Addressing Disinformation With Open-Ended Internet Voting Using Ring Signatures

Pedro Forjaz Figueiredo
pedronfigueiredo@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2021

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

1. Introduction

Since the 1980s, 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.

Electronic voting can be challenging. Some trust assumptions, such as untappable channels, 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.

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.

Disinformation 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[1] Project [1] 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.

[1]Supported by the European Union H2020 Research and Innovation Programme, with Grant Agreement number 825171.

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

EUNOMIA is a fully decentralized and distributed peer-to-peer platform, delegating the overall system management and tasks to a set of nodes. These nodes are operated by independent and unrelated entities, therefore, 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.

1.1. Related Work

Generally, voting protocols are categorized by the cryptographic procedure used to anonymize votes. The most frequently employed cryptographic approaches are mix networks, homomorphic encryption and blind signatures. While these three are the most common, there exist other techniques, namely, *threshold cryptography* [21], which finds its roots in the secret sharing scheme by Shamir [41], and zero-knowledge proofs [26]. 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.

The concept of mix network was first presented by Chaum [14] in 1981. In this technique, 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.

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.

After the primordial protocol by Chaum [14], many voting schemes followed [2, 6, 29, 32, 36, 39], which solved many security issues and achieved efficient results. Mix network voting systems [3, 15] usually implement the proposed protocols, with some variations. Furthermore, a significant number of proposals [5, 30, 33, 39] aimed at developing permutation proofs also emerged, allowing mixnet-based voting protocols to be more efficient.

Voting schemes based on homomorphic encryption work by encrypting all votes separately with a public key that 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 the ballot validity, should be fulfilled through proofs of correctness.

Cohen (now Benaloh) and Fischer [16] proposed the first homomorphic encryption voting protocol in 1985, using a centralized authority. Then, some proposals emerged [7, 8], which distributed the powers of the authorities among multiple machines. The subsequent approaches focused on efficiency [18, 19, 38] and on receipt-freeness [4, 28].

Blind signatures were introduced by Chaum [12] in 1983. 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.

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. Note that voters must send their signed ballot through anonymous channels, otherwise privacy may be compromised.

Fujioka et al. [25] are widely recognized as the pioneers of blind-signature-based voting schemes. The propositions which followed prioritized receipt-freeness [34, 35]. Then, multiple implementations of this type of voting system were put forward [20, 23, 27, 31], in which the tendency is to improve the robustness of the system, by distributing the powers of the authorities between a number of machines.

Regarding the connection between the current work and our work, a few remarks are pointed out. First, all the work presented so far has a strict separation between the voting and the tallying phases, which concerns the fairness property. 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. Then, before counting the ballots, the iden-

tities are usually discarded to avoid compromising privacy. 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 [23, 27, 31]. 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.

1.2. Contributions

This work focuses solely on the third part of the EUNOMIA Project [1], 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 final tally, if they wish to have their identity shown, or if they would like to stay anonymous.

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 server, supported by the previously mentioned peer-to-peer storage network.

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 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. Outline of the Paper

Section 2 provides the cryptographic building blocks for the solution. In Section 3, the proposed voting protocol is described. Section 4 then explains succinctly how this solution was implemented. Finally, Section 5 reports the security and performance evaluation of the protocol and Section 6 contains the concluding remarks.

2. Building Blocks

2.1. Cryptographic Assumptions

Definition 2.1. A function $f : \mathbb{N} \to \mathbb{R}_0^+$ 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. 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*.

The discrete logarithm assumption holds for a group $\mathbb{G}$ if no *probabilistic polynomial-time algorithms* can solve the DL problem in $\mathbb{G}$ with non-negligible probability.

2.2. Proofs of Knowledge

Zero-knowledge proofs of knowledge are protocols which allow one party to prove to another that they know a value, without revealing it. We can achieve this using an interactive protocol called sigma (Σ). The Schnorr protocol [40] is a particular example of the sigma protocol, which allows one to prove the knowledge of a discrete logarithm in a cyclic group.

To define proofs of knowledge formally, we resort to the symbolic notation proposed by Camenisch and Stadler [11]. We then generally represent proofs of knowledge as follows:

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

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

For example, for prover $\mathcal{P}$ to prove knowledge of x to verifier $\mathcal{V}$, such that $PK\{(x) : h = g^x\}$, where $\mathbb{G} = \langle g \rangle$ of order p and $h \in \mathbb{G}$ are known to all parties, and $x \in \mathbb{Z}_p$ is secret, one follows the protocol by Schnorr:

1. $\mathcal{P}$ picks k randomly from $\mathbb{Z}_p$, calculates $r := g^k$ and sends to $\mathcal{V}$;
2. $\mathcal{V}$ chooses a random challenge $c \in \mathbb{Z}_p$ and returns it to $\mathcal{P}$;
3. $\mathcal{P}$ sends $s := k + cx \ (mod\ p)$ to $\mathcal{V}$, who can verify that $g^s = h^c \cdot r$.

The great advantage of these proofs is that one can join together multiple statements about the secrets which knowledge is being proved. In particular, proofs of disjunction are pertinent to this work. Cramer et al. [17] proposed a witness indistinguishable protocol for these proofs of partial knowledge. To prove knowledge of x or y, such that $PK\{(x, y) : a = g^x \vee b = h^y\}$, one follows the protocol below. Note that $\mathcal{P}$ only knows x.

1. $\mathcal{P}$ picks k_x, c_y, s_y randomly from $\mathbb{Z}_p$, calculates
$$\begin{cases} r_a := g^{k_x} \\ r_b := h^{s_y} \cdot b^{-c_y} \end{cases}$$
and r_a and r_b sends to $\mathcal{V}$;
2. $\mathcal{V}$ chooses a random challenge $c \in \mathbb{Z}_p$ and returns it to $\mathcal{P}$;
3. $\mathcal{P}$ computes the challenge and the response for x as $c_x := c - c_y \ (mod\ p)$ and $s_x := k_x + c_x x \ (mod\ p)$, respectively, and sends c_x, c_y, s_x and s_y to $\mathcal{V}$, who can verify that (1)
$$\begin{cases} g^{s_x} = a^{c_x} \cdot r_a \\ h^{s_y} = b^{c_y} \cdot r_b \end{cases}$$
and (2) $c = c_x + c_y \ (mod\ p)$.

Fiat and Shamir [24] proposed a solution to transform the three-move 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, using a cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^*$, which eliminates the need of requesting a random challenge from the verifier. For example, for the Schnorr protocol above, the challenge would be $c := H(h \| g \| r)$.

Relevantly, this non-interactive protocol can be used to create a signature scheme, which is called a *signature proof of knowledge* (SPK). We represent such signature as

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

This example basically constitutes a Schnorr signature [40], where message m is signed using the private key x. This works exactly like a non-interactive protocol, except that message m is added to the challenge, such that $c := H(h \| g \| r \| m)$.

3. Voting Protocol

In this chapter, we propose a voting protocol to be integrated in the EUNOMIA platform.

3.1. 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.

The approach we propose 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.
- 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.

3.2. EUNOMIA

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*. Each node comprises a set of services, some of which carry relevance to the voting protocol. The most pertinent to this work are listed below.

- **AAA Server**: Responsible for authentication and authorization of users, this service provides a bridge between the EUNOMIA authentication and the voter registration process.

- **Discovery Server**: 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 data. It keeps all information regarding the voting processes, including ballots and voter credentials.

Each ESN runs on top of another node, which belongs to the social networking platform. 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.

3.3. Entities

The voting protocol comprises the following parties:

- **Voter**: Evaluates online posts by attributing characteristics of trustworthiness. In the EUNOMIA ecosystem, the voter is the social network user and carries out the actions through a device, which is called the digital companion (DC). They have a key pair, and we refer to the public and private keys of a voter as y and x, respectively.
- **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, each of these entities has a key pair, which is generated when the voting server is initialized for the first time.

3.4. Architecture

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

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 3.4.1. 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 device 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.

3.4.1 Registration

The registration process is carried out only once, preferably when the user is signing up with EUNOMIA. Despite only being part of the voting component, the registration relies heavily on the 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.

3.4.2 Voting and Validation

We use the notion of *pseudonym*, which allows users to hide their identity from the system, while providing linkability.

To formalize the pseudonym, we denote $\mathbb{G}$ as a prime order group and define $H_1 : \{0,1\}^* \to \mathbb{G}$ as a cryptographic hash function. It 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, the pseudonym is hard to reverse, so the identity of the user is preserved.

To ensure the pseudonym is correctly computed, it is calculated within the ring signature. The proposed ring signature is then 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. 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.

This construction uses the Chaum-Pedersen protocol [13] 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. [17], one can prove that, at least, one of the conjunctive statements is valid, revealing nothing else.

For a matter of simplicity, we assign $h := H_1(postId)$. We define $H : \{0,1\}^* \to \{0,1\}^*$ 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.

The protocol works as follows:

1. The voter 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:

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

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

3. Then, resorting to the Fiat-Shamir Heuristic [24], the voter generates a challenge c based on the common knowledge. The challenge is defined as

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

   Then, the voter computes a valid challenge for the legitimate statement as

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

4. The protocol proceeds to the last phase. 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:

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

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

Regarding feature validation, there exists a (t, n) threshold scheme [41], in which the public key belongs to a federation of n nodes, and the private key is distributed among the nodes. Whenever a new node joins the federation, it is responsible for generating a new key pair in a distributed manner. Note that features sent for validation, inside the ballot, are encrypted for privacy reasons.

When validating features, the manager inside the receiving node computes a partial decryption of the features ciphertext, and sends it to a threshold of other managers. Then, these open the features using their share of the private key, validate them, sign them, and send them back to the manager which requested their validation in the first place. Finally, the received signatures are validated and the features, decrypted, and unlinked to the user, are appended to the ballot.

3.4.3 Anonymization

In this setting, anonymity becomes remarkably 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 anonymization technique, such as Tor [22]. 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 [14]. 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.

3.4.4 Verifying and Deleting Votes

Through the pseudonym, which no one besides the owner of the private key can compute, we allow users to retrieve their previously cast ballot.

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. Still, because we assume there is low coercion in our scenario, we do not develop on this topic further.

3.4.5 Tallying Votes

The tallying process is simple and requires small computational effort. 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 that signed the ballots returned.
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.
4. The tally is signed and returned.

Note that 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. Implementation

4.1. Credentials

Each entity of the protocol has a credential. The parties and the cryptosystems used for their credentials are stated below.

- **Manager, Tallier and Anonymizer**: During the first initialization process, all of them generate distinct 2048-bit RSA [37] key pairs. Then, they issue self-signed certificates, which are used for testing purposes, and are then 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 [9, 10]. The public key is a point

in the curve, which, if compressed 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 [10] and x is the randomly generated private key.

4.2. Voting Server and Client

The voting server is a service which runs inside the EUNOMIA services nodes. Each service provides an Application Programming Interface (API), which works as an abstraction layer to the other services, 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 application, and runs an embedded web service. This web service is powered by the Grizzly[3] and Jersey[4] frameworks and it provides the functionality for the aforementioned API.

Relevantly, EUNOMIA nodes are built on top of social network nodes. To run initial tests, we have used Mastodon[5], which is a distributed social networking protocol. Also, given the necessity for modularity of the components, all services run in the same machine, but are virtually isolated from each other using Docker[6].

Regarding the client, it provides the functionality required from a user, with regard to the voting protocol. Remind that the digital companion (DC) 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, which can be easily executed by any browser, using a Javascript engine. When the code is executed outside the browser, Node.js[7] is used as runtime environment. In fact, we set up the client library to work primarily with Node.js, which allowed portability to other settings, such as mobile applications.

4.3. Decentralized EUNOMIA

Decentralized EUNOMIA[8] 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 Mastodon social networking protocol.

[3] `https://javaee.github.io/grizzly`
[4] `https://eclipse-ee4j.github.io/jersey`
[5] `https://joinmastodon.org`
[6] `https://www.docker.com`
[7] `https://nodejs.org`
[8] `https://decentralized.eunomia.social`

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.

5. Evaluation

5.1. Security

For the system to function according to the expectations, we need to provide some trust assumptions. These are listed below.

- The user trusts the digital companion.
- The voter registration process is carried out by an honest manager.
- At least a threshold of voting servers, which includes manager, tallier and anonymizer, is honest.
- There is, at least, one honest anonymizer in a mix network.
- The ECDLP is hard to solve, the RSA assumptions hold and the SHA-256 hash function implements a random oracle.
- The communication channels between all entities provide confidentiality and integrity.

In subsection 1.2, the desired properties of the system were outlined. We assess the compliance of the protocol with respect to those properties.

Accuracy. 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 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.

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. Notwithstanding, duplicate ballots can still be validated if a threshold of talliers is corrupted. Nevertheless, this can be easily detected 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 and malicious behavior can be tolerated.

5.2. Performance

Setup. 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.

Registration. 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 about 725 ms. Furthermore, the generation of credentials in the device of the user takes an average of 12 ms to complete.

Voting. We started by assessing the performance of the ring signature. The results are shown in Table 1. 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, depending on the number of keys k and on the number of talliers t required to sign the ballot. These measurements are presented in Table 2. For each k and t, five experiments were conducted, and these were performed using trust/no-trust ballots.

Regarding the number of keys k, there exists a 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.

	Generation	Verification	Size
$k = 5$	61 ms	23 ms	2.3 KB
$k = 10$	107 ms	23 ms	4.5 KB
$k = 30$	291 ms	48 ms	13.1 KB
$k = 100$	924 ms	110 ms	43.1 KB

Table 1: Performance of the ring signature.

	$t = 1$	$t = 2$	$t = 3$
$k = 5$	694	2151	2870
$k = 10$	786	2258	2927
$k = 30$	905	2500	3163
$k = 100$	1634	3183	3927

Table 2: Average time to vote (in milliseconds).

Tallying. 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. Table 3 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 3: Average tallying times (in milliseconds).

6. Conclusions

In this work, a proposal for an internet voting system was elaborated. This system is integrated in EUNOMIA, and designed according to its principles. 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.

Furthermore, a fully functioning prototype was implemented, which was tested with three hundred users. The next testing phase is planned for Jan-

uary 2021, and will be performed in association with the social journalism platform Blasting News[9], which counts with more than 100 million monthly readers.

References

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

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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.

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

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

[10] 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.

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

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

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

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

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

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

[28] 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.

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

[30] 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.

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

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

[33] 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.

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

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

[36] 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.

[37] 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.

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

[39] 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.

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

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