5. If they are valid, each manager returns the features to the original manager, fully decrypted and signed.
6. The manager validates the received signatures, discards them, and forwards the features to the tallier, to be appended to the ballot.

We point out a few considerations regarding feature validation. First, due to the continuous tallying requirement of the voting protocol, dealing with features when voting raises many privacy concerns. In fact, one can never be sure that the features will not be correlated with the ballot, once it is stored.

Second, this proposal is very high level, and based on the algorithms developed in the referred work. Third, the implementation is rather tricky and complex so, since the main focus was the development of the voting system, we have decided to leave it as future work and not to approach it in the evaluation.

4.4.3 Anonymization

Privacy is one of the greatest concerns of every voting system. In this setting, anonymity becomes even more difficult to achieve due to the open-ended, continuous-tallying scenario. While we solve part of the problem by using pseudonyms to hide the identities of the users, there still exists the chance to correlate users and ballots through the internet connection, or even using timing analysis.

Therefore, a mix network was implemented between users and talliers, which randomly delays and shuffles the ballots received, acting as an anonymous channel. The mixnet provides protection against passive eavesdroppers and ensures anonymity even if some mixes are malicious. We highlight that users can opt out of this anonymization process and send the ballot directly to the tallier, or even use another anonymous channel, such as Tor [50]. Still, the main advantage of using ours is that it requires no additional effort.

This mix network is similar to the decryption mixnet proposed by Chaum [37]. It is a *low-latency* mix network, composed of a set of anonymizers placed sequentially. Each anonymizer (or mix) receives a ballot, removes a layer of encryption, holds the ballot for some time while waiting for others, scrambles the ballots and, only then, forwards them to the next anonymizer or to the tallier. This hinders traffic correlation attacks.

We now formalize the anonymization scheme. As mentioned before, each anonymizer A_i has a public key, which we refer to as A_i^{pk}. The ballot is represented as B and, for each layer of encryption, a random symmetric key K_i is generated. Finally, remind that $E(m)_k$ is a function which encrypts message m using a symmetric or public key k.

To start the anonymization process, the user randomly picks a list of n anonymizers and computes the initial message

$$E(K_n, A_{n-1})_{A_n^{pk}},\ E(E(K_{n-1}, A_{n-1})_{A_{n-1}^{pk}}, ..., E(E(K_1)_{A_1^{pk}}, E(B)_{K_1})_{K_2}...)_{K_n}.$$

The first anonymizer A_n receives the message and decrypts the first part of the message to retrieve the symmetric key K_n and the identifier of the next anonymizer A_{n-1}. Then, the same anonymizer A_n removes the first layer of encryption using key K_n and forwards the decryption to the next anonymizer A_{n-1}. This process is repeated until the message reaches the last anonymizer A_1. When it does, the message should be

$$E(K_1)_{A_1^{pk}}, E(B)_{K_1}$$

which can be trivially decrypted by A_1, obtaining the ballot B and, then, passing it on to the tallier for validation.

We underline that this mix network provides no proof of correct operation, which we disregard since it carries no relevance to the voting protocol. In fact, verifiability is achieved through other means, discussed in Section 6.2, and the protocol allows users to send the same ballot through the mix network as many times as they want.

Another relevant consideration is that everyone can send data through the mix network, without authentication. Therefore, to avoid spamming, or even denial of service, the first anonymizer, which is in contact with the user, authenticates the user, using the authentication service of EUNOMIA, before sending the ballot through the anonymization circuit.

Finally, all information necessary to build a mix network, including public keys and identifiers, is available on the discovery service.

4.4.4 Verifying and Deleting Votes

In this context, it is appropriate to allow users to check if their ballot was cast, and if it represents correctly their intentions. Through the pseudonym, which no one besides the owner of the private key can compute, we allow users to retrieve their previously cast ballot. While we do not approach receipt-freeness in this proposal, we acknowledge that the possibility to verify previous votes leads to the loss of that property.

Additionally, users can delete past ballots. These are removed using the pseudonym as well. We point out that being able to delete ballots can imply coercion-resistance, since a voter can vote in the presence of the coercer and later change their vote alone. Nevertheless, the coercer can learn the pseudonym and compromise their privacy for that post. Still, because we assume there is low coercion in this scenario, we do not develop on this topic further.

Finally, the manager is the entity responsible for carrying out both the verification and the deletion of ballots from the bulletin board.

4.4.5 Tallying Votes

Tallies are updated continuously as the results need to be shown to the users as close as possible to real time. The tallying process is simple and requires small computational effort. Furthermore, anyone can request a tally for a post at any time. The tallying process runs as follows:

1. A tally is requested and the tallier gets all ballots for a specific post from the bulletin board.
2. The tallier requests, from the discovery service, the certificates of all the talliers which signed the ballots returned. These are then saved in a cache to expedite future tallying processes.
3. All ballots are verified, one by one, through the signatures appended to them during the validation phase. Ballots are valid as long as a threshold of signatures is valid. Invalid ballots are discarded and removed from the bulletin board.
4. The tally is signed and returned.

Regard that it is not the responsibility of the voting system to sort out the votes in a certain way. In fact, the system is totally unaware of the context, and simply returns a list of authentic ballots, cast by eligible voters. Only afterwards, the user interface, independently of the voting protocol, arranges the votes as desired, while taking into account the impartiality of EUNOMIA in the trustworthiness evaluation process.

4.5 Summary

To summarize, we propose an open-ended voting protocol, which features accuracy, privacy and verifiability. The system is built upon the decentralized infrastructure of EUNOMIA and it is applied to the context of information credibility evaluation, by allowing users to vote on trustworthiness properties of posts on social media.

The proposed ring signature scheme, along with the pseudonym, provide the system with the means to assess the eligibility of users, without compromising their privacy. We also suggest a technique to enable users to append their features to a ballot, while maintaining some privacy assurances. Furthermore, a simple mix network is developed, which prevents an adversary from correlating a ballot with a user, in the voting phase.

In addition, we provide users with a way to verify and delete their votes, and designed a simple and straightforward tallying process.

In the end, the system complies with the objectives put forward in Section 4.1, and provides an efficient way to vote in an open-ended, continuous-tallying environment which, to the best of our knowledge, did not exist before.

Chapter 5

Implementation

This chapter gives out the details of how the system was implemented. We start by tackling the cryptosystems used to generate the credentials referred in Section 4.3.1. Then, considering the entities which take part in the protocol, our implementation is divided into two segments: the *voting server* and the *client*. The details of the implementation for both of them are provided in Sections 5.2 and 5.3, respectively, along with some code and data examples. Furthermore, in the client section, an explanation and statistics are presented for a pilot testing experiment.

5.1 Credentials

In this section, we extensively describe the credential system for the entities involved in the protocol. We start by listing the parties and stating the cryptosystems used. Then, subsection 5.1.1 explains how the credentials are managed and accessed.

- **Manager, Tallier and Anonymizer**: During the first initialization process, all of them generate distinct 2048-bit RSA [96] key pairs. Then, they issue self-signed certificates to certify the knowledge of the generated private keys. These certificates are used for testing purposes and they are supposed to be replaced by certificates signed by a certificate authority.
- **Voter**: When registering with the voting system, the voter generates a key pair using the twisted Edwards form of the elliptic curve Curve25519 [23, 24]. The public key is a point in the curve, which, if compressed[1], has 256 bits. The private key also has 256 bits, and it is a scalar value. The public key Y is generated by calculating xG, where G is the predefined base point of the curve [24] and x is the randomly generated private key.

The choices of cryptosystem and key lengths were based on the security standards by NIST [13] and SafeCurves[2], while also taking efficiency into account.

[1] Point compression in elliptic curves is a technique which reduces the size of the representation of points to save memory. The usual procedure is to use only one of the coordinates to represent the point, and then append the sign bit of the other coordinate.

[2] `http://safecurves.cr.yp.to` (accessed December 31, 2020)

5.1.1 Management

The credential management process is now outlined. Starting with the manager, the tallier and the anonymizer, the management of their credentials is fairly straightforward. In fact, there exists only one step: once the first initialization is completed, their certificates are placed, in X.509[3] format, in the discovery service, associated with the identifier of the voting server. The certificates are never deleted, even if the node is removed from the federation, otherwise some ballots would become invalid. The private keys are saved locally in a file during the generation process.

Concerning the voter, their public key is sent to the manager during the registration process, and it is saved in the storage service in *Base64* encoding, available to everyone. The private key, besides being saved in the local storage of the digital companion, is also stored remotely, in the storage service. However, it is not saved in plaintext, instead it is encrypted first. This procedure relies on the Password-Based Key Derivation Function 2 (PBKDF2) [80] to generate a 128-bit symmetric key, using a user chosen password, a random salt and 10000 iterations. This key then encrypts the private key using AES [44] in cipher block chaining (CBC) mode. The private key is kept in storage, encrypted, and can be requested whenever needed.

5.2 Voting Server

The voting server is a service which runs inside the EUNOMIA services nodes. The design practices of EUNOMIA promote the development of fully decoupled services, that is, each service provides an Application Programming Interface (API), which works as an abstraction layer to the other services. The voting server is no exception, so we specified an API to deliver the functionality of the voting server to the rest of the EUNOMIA ecosystem.

This component is implemented as a Java [11] application, and runs an embedded web service. This web service is powered by the Grizzly[4] and Jersey[5] frameworks and it provides the functionality for the aforementioned API. We used Maven[6] to compile, package and execute the project, and JUnit[7] to conduct unit tests.

As referred before, the voting server encompasses the manager, the tallier and the anonymizer. Since it would not be practical to implement them in separate machines, we have segregated the logic of each component within the same application. Nevertheless, they all still rely on a library we developed, which contains the vital functions for any service to work, ranging from storage to authentication.

Relevantly, EUNOMIA nodes are built on top of social network nodes. To run initial tests, we have used Mastodon[8], which is a distributed social networking protocol. Owing to the design principles of EUNOMIA, other social networks can be integrated with slight effort. Additionally, given the necessity

[3]`https://www.itu.int/rec/T-REC-X.509` (accessed December 31, 2020)
[4]`https://javaee.github.io/grizzly` (accessed December 31, 2020)
[5]`https://eclipse-ee4j.github.io/jersey` (accessed December 31, 2020)
[6]`https://maven.apache.org/index.html` (accessed December 31, 2020)
[7]`https://junit.org` (accessed December 31, 2020)
[8]`https://joinmastodon.org` (accessed December 31, 2020)

for modularity of the components, all services run in the same machine, but are virtually isolated from each other using Docker[9].

In the next sections, some topics related to the voting server are approached. First, the libraries on which it relies to perform some operations are listed. Then, we explain how it is initialized, followed by the voter registration and anonymization processes. Finally, the ballot structure is approached.

Libraries

The voting server relies on some external libraries to perform some actions, ranging from cryptographic operations to communication. We list those libraries below and explain what they are used for, in this context.

- ***BouncyCastle***[10]: Generate, sign and read X.509 certificates.
- ***EdDSA-Java***[11]: Provides the parameters and the operations for the twisted Edwards Curve25519 [23, 24], which the system uses to verify the ring signatures and pseudonyms and to validate public keys.
- ***OkHttp***[12]: Handles all communications of the voting server with other services.
- ***Gson***[13]: Parses and generates JSON [91] code for the data exchanged with all services and the digital companion.

Initialization

Each voting server, when launched, goes through an initialization process. In this process, three environment variables define the crucial parameters for the service to execute correctly, which are:

- **VOTING_ID**: the universally unique identifier (UUID) of the voting server, which distinguishes it from the remaining ones in the same federation;
- **VOTING_TOKEN**: the token which identifies this component in the EUNOMIA authentication service, preventing unauthorized services from pretending to be this voting server;
- **AUTH_PROVIDER**: address of the social network authentication provider.

Then, the voting server checks if a file containing the certificates and private keys of each component already exists. If this file does not exist, new credentials are generated and posted to the discovery service. This file, along with the environment variables, facilitate the portability of a voting server between machines. Finally, to conclude the initialization, the voting server performs a few operations to register with the other services in the node, such as authentication.

[9]`https://www.docker.com` (accessed December 31, 2020)
[10]`http://bouncycastle.org` (accessed December 31, 2020)
[11]`https://github.com/str4d/ed25519-java` (accessed December 31, 2020)
[12]`https://square.github.io/okhttp` (accessed December 31, 2020)
[13]`https://github.com/google/gson` (accessed December 31, 2020)

Voter Registration

The voter registration process is carried out with support from the authentication service, and it was designed to be performed concurrently with the user registration in EUNOMIA. This procedure is described in Section 4.4.1. In this section, the technical details of the implementation are addressed.

The simplified code for the voter registration is shown in Figure 5.1.

```java
public boolean register(String token, String userId, String publicKey, String privateKey) {
        if (!Cryptography.ECC.validatePoint(publicKey))
                return false;

        if (!Authentication.verifyUser(token, userId))
                return false;

        // check if voter already exists
        if (!Storage.get("properties.user_id", "eq", userId).isEmpty())
                return false;

        if (Storage.save(userId, publicKey, privateKey))
                return true;

        return false;
}
```

Figure 5.1: Sample code, in Java, for the voter registration process.

After the digital companion generates the Ed25519 key pair, it is sent to the voting server. The public key is verified in the `validatePoint()` method, which confirms that it is a valid point in the curve. Then, resorting to the EUNOMIA authentication service, it verifies if the user is legitimate using the `verifyUser()` function.

Subsequently, the voting server issues a request to the storage server to check if a voting credential for that user already exists. If this condition checks out, the `save()` method is called, which stores the key pair: the public key in *Base64* and the encrypted private key, also in *Base64*, along with the initialization vector (IV) and the salt, which are contained in the `privateKey` variable.

When the user does not have the private key locally, it is requested to the voting server, in a procedure similar to this one. With help from the `verifyUser()` method, the voting server verifies if the user is the owner of that private key, and, if this condition validates, calls storage using `get()` to retrieve the private key and return it to the user.

Anonymization

In this section, we sort out the implementation details of the anonymization process. As mentioned in Section 4.4.3, a mix network is built to simulate an anonymous channel. The work of each anonymizer as a mix is quite simple: there is a predefined time interval, after which ballots are randomly shuffled and sent to the next anonymizer. Moreover, the anonymizer is constantly receiving ballots, which are immediately decrypted and placed in a queue to be dispatched.

By default, each circuit of anonymization has three anonymizers and the time interval is set at five seconds. The parameters are customizable by the user and the by the node administrator, respectively, to fit the desired requirements. One relevant aspect is that the circuit is built by the client, with whichever anonymizers they would like to use.

Each package received by an anonymizer is a dictionary with the following keys:

- **data**: Symmetrically encrypted using AES-CBC and a random 128-bit key, it is the information sent to the next anonymizer or to the tallier, which is decrypted first.
- **info**: Encrypted using RSA and the public key of the receiving anonymizer, it contains relevant information, structured in a dictionary, including the symmetric key and the initialization vector (IV) to decrypt the **data** value and the identifier of the next anonymizer.

Because packages are successively encrypted, when the **data** key is decrypted to be forwarded to the subsequent anonymizer, the result of the decryption will have this exact same structure. When the last anonymizer decrypts the **data** value received, the result will be a ballot, which is sent to the tallier to be validated.

Ballots

As initially proposed, the ballots in our protocol are fully flexible and independent of the operation of the system. The ballot is constructed using JSON [91] and this is the only format supported. Figure 5.2 shows one possible valid ballot, built in JSON. It does not show neither the proof (ring signature) nor the tallier signatures as they would take up too much space unnecessarily.

```
{
    ”post_id”: ”abc”,
    ”votes”: {
        ”trust”: 1,
        ”no_trust”: 0
    },
    ”features”: {
        ”followers”: 1234
    },
    ”proof”: {...},
    ”signatures”: {...}
}
```

Figure 5.2: Example of a valid ballot in JSON format.

A ballot is a dictionary, which follows a strictly defined structure to be well-formed. The two essential keys are the `post_id` (post identifier) and the `votes`. The latter contains the choices of the user. Figure 5.2 depicts the simplest example, which is a trust/no-trust vote. Nonetheless, this value can be customized to fulfill the needs of different posts, which may include open questions. The `features` key is optional and holds the features which the user decided to append to the ballot.

Finally, for the ballot to be valid, it must include the ring signature computed by the user (`proof`) and the signatures of the talliers (`signatures`). The last-mentioned is a dictionary, which matches the identifier of the voting server to a signature of the `post_id`, the `votes` and the optional `features`.

5.3 Client

The client provides the functionality required from a user, with regard to the voting protocol. As referred in Section 4.2, the digital companion is an interface running on a device, such as a smartphone or a computer. Similarly to the EUNOMIA node, the functionality of the digital companion can be seen as modular, so the client works as a external library integrated in the DC. It does not provide a web API, instead functions can be invoked directly after importing this library.

This component is implemented in Javascript [55], which can be easily executed by any browser, using a Javascript engine. When the code is executed outside the browser, Node.js[14] is used as runtime environment. In fact, we set up the client library to work primarily with Node.js, which simplified the development process remarkably, and allows portability to other settings, such as mobile applications.

The client library is logically divided into three segments. The first one provides cryptographic support for the remainder of the client, which comprises encryption, decryption, signature verification and operations on elliptic curves. The second includes functionality related to local storage, registration of voters and communication with storage and discovery services. Finally, the last one implements the voting operations and works as an abstraction layer to be called by the user interface.

We approach parts of the technical implementation of this segment of the project in the next subsections. First, the external libraries and their respective functionality are listed. Then, the pseudonym generation is described, followed by an example of how the library is used by the user interface. Finally, Section 5.3.1 gives an overview of the initial testing of the entire project.

Libraries

Like the voting server, the client library also depends on a few frameworks to provide some functionality. The most relevant libraries used are enumerated below and their utility is clarified.

- ***Elliptic***[15]: Generates public keys using the twisted Edwards Curve25519 [23, 24], and provides the parameters and the cryptographic support to compute the ring signatures and the pseudonyms.
- ***Forge***[16]: Provides support for all AES [44] and RSA [96] cryptographic operations, which includes encryption, signature verification and random value generation.
- ***Axios***[17]: Handles all network communications to the voting server.
- ***Level***[18]: It is a key-value storage based on LevelDB[19], which is used to store and manage data

[14]`https://nodejs.org` (accessed December 31, 2020)
[15]`https://github.com/indutny/elliptic` (accessed December 31, 2020)
[16]`https://github.com/digitalbazaar/forge` (accessed December 31, 2020)
[17]`https://github.com/axios/axios` (accessed December 31, 2020)
[18]`https://leveljs.org` (accessed December 31, 2020)
[19]`https://github.com/google/leveldb` (accessed December 31, 2020)

locally, such as previous ballots, voting server information and voter credentials.

Pseudonyms

The pseudonyms are one of the fundamental parts of the voting protocol. As described in Section 4.4.2, the pseudonym is deterministic and calculated using the post identifier and the private key of the voter. The sample code for this process is shown in Figure 5.3.

```javascript
function computeNym(privateKey, postId) {
    // get hashed point and calculate nym
    let H = ecc.hashToPoint(postId);
    let nym = H.mul(privateKey);

    return nym;
}
```

Figure 5.3: Sample code, in Javascript, for the calculation of a pseudonym.

As one can notice, the calculation of the pseudonym is simple and straightforward. In fact, one just needs to hash, using `hashToPoint()`, the `postId` to a point in the subgroup of points generated by the base point of the Ed25519 curve, and then multiply the result by the private key. Note that this code follows the elliptic curve notation.

Voting Example

We now provide an example of how the client library can be used by other parties. Figure 5.4 shows a code workflow, which takes the required steps to perform a voting operation. Also shown in Figure 5.4 is the request for a tally, for the same post the user voted on.

```javascript
const lib = require("lib.js");

lib.initialize(token, userId, secret).then(r => {
    return lib.getBallot(token, postId, userId);
}).then(r => {
    // update ballot accordingly
    let ballot = r.message.ballot;
    ballot.votes.trust = 1;

    return lib.vote(token, userId, ballot, anonymous = false);
}).then(r => log(r)) // voted successfully
.catch(e => log(e)); // handle errors

lib.tally(postId).then(r => log(r)) // output tally
.catch(e => log(e)); // handle errors
```

Figure 5.4: Sample code workflow for voting, in Javascript.

It can be seen that carrying out a voting operation takes very few lines of code. It consists of three operations: `initialize()` registers or retrieves credentials for `userId`, `getBallot()` requests a new ballot for post with identifier `postId`, and `vote()` performs the voting action, which includes issuing a ring signature, preparing the ballot for anonymization (if needed) and sending it to the voting server. Finally, a tally request for `postId` is also executed using the `tally()` function.

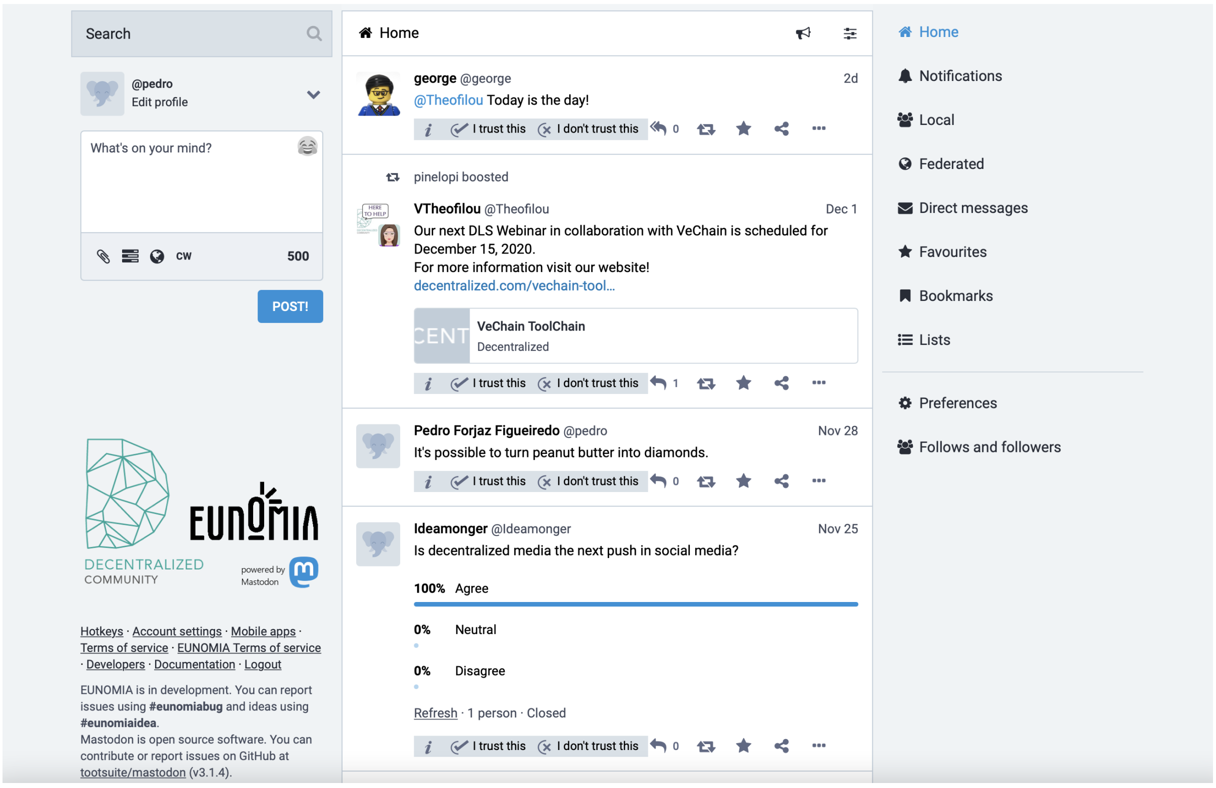
For every request, while the responses are logged in this example, they can be handled as desired by the user interface. The client library makes it remarkably simple to perform any call to the voting server, which makes the voting protocol easy to use.

5.3.1 Decentralized EUNOMIA

Decentralized EUNOMIA[20] was designed as a proof of concept for the EUNOMIA platform. The goal of this experiment was to test the full functionality of the EUNOMIA Project, which includes voting. It was built on top of the above-mentioned Mastodon social networking protocol.

Some metrics were collected throughout a period of nine days. A total of 286 users signed up, with an average of 232 daily active users. The system received around 80000 tally requests and about 5000 ballots were cast.

The Mastodon platform provides a user interface very similar to the well-known social network Twitter[21]. There is a news feed, composed of posts written by users. Figure 5.5 shows an example of this social networking feed.



george @george 2d
@Theofilou Today is the day!

pinelopi boosted
VTheofilou @Theofilou Dec 1
Our next DLS Webinar in collaboration with VeChain is scheduled for December 15, 2020.
For more information visit our website!
decentralized.com/vechain-tool...
VeChain ToolChain
Decentralized

Pedro Forjaz Figueiredo @pedro Nov 28
It's possible to turn peanut butter into diamonds.

Ideamonger @Ideamonger Nov 25
Is decentralized media the next push in social media?
100% Agree
0% Neutral
0% Disagree
Refresh · 1 person · Closed

Hotkeys · Account settings · Mobile apps · Terms of service · EUNOMIA Terms of service · Developers · Documentation · Logout

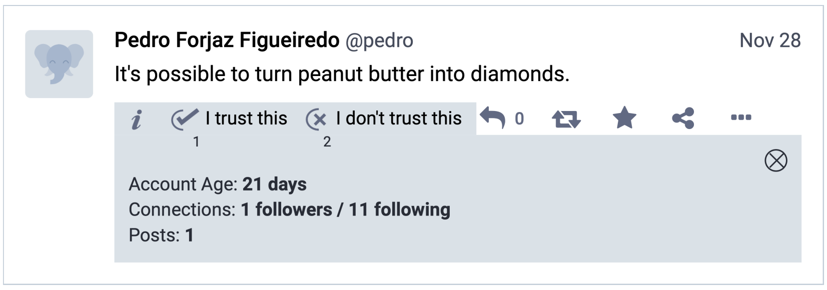
EUNOMIA is in development. You can report issues using **#eunomiabug** and ideas using **#eunomiaidea**.
Mastodon is open source software. You can contribute or report issues on GitHub at tootsuite/mastodon (v3.1.4).

Figure 5.5: News Feed of Decentralized EUNOMIA.

Users interact with EUNOMIA through this interface. The list of posts is picked automatically for each

[20]`https://decentralized.eunomia.social` (accessed December 31, 2020)
[21]`https://twitter.com` (accessed December 31, 2020)

user. EUNOMIA then provides tools to evaluate the information on these posts, including the source and respective modifications, and the possibility of voting on the trustworthiness of a post. Figure 5.6 depicts a post of this feed with more detail.


Pedro Forjaz Figueiredo @pedro Nov 28

It's possible to turn peanut butter into diamonds.

Account Age: **21 days**
Connections: **1 followers / 11 following**
Posts: **1**

Figure 5.6: Example of a Post in Decentralized EUNOMIA.

As shown, the post has two voting options, which correspond to the simplest scenario: *trust* or *no trust*. Nonetheless, each ballot can be customized according to the needs of each post. Below the voting buttons, a tally is presented for each option.

Chapter 6

Evaluation

In this chapter, a complete and thorough evaluation of the system is performed. Initially, we set out to create a solution which complied with the desired properties, taking into account the design options of EUNOMIA. So, we start by reviewing the security of our proposal in Section 6.1, which includes trust assumptions, a threat model and proofs of security for the properties of the ring signature. Then, based on the security model built, we explain how the protocol complies with the intended requirements in Section 6.2.

Finally, in Section 6.3, the performance of the system is evaluated by providing time and bandwidth metrics, explaining some choices concerning the technologies used and making some considerations with regard to the efficiency, scalability and practicality.

6.1 Security

This section assesses the voting system, in terms of security. Trust assumptions are approached in Section 6.1.1. Then, in Section 6.1.2, we provide a threat model, for all possible adversaries, which also describes how the threats are mitigated. Lastly, the properties of the ring signature scheme are defined, and their proofs of correctness are sketched in Section 6.1.3.

6.1.1 Trust Assumptions

For the system to function according to the expectations, we need to provide some assumptions of trustworthiness. We refer to Section 4.2.2 for the security model of EUNOMIA, which is the backbone of the security model of the voting protocol. The trust assumptions are presented with respect to all entities and eventual consequences are discussed, should these assumptions be infringed.

Trust Assumption 1. The user trusts the digital companion.

The digital companion is a conceptualization of an interface between the users and the remainder of the ecosystem. This interface can run in any type of device, ranging from a smartphone to a personal computer, which can belong to the user. Because we address pure internet voting, voting can happen in any type of environment, including a malicious one. Therefore, the device of a voter can be compromised in many ways, including through the software stack, the network or even physically.

For this reason, we cannot ensure that the digital companion is not corrupted, and that it does not change the vote or reveal the identity of the user. Therefore, we assume it is not vulnerable to such attacks, and that it performs the operations correctly. This way, users can trust the digital companion. If this assumption does not hold, that is, if a user does not trust any device, voting is infeasible. Note, however, that users may be malicious and try to corrupt the system using the digital companion.

Trust Assumption 2. The voter registration process is carried out by an honest manager.

The registration process is conducted by a single manager, which verifies if the user has registered before as a voter. If this manager is malicious, it can register the same user multiple times as a voter and allow the registration of non-unique public keys. Furthermore, it can disclose the correlation between users and public keys, although this is irrelevant to our scheme, since the public keys are hidden behind a ring signature.

Trust Assumption 3. At least a threshold t of voting servers, which includes manager, tallier and anonymizer, is honest.

As referred in the security model of EUNOMIA, at least a subset of nodes must be honest and willing to cooperate in order for the system to function properly. Because the voting server is a service inside of each node, we rely on this assumption to conduct the voting processes. This threshold t (out of n voting servers) is configured when launching a new federation.

If there is not a threshold t of honest voting servers, users cannot vote. Moreover, note that, when a node refuses to work, the user can simply switch to an honest node.

Trust Assumption 4. There is, at least, one honest anonymizer in a mix network.

We implement a mix network between the user and the tallier, which the user can opt out of during the voting phase. This mixnet intends to simulate an anonymous channel, which provides privacy guarantees to the users. While only one honest mix is enough to ensure anonymity, a sequence of mixes would yield greater privacy assurance.

Regarding anonymization, we make some additional considerations. First, we assume that there exist, at all times, enough ballots to be shuffled, that is, multiple users vote approximately at the same time. If this condition is not fulfilled, a dishonest tallier could correlate ballots and users through timing. And second, the symmetric keys are always generated at random and used only once, otherwise privacy may be compromised.

Trust Assumption 5. The ECDLP is hard to solve, the RSA assumptions hold and the SHA-256 hash function implements a random oracle.

The RSA is a standard and well-studied assumption. The ECDLP is less studied, although there exists no efficient method to solve it. Should these assumptions be compromised, an attacker would be able to pretend to be any authority of the voting system and vote on behalf of other voters, respectively.

Trust Assumption 6. The communication channels between all entities provide confidentiality and integrity.

This assumption is inherited from the security model of EUNOMIA. While we do not study and approach these channels in this work, one can suggest the use of well-known secure networking protocols, such as TLS [95], to ensure confidentiality and integrity. If this assumption does not hold, the protocol cannot be sure of the authenticity of any data exchanged.

6.1.2 Threat Model

Threat models are useful to address potential threats to a system. In this section, we define some types of attackers, present some possible attacking scenarios and evaluate the security of the proposed protocol, with respect to each of the attackers and scenarios. Then, we either set forth the mitigation techniques performed, or choose to accept the risk.

Before starting the security evaluation of the system, we need to know what type of adversary the system is running against, so some potential attacker profiles are outlined.

Attacker 1. This is the most basic attacker. It can be anyone with a personal computer and very limited resources. We assume this entity is registered in EUNOMIA since it would allow easier interaction with the system, although this may not be the case. Through the digital companion, they can communicate with some services of the EUNOMIA node, including the voting service, but they cannot tap network channels nor access other personal computers.

Attacker 2. We generally define this attacker as someone with full access to the voting server of an EUNOMIA node. It has many capabilities, such as interacting directly with other voting servers and with all services within the same node. It can add, modify and delete data related to the voting protocol and can tap, but not interfere with, messages exchanged between other voting servers. This attacker, in particular, is an abstraction, and two concrete variants are defined below.

Attacker 2.1. It can be any node, except the one which deals with the user.

Attacker 2.2. It is the node which interacts directly with the user.

Attacker 3. This attacker is the strongest one. It has the power to control some of the nodes, including the node in contact with the user. Also, it can tap all network channels between every system entity, including user devices, as well as introduce counterfeit messages, and modify and suppress messages in transit.

One important note about all of these attackers is that they can only perform polynomial-time computations. Additionally, the collusion between attackers of different categories, particularly between attacker 1 and attackers 2 or 3, can occur, nevertheless it is not approached since it presents no threats besides the ones already addressed.

The next step is to specify some expected (normal) scenarios of the voting protocol, which mention the components involved and describe a simplified flow of the actions.

Expected Scenario 1. The user scrolls through the posts in the digital companion, chooses one post, generates a valid ballot with their choice, votes (with or without anonymization), the tallier gathers the required signatures and the ballot is stored.

Expected Scenario 2. The user deletes their vote through the digital companion and the manager removes the ballot from storage (or bulletin board).

Expected Scenario 3. The tallier gathers all ballots, for a certain post, from the bulletin board, verifies their signatures and delivers the correct results to the digital companion, to be presented to the users.

To identify the threats to our system, we use the well-known STRIDE [72] technique. The name STRIDE is a acronym for the threats to the most relevant security properties: *spoofing* (authenticity), *tampering* (integrity), *repudiation* (non-repudiation), *information disclosure* (confidentiality), *denial of service* (availability) and *elevation of privilege* (authorization). We evaluate the security of our protocol with regard to these six categories.

Some attacking scenarios are now defined, which are related to the normal executions of the protocol. For each scenario, the compromised STRIDE property is specified. We highlight that only voting (which includes validation) and tallying settings are considered, as we assume that registration (**TA 2**) is conducted properly.

Attacking Scenario 1. The user tries to vote more than once on the same post. (*Tampering*)

Attacking Scenario 2. The user tries to vote while pretending to be another user. (*Spoofing*)

Attacking Scenario 3. The attacker tries to vote using an unauthorized ballot. (*Tampering*)

Attacking Scenario 4. The attacker validates an invalid ballot and claims that it was not validated by them. (*Repudiation*)

Attacking Scenario 5. The user votes and the attacker tries to change the vote during voting. (*Tampering*)

Attacking Scenario 6. The user votes and the attacker tries to reveal the vote, along with the identity of the user, during or after voting. (*Information Disclosure*)

Attacking Scenario 7. The attacker tries to change or delete the vote of a user stored in the bulletin board. (*Tampering*)

Attacking Scenario 8. One or more attackers try to vote, request tallies or delete ballots multiple times simultaneously, in an attempt to drain the resources of a voting server. (*Denial of Service*)

Attacking Scenario 9. The attacker produces an incorrect tally deliberately and claims they are not responsible. (*Repudiation*)

We now proceed to the core task of the threat modeling process, which is the identification of threats. To carry out this process efficiently, the threats are grouped by attacker, from the weakest to the strongest. For each attacker, we indicate the normal operation of the protocol, the eventual attacks which can be conducted in that scenario, along with the compromised component. Finally, each discovered threat states the mitigation measures implemented.

We underline that, for each attacker, only applicable expected scenarios and viable attacking scenarios are approached, considering their capabilities. Moreover, whenever the mitigation is related to a trust assumption, we refer to that trust assumption as **TA**.

Attacker 1 (User)

Expected Scenario	Attacking Scenario	Component	Mitigation
1	1	Tallier	The ring signature and the pseudonym prevent double voting.
	2	Tallier	The pseudonym cannot be calculated by anyone, except by the owner of the private key (**TA 5**).
	3	Tallier	The ring signature grants the eligibility.
	5	Tallier	During voting, the tallier drops every incoming request to vote with the same pseudonym.

1	6	Tallier	The identity of the voter is hidden behind the pseudonym after voting (**TA 5**). This attacker cannot do anything during voting.
	8	Tallier	Each node implements security measures against denial of service.
2	6	Manager	The credentials of the voter are hidden behind the pseudonym (**TA 5**).
	8	Manager	Each node implements security measures against denial of service.
3	8	Tallier	Each node implements security measures against denial of service.

Table 6.1: Threat identification for Attacker 1.

Attacker 2.1 (Random Node)

Expected Scenario	Attacking Scenario	Component	Mitigation
1	3	Tallier	The ring signature can only be computed by a valid voter (**TA 5**) and a threshold t of talliers must sign it.
	4	Tallier	The ballot is signed when validated.
	5	Tallier	The ballot is signed using the ring signature.
	6	Anonymizer	The corruption of this anonymizer can be afforded as only one honest anonymizer is required to provide anonymization (**TA 4**).
2	Does not apply because this attacker does not deal directly with the user.		
3	4	Tallier	The ballot is signed when validated.
	6	Tallier	The credentials of the voter are hidden behind the pseudonym (**TA 5**).
	9	Tallier	Every tally is signed before being returned.

Table 6.2: Threat identification for Attacker 2.1.

Attacker 2.2 (User Node)

Expected Scenario	Attacking Scenario	Component	Mitigation
1	2	Tallier	A valid pseudonym can only be calculated by the owner of the private key. Though, the attacker can access the encrypted private key.
	3	Tallier	The ring signature can only be computed by a valid voter (**TA 5**) and a threshold t of talliers must sign it.
	4	Tallier	The ballot is signed when validated.
	5	Tallier	The ballot is signed using the ring signature.
	6	Anonymizer	As first anonymizer, it can keep the identity of the user and try to correlate after it leaves the circuit. In any case, only one honest anonymizer is needed to provide anonymization (**TA 4**).
2	6	Manager	After voting, the credentials of the voter are hidden behind the pseudonym (**TA 5**).
	7	Bulletin Board	The ring signature ensures the integrity of the ballot. However, there is no protection against deletion.
3			Same as Attacker 2.1.

Table 6.3: Threat identification for Attacker 2.2.

Attacker 3 (Set of Nodes)

Expected Scenario	Attacking Scenario	Component	Mitigation
1	2	Tallier	A valid pseudonym can only be calculated by the owner of the private key. Though, the attacker can access the encrypted private key.

	3	Tallier	If the attacker controls a threshold t of talliers, it can vote as many times as it wants, on any post (**TA 3**). However, the ballot will not have a valid ring signature, and eventual upcoming eligibility checks will not uphold.
	4	Tallier	The ballot is signed when validated.
1	5	Tallier, Network Channels	The ballot is signed using the ring signature (**TA 5**) and network channels provide security guarantees (**TA 6**).
	6	Anonymizer	If the attacker controls all anonymizers of the mix network, users might lose their anonymity (**TA 4**). After voting, the ring signature and the pseudonym hide the credentials of the voter, even if all nodes are compromised (**TA 5**).
	6	Manager	After voting, the credentials of the voter are hidden behind the pseudonym, even if all nodes are compromised (**TA 5**).
2	7	Bulletin Board	The ring signature ensures the integrity of the ballot, even if all nodes are corrupted (**TA 5**). However, there is no protection against deletion, which can go undetected with high probability if multiple nodes are malicious.
3	Same as Attacker 2.1 and 2.2 as controlling multiple nodes does not influence tallying.		

Table 6.4: Threat identification for Attacker 3.

6.1.3 Ring Signature

This section formalizes the ring signature scheme proposed in Section 4.4.2. The ring signature is a tuple of algorithms (**Generate**, **Sign**, **Verify**, **Link**), which are described below.

- **Generate** is a *probabilistic polynomial-time* (PPT) algorithm, which produces a key pair (y, x), where y is the public key and x is the private key.
- **Sign** is a PPT algorithm which, upon receiving a message m, a set of public keys $\mathcal{Y}$, a private key x and a post identifier p, outputs a signature σ on message m, which contains the pseudonym nym and the set $\mathcal{Y}$.
- **Verify** is a polynomial-time algorithm which, on input of a message m, a post identifier p and a signature σ (which already contains the set of public keys), returns the validity of the signature.

- **Link** is a polynomial-time algorithm, which receives two signatures σ_1 and σ_2, and returns whether they were issued by the same signer, for the same post.

For this scheme, three fundamental properties are proposed: *linkability*, *anonymity*[1] and *unforgeability*. In the following paragraphs, each property is clarified, and their validity and applicability to the ring signature is demonstrated.

Linkability. A ring signature is *linkable* if one can say that two distinct ring signatures were signed by the same private key, for the same post, independently of the set of public keys or the message. Relevantly, if the same private key generates two ring signatures for different posts, these will not be linkable.

Theorem 6.1.1 (Linkability)**.** *The proposed ring signature scheme provides linkability.*

Proof. Proving that our scheme achieves linkability is straightforward. The pseudonym generated by the **Sign** algorithm is deterministically computed using the post identifier p and the private key x of the signer. Therefore, the **Link** procedure simply has to compare the pseudonyms of two distinct ring signatures to know if they are linked. □

Anonymity. A ring signature is *anonymous* if it is not possible to determine which ring member actually issued the signature. In our construction, the public key can be inferred when the same signer issues a valid signature for the same post, and the set of public keys is completely different (but contains their public key). Furthermore, the signer loses anonymity if either their private key is revealed or every private key, except theirs, is disclosed.

Theorem 6.1.2 (Anonymity)**.** *The proposed ring signature construction is anonymous under the discrete logarithm assumption, in the random oracle model.*

Proof. The ring signature is built upon the proofs of partial knowledge by Cramer et al. [40]. Therefore, to demonstrate that it is infeasible for the verifier to know which private key was used to issue the signature, one just has to prove *witness indistinguishability*, that is, a cheating verifier, after engaging in multiple executions of the protocol, cannot tell which witness the prover is using. Remind that our scheme is a conversion from an interactive HVZK protocol to a single-move, signature scheme, using the Fiat-Shamir heuristic [54]. We refer to *Theorem 8* of Cramer et al. [40] to prove, based on the classic interactive HVZK sigma protocol, that the interactive proofs of partial knowledge are, indeed, witness indistinguishable. The authors also suggested that these proofs of partial knowledge could be transformed into ring signature schemes, such as ours. In fact, replacing the honest verifier with a random oracle [16] is comparable to forcing the verifier to be honest, since the random oracle generates random and independent challenges, similarly to an honest verifier [45]. Therefore, zero-knowledge is automatically achieved when using random oracles, and, as attested by Cramer et al. [40], zero-knowledge implies witness indistinguishability. Since the hash function used is assumed to work as a random oracle, our ring signature is witness indistinguishable. □

[1] Anonymity in ring signatures can also be called *signer-ambiguity*.

Unforgeability. This notion refers to the impossibility of an entity to issue a ring signature if they do not own a private key of the ring. In other words, a ring signature is *unforgeable* if no one besides the ring members can generate a valid ring signature.

Theorem 6.1.3 (Unforgeability)**.** *The proposed ring signature scheme is unforgeable under the discrete logarithm assumption, in the random oracle model.*

Proof. To prove the system is secure against existential forgery[2], the following contradiction is formulated: if an entity, which does not belong to the ring, can generate a valid ring signature with non-negligible probability of success, then the DLP (or ECDLP) can be solved in polynomial time with non-negligible probability. This statement has been proved by Pointcheval and Stern [93], in the random oracle model [16], for signatures schemes which result from the transformation of HVZK protocols. □

6.2 Compliance

In Section 1.2, the desired properties of the system were outlined. This section assesses the compliance with respect to those properties, taking into account the security evaluation presented in the previous section.

Accuracy. We split the accuracy property into two parts: *completeness* and *soundness*. Regarding completeness, if the third trust assumption holds, one can be sure that valid ballots are appropriately signed and added to the bulletin board. The system also satisfies soundness by making sure, through the ring signature, that only eligible voters are able to cast valid ballots. However, note that a threshold of corrupt talliers can post an invalid ballot (without a ring signature) to the bulletin board. Still, any honest party verifying the tally can trivially identify all invalid ballots.

Privacy. The identity of the user is hidden behind a mix network and a pseudonym, so privacy is fulfilled. However, if one were to break the fourth assumption, users could be associated with their ballots. Furthermore, and more serious, if the underlying cryptographic assumptions are broken, all ballots, including previously cast ones, can be trivially correlated to the public keys of the voters, by inverting the pseudonym. Still, if the registration process is executed honestly, the public keys are not linkable to the users.

Uniqueness. The pseudonym allows the system to ensure that a voter cannot vote twice on the same post, and the ring signature guarantees that the pseudonym is well formed. Unless the ring signature scheme is broken, which contradicts the fifth assumption, it is not possible to vote twice with the same key. Uniqueness could, however, be disrupted if a user owns multiple registered key pairs, which cannot happen because the registration process is performed correctly. Finally, duplicate ballots can still be

[2]Existential forgery refers to the possibility of an adversary to forge a signature on a single message, over which they have no control, so it may have no meaning or be random [61].

validated if a threshold of talliers are corrupted. Nevertheless, this can be easily detected later by any honest entity verifying a tally.

Verifiability. We divided verifiability into two components: *individual* and *universal*. Voter (or individual) verifiability is trivially achieved thanks to the possibility of any user to check their vote through the pseudonym. Regarding universal verifiability, one can request access to the bulletin board to verify if all recorded and valid ballots are present in the tally calculated by the voting server. Furthermore, everyone can confirm that only eligible voters cast a ballot, by verifying the ring signatures of each ballot.

Robustness. Ensuring availability when the system partially fails is inherited from EUNOMIA. In fact, most of the data is synchronized across the federation and, because the system is decentralized, some node failures can be tolerated. Additionally, in Section 6.1.2, it is shown that the system is resistant to malicious behavior performed by users and nodes, and even to the collusion of some nodes.

6.3 Performance

In this section, the performance of the proposed protocol is evaluated. Because the system deals directly with users, performance is a fundamental aspect, both for voting and tallying. To conduct a thorough evaluation, we assess the time taken by some operations, as well as the size of the data exchanged, while taking into consideration the environment in which each of the components is executed. These assessments were carried out until stable and reliable measurements were achieved. In addition, all results are discussed and some deliberations are made regarding tradeoffs and scalability.

Setup. As mentioned before, the solution encompasses two parts: the client library, which runs on user devices, and the voting server, which executes on remote machines. To simulate the client, a personal computer running MacOS 11, with four 2.6 GHz Intel Core i7 cores and 16 GB of RAM was used. This computer operates over a network averaging 100 Mbps of speed. The experiments for the voting server were deployed on a virtual machine running Ubuntu 18.04, with one Intel Core processor capable of 2.5 GHz of clock speed, 1 GB of RAM and networked on a 100 Mbps internet connection. Both environments attempt to recreate the expected conditions in which the protocol is executed.

Registration. This process involves the generation of credentials, which are then processed by a single voting server. After five tests, we concluded that the entire registration process takes around 900 ms, including network communication time, and that 0.6 KB of data are exchanged. On the voting server side, this operation lasts 725 ms on average, with a standard deviation of 60 ms. The generation of credentials in the device of the user, which comprises the computation of an Ed25519 key pair, generation of a symmetric key using PBKDF2 and encryption of the private key with such symmetric key, takes an average of 12 ms to complete, with a standard deviation of 2.5 ms.

Voting. To evaluate the performance of this stage, we started by assessing the ring signature, which is the core of the voting phase. Additionally, we measured the time to cast a vote, from the time the user decides to vote until a response is returned from the voting server. Table 6.5 presents these metrics, namely, the average time to generate and verify one ring signature with k keys, the size of that ring signature, and the mean time it takes to vote, from the perspective of the client, depending on the number of keys k and on the number of talliers t required to sign the ballot. Furthermore, the average voting time from the standpoint of the voting server (excluding communication with users) is also included in the metrics, in parenthesis. Finally, for each k and t, five experiments were conducted, and these were performed using trust/no-trust[3] ballots.

	Ring Signature			Voting		
	Generation	Verification	Size	$t = 1$	$t = 2$	$t = 3$
$k = 5$	61 ms	23 ms	2.3 KB	694 (409) ms	2151 (1847) ms	2870 (2567) ms
$k = 10$	107 ms	23 ms	4.5 KB	786 (386) ms	2258 (1916) ms	2927 (2565) ms
$k = 30$	291 ms	48 ms	13.1 KB	905 (371) ms	2500 (1932) ms	3163 (2599) ms
$k = 100$	924 ms	110 ms	43.1 KB	1634 (468) ms	3183 (1974) ms	3927 (2759) ms

Table 6.5: Performance of the voting phase for a trust/no-trust ballot.

Regarding the ring signature, a relevant note is pointed out. Generally, elliptic curve libraries pre-compute or cache some values in order to speed up future operations, which is the case for our libraries. When k changed, the times were registered only after two or three trials, until stable results were achieved. Nevertheless, one could expect them to decrease slightly more, both for generation and verification, due to the caching capability.

Concerning the voting times, there are a few factors which can influence these measurements. First, when $t > 1$, the receiving tallier is responsible for broadcasting the ballot and gathering signatures. This process is easily affected by the node communication system. In fact, communication overheads can be irregular, and the estimated overhead per message is between 150 and 300 ms.

Second, the voting server is a virtually isolated component, as mentioned in Section 5.2. As a matter of fact, it communicates with other services in the same machine using a virtual network. The development of decoupled services, despite being an advantage when integrating new services, produces a great overhead when services need to interact with each other. This is applicable to the voting component because multiple requests are made to the authentication, discovery and storage services.

Finally, for $t = 2$ and $t = 3$, one would expect the voting times to be similar, since broadcasting to talliers and subsequent operations should be concurrent. Once again, this is due to unusual behavior of the node communication service. For all these reasons, we establish the reference times to validate a ballot as the ones when $t = 1$, from the perspective of the voting server.

By default, the system uses three talliers ($t = 3$) and five public keys ($k = 5$) when voting. Natu-

[3]Remember that a trust/no-trust ballot is the simplest and lightest one, where the user chooses whether to trust or not to trust the information of a post. This ballot takes around 0.1 KB of space, without features, ring signature or tallier signatures.

rally, when this threshold t of talliers increases, the system becomes more robust and tamper-resistant, nonetheless time performance is expected to decline due to the aforementioned overheads.

Regarding the number of keys k, there exists another privacy and performance trade-off, that is, when k increases, privacy improves but performance worsens. Performance was prioritized by setting $k = 5$ by omission, which implies that an adversary would need to corrupt four users to be able to infer which key was indeed used to issue the ring signature. Still, in the future, the system may allow users to choose the number of keys used to build their ring signature.

Tallying. Along with voting, this is one of the most important metrics of the protocol. Because the system is continuously issuing tallies, the evaluation and optimization of this process is essential for this work. We recall that each ballot is validated upon a tallying request, which encompasses the verification of all signatures of all ballots. By default, one ballot is signed by three talliers. Table 6.6 lists the tally issuance times for n trust/no-trust ballots, with a ring signature of size $k = 5$, each of them signed by t talliers. These timing statistics only consider the perspective of the voting server.

	$t = 2$	$t = 3$	$t = 4$
$n = 10$	61	63	59
$n = 100$	139	136	124
$n = 500$	372	411	414

Table 6.6: Average tallying times (in milliseconds).

Note that, after running benchmark tests using the OpenSSL[4] tool, the voting server verifies an RSA 2048-bit signature in 0.032 ms on average. That being said, it can be seen that there exists no substantial difference when using between two and four talliers. The time difference could be noticeable though, if the number of ballots n was to be significantly greater than 500. For example, based on the benchmark metrics, if $n = 100000$, the extra signatures would take an additional 3.2 seconds to be verified.

Some relevant considerations are still pointed out. First, these calculations do not include the time to retrieve tallier certificates from the discovery service, which takes, on average, 100 ms. We have decided to exclude it since it is only done once, and then cached locally. Second, as mentioned before, these time measurements do not cover communication time with the client. Anyway, based on our measurements and under the specified conditions, we estimate that the additional networking time would be between 100 and 200 ms, for $10 < n < 500$.

Furthermore, one might find that the results for $n = 10$ and $n = 100$ are not anticipated, since the expectation would be for the average times to increase alongside the number of signatures to be verified. In fact, as mentioned before, the voting server relies on another service for storage which, despite being on the same server, may not deliver linear timing results. Therefore, this particular outcome may be due to a few factors, such as irregularities in the communication or data caching policies.

[4]`https://www.openssl.org` (accessed December 31, 2020)

Finally, there is one valuable optimization, which can reduce tallying times. The ballots could be cached and only new ballots would be retrieved whenever a new tally is requested. This ballot caching feature was not implemented yet, although we leave it as future work.

Chapter 7

Conclusions

In the past decades, the number of people accessing information through social media and the internet has been increasing. In fact, nowadays, it is significantly easier to reach people using computer-mediated communication, and to change their beliefs or attitudes.

Disinformation has been gaining momentum in the last few years, either in the form of deceptive advertising, distorted news, or even government propaganda. This is a very controversial issue, since online persuasion has become easier and more common throughout the past decades.

Following the growth of these phenomena, the EUNOMIA Project comes along to help fight the spread of disinformation. This platform comprises multiple techniques to assist people in assessing the reliability of information on social media. Among them, there is a voting system which allows users to vote on trustworthiness characteristics of social networking posts.

Nevertheless, designing such a voting system is a challenging task. As a matter of fact, it should support multiple simultaneous, open-ended voting processes with flexible ballots, and must be able to grant eligibility and avoid double voting without compromising privacy. Furthermore, it has to respect the properties set forth in Section 1.2, such as accuracy and verifiability. To date, and to the best of our knowledge, there are no systems which feature these characteristics.

In this work, a proposal for the above-mentioned internet voting system was elaborated. Naturally, this system is integrated in EUNOMIA, and designed according to its principles, yielding robustness. We use ring signatures and pseudonyms to grant eligibility, and implement a mix network to ensure anonymity. Additionally, the system is fully transparent and provides users with the capability of verifying their votes, and that the tallying process is performed correctly.

The details of the implementation were advanced in Chapter 5, and a thorough security and performance evaluation was conducted, which results are presented in Chapter 6. The evaluation showed the system complies with the initially defined properties, and that it puts forward the adequate mechanisms to fight against a range of relevant attacks, carried out by both weak and powerful attackers.

As referred in Section 5.3.1, the entire EUNOMIA ecosystem, including, and relevantly, the voting component, underwent an initial test with a few hundred users. The next testing phase is planned for

January 2021, and will be performed in association with the social journalism platform Blasting News[1], which counts with more than 100 million monthly readers.

7.1 Achievements

Throughout the course of this dissertation, multiple achievements were accomplished, which complied with the initially defined objectives. These achievements are listed below.

- Comprehensive survey on electronic voting systems, which includes their strengths and limitations.
- Proposal for an efficient internet voting protocol, based on ring signatures, which features accuracy, privacy and verifiability, while supporting multiple simultaneous, open-ended voting processes and flexible ballots.
- Full prototype of this voting system, implemented in Java and Javascript.
- Proof of concept for this system, in the context of fighting disinformation, with three hundred users.

7.2 Future Work

The proposed system can be further optimized. We leave some future work suggestions, which would make this solution more efficient and secure.

First, the proposed ring signature requires $O(n)$ (linear) time to be generated and verified. This scheme could be improved by using a technique which allows the generation and verification of the ring signature in $O(1)$ (constant) time. One possibility could be the use of accumulators [19] in zero-knowledge set-membership proofs [31]. Nevertheless, this improvement is significantly more complex than the one proposed, and has unwieldy implementation details.

Another efficiency optimization, already mentioned in Section 6.3, is to cache ballots when tallying, which would significantly reduce tallying times and make the system more efficient.

Shifting to security enhancements, we suggest the introduction of dummy ballots in the circuit of anonymizers to deceive potential adversaries. This refinement removes the need to assume that there would be enough ballots to be shuffled. Additionally, another optimization to the mix network is to add padding to all encrypted ballots, such that the ciphertexts would always have the same size.

The ballot deletion process can also be improved, in order to avoid unauthorized deletion of ballots by randomly generating pseudonyms. The solution would be to use a zero-knowledge proof to prove the correct computation of the pseudonym.

Finally, we leave feature validation to be implemented in the future.

[1] `https://www.blastingnews.com` (accessed December 31, 2020)

Bibliography

[1] EUNOMIA Deliverable 3.2 (Specifications and architecture design). Technical report, University of West Attica (UWA). `https://eunomia.social/EUNOMIA-D3.2-Architecture-SUBMITTED.pdf/` (accessed December 31, 2020).

[2] EUNOMIA Project. `https://eunomia.social/`. (accessed December 31, 2020).

[3] M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 437–447. Springer, 1998.

[4] M. Abe and E. Fujisaki. How to date blind signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 244–251. Springer, 1996.

[5] M. Abe and T. Okamoto. Provably secure partially blind signatures. In *Annual International Cryptology Conference*, pages 271–286. Springer, 2000.

[6] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–432. Springer, 2002.

[7] B. Adida. Advances in cryptographic voting systems. *Caltech/MIT Voting Technology Project*, 2006.

[8] B. Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[9] B. Adida, O. De Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. *EVT/WOTE*, 9(10), 2009.

[10] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[11] K. Arnold, J. Gosling, D. Holmes, and D. Holmes. *The Java programming language*, volume 2. Addison-Wesley Reading, 2000.

[12] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Annual International Cryptology Conference*, pages 255–270. Springer, 2000.

[13] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, et al. *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration, 2006.

[14] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283, 2001.

[15] S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 263–280. Springer, 2012.

[16] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, 1993.

[17] J. Benaloh. Simple verifiable elections. *EVT*, 6:5–5, 2006.

[18] J. Benaloh. Ballot casting assurance via voter-initiated poll station auditing. *EVT*, 7:14–14, 2007.

[19] J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 274–285. Springer, 1993.

[20] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553, 1994.

[21] J. C. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 52–62, 1986.

[22] J. D. C. Benaloh. Verifiable secret-ballot elections. 1989.

[23] D. J. Bernstein. Curve25519: new diffie-hellman speed records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.

[24] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

[25] D. Boneh. The decision diffie-hellman problem. In *International Algorithmic Number Theory Symposium*, pages 48–63. Springer, 1998.

[26] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Annual International Cryptology Conference*, pages 41–55. Springer, 2004.

[27] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

[28] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Annual International Cryptology Conference*, pages 465–480. Springer, 2002.

[29] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145, 2004.

[30] J. Camenisch. Efficient and generalized group signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–479. Springer, 1997.

[31] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Annual International Cryptology Conference*, pages 61–76. Springer, 2002.

[32] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Annual International Cryptology Conference*, pages 410–424. Springer, 1997.

[33] D. Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[34] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Annual International Cryptology Conference*, pages 89–105. Springer, 1992.

[35] D. Chaum and H. van Antwerpen. Undeniable signatures. In *Conference on the Theory and Application of Cryptology*, pages 212–216. Springer, 1989.

[36] D. Chaum and E. van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[37] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[38] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368. IEEE, 2008.

[39] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. 1985.

[40] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[41] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 72–83. Springer, 1996.

[42] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[43] L. F. Cranor and R. K. Cytron. Sensus: A security-conscious electronic polling system for the internet. In *Proceedings of the Thirtieth Hawaii International Conference on system sciences*, volume 3, pages 561–570. IEEE, 1997.

[44] J. Daemen and V. Rijmen. AES proposal: Rijndael. 1999.

[45] I. Damgård. On Σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002.

[46] Y. Desmedt. Society and group oriented cryptography: A new concept. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 120–127. Springer, 1987.

[47] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Conference on the Theory and Application of Cryptology*, pages 307–315. Springer, 1989.

[48] Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 557–572. Springer, 2000.

[49] W. Diffie and M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[50] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[51] B. W. DuRette. Multiple administrators for electronic voting. *Bachelor thesis, Massachusetts Institute of Technology, Boston, USA*, 1999.

[52] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[53] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, 1990.

[54] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[55] D. Flanagan. *JavaScript: the definitive guide*. O'Reilly Media, Inc., 2006.

[56] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.

[57] E. Fujisaki. Sub-linear size traceable ring signatures without random oracles. In *Cryptographers' Track at the RSA Conference*, pages 393–415. Springer, 2011.

[58] E. Fujisaki and K. Suzuki. Traceable ring signature. In *International Workshop on Public Key Cryptography*, pages 181–200. Springer, 2007.

[59] C. F. Gauss. *Disquisitiones arithmeticae*, volume 157. Yale University Press, 1966.

[60] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[61] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988.

[62] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[63] M. A. Herschberg. *Secure electronic voting over the world wide web*. PhD thesis, Massachusetts Institute of Technology, 1997.

[64] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 539–556. Springer, 2000.

[65] M. Jakobsson. A practical mix. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 448–461. Springer, 1998.

[66] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 143–154. Springer, 1996.

[67] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX security symposium*, pages 339–353. San Francisco, USA, 2002.

[68] R. Joaquim, A. Zúquete, and P. Ferreira. REVS – a robust electronic voting system. *IADIS International Journal of WWW/Internet*, 1(2):47–63, 2003.

[69] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections*, pages 37–63. Springer, 2010.

[70] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[71] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[72] L. Kohnfelder and P. Garg. *The threats to our products*. Microsoft Interface, Apr. 1999.

[73] D. W. Kravitz. Digital signature algorithm, July 27 1993. US Patent 5,231,668.

[74] J. K. Liu and D. S. Wong. Linkable ring signatures: Security models and new schemes. In *International Conference on Computational Science and Its Applications*, pages 614–623. Springer, 2005.

[75] J. K. Liu, V. K. Wei, and D. S. Wong. A separable threshold ring signature scheme. In *International Conference on Information Security and Cryptology*, pages 12–26. Springer, 2003.

[76] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, 2004.

[77] Ü. Madise and T. Martens. E-voting in Estonia 2005. The first practice of country-wide binding Internet voting in the world. In *Electronic Voting 2006 – 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting. CC*. Gesellschaft für Informatik eV, 2006.

[78] M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 125–132. Springer, 1996.

[79] V. S. Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

[80] K. Moriarty, B. Kaliski, and A. Rusch. PKCS# 5: Password-based cryptography specification version 2.1. RFC 8018, Jan. 2017.

[81] M. Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.

[82] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001.

[83] V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In *International Conference on the Theory and Application of Cryptology*, pages 164–170. Springer, 1994.

[84] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto. An improvement on a practical secret voting scheme. In *International Workshop on Information Security*, pages 225–234. Springer, 1999.

[85] T. Okamoto. An electronic voting scheme. In *Advanced IT Tools*, pages 21–30. Springer, 1996.

[86] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *International Workshop on Security Protocols*, pages 25–35. Springer, 1997.

[87] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[88] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 248–259. Springer, 1993.

[89] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[90] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer, 1991.

[91] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč. Foundations of JSON schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273. International World Wide Web Conferences Steering Committee, 2016.

[92] B. Pfitzmann. Breaking an efficient anonymous channel. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 332–340. Springer, 1994.

[93] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer, 1996.

[94] M. O. Rabin. Digitalized signatures. *Foundations of secure computation*, pages 155–168, 1978.

[95] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, Aug. 2018.

[96] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[97] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

[98] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *Annual International Cryptology Conference*, pages 411–424. Springer, 1994.

[99] K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 393–403. Springer, 1995.

[100] D. Sandler, K. Derr, and D. S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium*, volume 4, page 87, 2008.

[101] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer, 1989.

[102] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[103] N. P. Smart. *Cryptography made simple*. Springer, 2016.

[104] P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu, and D. S. Wong. Separable linkable threshold ring signatures. In *International Conference on Cryptology in India*, pages 384–398. Springer, 2004.

Appendix A

Diagrams

A.1 Voting Components

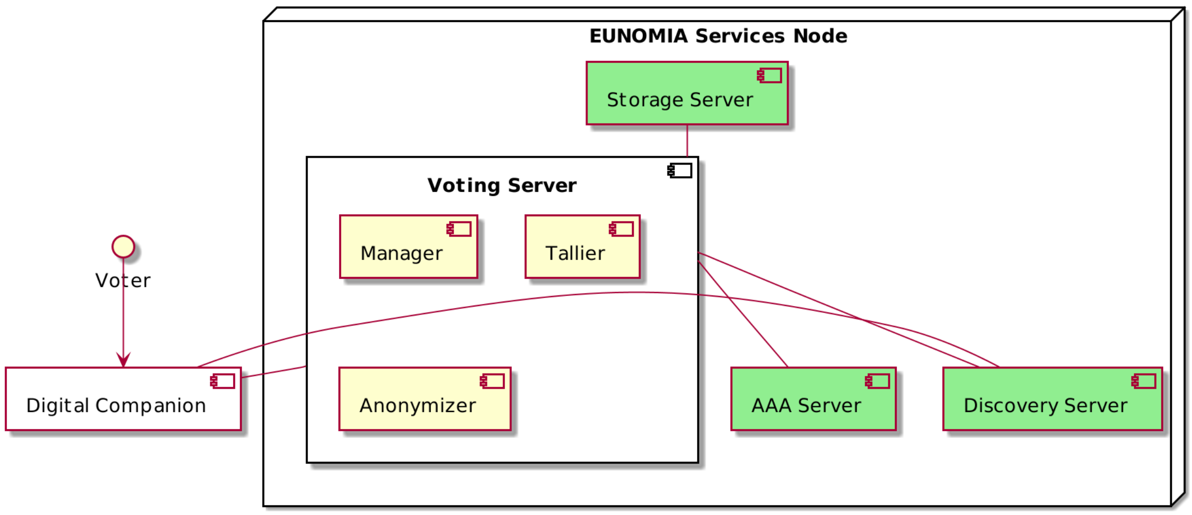
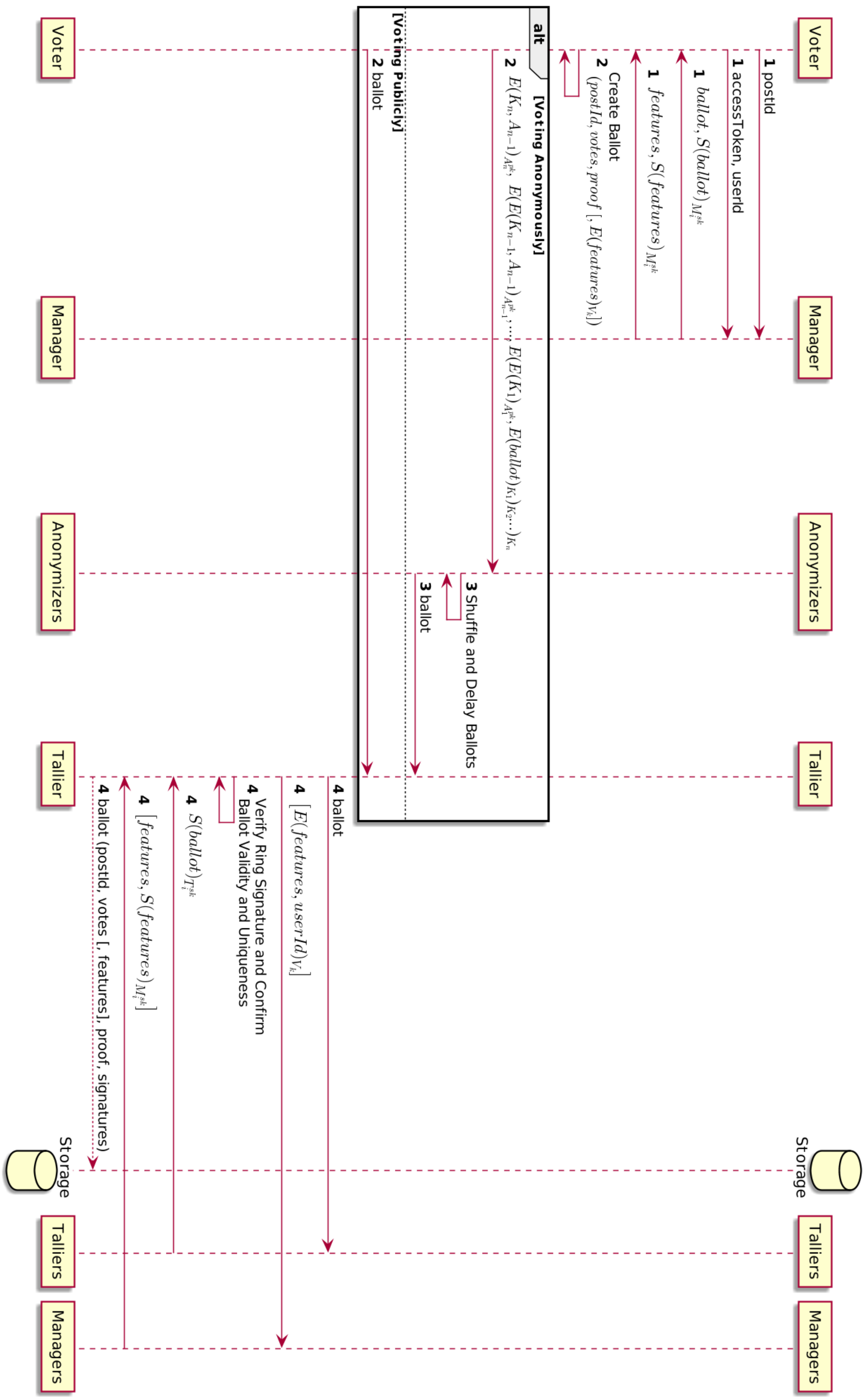


Figure A.1: Components of the solution.

A.2 Voting Protocol



Voter
Manager
Anonymizers
Tallier
Storage
Talliers
Managers
1 postId
1 accessToken, userId
1 $ballot, S(ballot)_{M_i^{sk}}$
1 $features, S(features)_{M_i^{sk}}$
2 Create Ballot
$(postId, votes, proof\,[, E(features)_{V_i}])$
alt
[Voting Anonymously]
2 $E(K_n, A_{n-1})_{A_n^{pk}}, E(E(K_{n-1}, A_{n-1})_{A_{n-1}^{pk}}, \dots, E(E(K_1)_{A_1^{pk}}, E(ballot)_{K_1})_{K_2} \dots)_{K_n}$
3 Shuffle and Delay Ballots
3 ballot
[Voting Publicly]
2 ballot
4 ballot
4 $[E(features, userId)_{V_i}]$
4 Verify Ring Signature and Confirm
Ballot Validity and Uniqueness
4 $S(ballot)_{T_i^{sk}}$
4 $[features, S(features)_{M_i^{sk}}]$
4 ballot (postId, votes [, features], proof, signatures)

Figure A.2: Sequence diagram of the proposed protocol.