A two authorities electronic vote scheme

Antonio M. Larriba[1], José M. Sempere[2], and Damián López[2]

[1] Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
anlarflo@dsic.upv.es
[2] VRAIN - Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
{jsempere,dlopez}@dsic.upv.es

June 24, 2020

Abstract

In this paper we propose a new electronic multi-authority voting system based on blind signatures. We focus on the open problem of the efficiency of electronic voting systems. Most of the proposed systems rely on complex architectures or expensive proofs, in this work we aim to reduce the time-complexity of the voting process, both for the voter and the authorities involved. Our system is focused on simplicity and it is based on the assumption of two unrelated entities. This simplicity makes our approach scalable and flexible to multiple kinds of elections. We propose a method that limits the number of authorities to only 2 of them; we reduce the overall number of modular operations; and, propose a method which cut downs the interactions needed to cast a vote. The result is a voting protocol whose complexity scales linearly with the number of votes.

Keywords: Electronic vote; Homomorphic cryptography; Multi-authority system; Blind signatures; RSA.

1 Introduction

The design of electronic voting schemes is an extensive area of research and has received attention given its importance on the democratic society we live in. Both theoretical [28, 36] and more applied results [1, 17, 38, 42] have been proposed. The motivation for new developments, not limited by this enumeration, include the time efficiency in the corporations' decision making process, the reduction of the economic costs of general elections or referendum, the search for an increase in the elections' participation, or

the participation of the people in the political agenda. All these factors are relevant and will have a role in the eventual generalization of electronic voting. Nevertheless, it is out of the scope of this paper the analysis of these factors and their relevance in future practical implementation of electronic voting.

Here, we present a verifiable voting scheme that, under some conditions, guarantees that the desired properties of an electronic voting system are hold. Explicitly: that it is not possible to relate a vote with the elector who cast it (privacy); that it is unfeasible for any party in the system to modify a ballot without detecting the forgery (integrity); that the final tally considers only verified and correct ballots (correctness) and any elector in the census can check that her vote was accounted as intended (verifiability).

The system we present, the Two-Authorities Voting Scheme (TAVS), can be seen as an adaptation of Chaum [12] blind signature scheme to electronic voting. TAVS is based on two unrelated authorities: an *Identification Authority* (IA) that checks the membership of a potential elector in the census; and, a *Remote Polling Station* (RPS) where the electors cast their votes. Individual validation can be achieved without the need of implement time expensive zero-knowledge procedures [5].

Our goal is to introduce a more efficient and simpler voting scheme, where simpler regards to the number of partners implied in the election, as well as to the number and complexity of the steps to carry out. The outcome of this improvements imply a reduction of the global computational complexity. The simplicity of our approach also allows a better understanding on how the system works. The advantage of this is two-fold: academics can review and discuss the protocol with ease, and voters can get a betters grasp of the scheme despite not being experts on the field. TAVS allows the elector to (anonymously) check that her vote has been included into the count. The final tally can be later anonymously audited in order to verify the correction in the count. In all the processes privacy is guaranteed.

In order to allow the anonymous certification of the elector's vote we make use of an operation with homomorphic property. In the cryptography framework, homomorphic operations allow to operate over ciphertexts equivalently as it would do on plain text without revealing the encrypted content. Blind signature schemes take advantage from homomorphic properties to permit the disguised signature of a message. For the sake of simplicity, we present the scheme taking into account the RSA signature method [33] because of the homomorphic properties of the modular exponentiation. Nevertheless, the scheme here proposed can be modified to consider any other process with homomorphic properties.

As mentioned above, it is assumed that the two authorities implied are not related in any way. Also, unlike Chaum's original proposal, no communication is established in order to share elector's information. It is not assumed the authorities are honest and provide methods to detect mali-

cious behavior of the authorities. Apart from this consideration, we do not consider many aspects that can be solved using general and well-known procedures in the literature, which can be considered as out of the scope of this paper. Thus, from now on:

- In order to be the more general the better, this paper does not address the way to code the votes. In the description of the scheme, we only consider the number that comes out from the binary representation of the vote, regardless how it is obtained. Thus, the voting scheme proposed here is described regardless the coding of the vote, and, therefore, any voting protocol is suitable to be implemented (plurality, approval, ranked, Borda systems, Condorcet systems, and single or multiple races as well, see [8] or [34]).

- It is assumed that all the auxiliary methods and procedures work properly and no weakness of the scheme can be derived from them (i.e., signature and hash functions are considered secure).

- It is assumed that the organization supporting the elections provides a private identification to every elector in the census. Taking into account particular features of the census and its geographical distribution, this identification can be implemented either physically or electronically, and distributed in many ways before the elections date. Of course, any existing suitable identifier, can be considered as well.

- The communication channels are considered to be secure. We note that the distributed identification would permit to implement secure channels to communicate the electors and the implied authorities in the voting scheme during the elections.

The rest of the paper is organized as follows: in Section 2 we review relevant papers that propose state-of-the art electronic voting systems; in Section 3 we introduce and describe in detail our proposal; in Section 4 we analyze the properties of our voting scheme and in Section 5 we show the asymptotic time-complexity of our approach. Finally, in Section 6 we present the final conclusions of this work and appraise the contributions.

2 Related work

Blinds signatures were proposed by David Chaum [12, 13] as a mechanism to allow signing a disguised message that cannot be linked to un-blinded content. The original works were based on RSA, but some signing protocols based on the discrete logarithm problem appeared later [9]. Blind signatures were defined with the following three functions in the original work:

- A signing function s private to the signer. Its corresponding public inverse s^{-1} such that $s^{-1}(s(x)) = x$ and s^{-1} gives no clue about s.
- A computing function c and its inverse c^{-1}, known only to the provider, such that $c^{-1}(s(c(x))) = s(x)$ and $c(x)$ and s give no clue about x.
- A redundancy checking predicate r, that checks for sufficient redundancy to make search for valid signatures impractical.

Blind signatures were originally presented as a way to produce untraceable electronic payments. Given the similarities between double spending and double voting, these signatures were also applied to voting systems. Indeed, the original proposal mentions this application. We adapt the results of this work to propose a complete voting protocol that takes into account new issues as identification and re-blocking. In this section we review the most relevant literature regarding voting systems: some of them related with blind signatures, some with multi-authority systems and some because of their relevance in the evolution of voting protocols. Recently, voting protocols have shifted to more recent approaches such as ring signatures (e.g: [43, 44]) or Blockchain-based methods (e.g: [45, 46]).

The first electronic voting scheme proposed was based on mix-nets in an anonymous channel in the context of mail services [10, 11]. This scheme does not require a universally trusted authority and if at least one of the authorities is honest through the cascade of mixes, the system remains private.

Cohen et al. introduced in [14] a robust and verifiable system based on the $r - th$ residue problem [4, 31]. They introduced an interactive proof which only required one partner to actively participate. The electors and the convening organization interact through a public bulletin board, all interactions are time-stamped by a global clock. An on-request beacon [32] is introduced to add a source of randomness. The main drawback of the system is that the organization has the ability to read any vote. The system works for 'Yes/No' polls and can be extended for multi-way elections, nevertheless, for a m-multi-way elections, it implies r to be the product of m large prime values, which lead to more expensive modular exponentiations.

In [16], Cramer et al. proposed a multi-authority voting system based on proofs of knowledge and the homomorphic properties of the ElGamal cryptosystem [19]. This approach uses a threshold scheme [18] to share the decryption private key among the n authorities. In this scenario, each authority commits to publicly share its secret in order to avoid malicious changes. The decryption needs at least half plus one of the n authorities to recover the secret key. The cost is linear with respect to the number of electors in the case of 'Yes/No' elections. However, a multi-way election would imply the increase of the number of zero-knowledge proofs, and, therefore, the increment of the cost of the scheme.

Juang et al. propose in [22] a robust and verifiable multi-authority system that allows abstention after the registration phase as well as opening objections to the tally without compromising the privacy of the voter. This scheme uses distributed blind signatures [9, 13] in order to disperse the power of a single authority. The authors present a quite complex architecture involving seven phases and multiple partners: electors, administrators, scrutineers and the counter, where voters, firstly, encrypt their votes and apply blind threshold signature techniques to, secondly, get their vote signed by administrators. In the voting phase, voters can craft their real encrypted votes from the blind encrypted ones. Once the vote is crafted they can send it to the counter through an untraceable electronic mail system. Votes are published, if no objections appear the scrutineers send their pieces of the secret key to the counter. Votes are decrypted and results are published. The system preserves the electors' privacy from the rest of entities of the system. The time complexity of the system is determined by the complexity of the blind threshold signature scheme and the preparation phase where authorities must cooperate between them. In this system, the number of modulo operations scales linearly with respect to the number of administrators for each voter and for each administrator.

A multi-candidate and multi-authority voting system was proposed in [3] which shares homomorphic properties, zero-knowledge proofs and a threshold system similar to the scheme presented in [16]. In their paper, the authors state that their scheme provides receipt-freeness (see, for instance, [6]), which guarantees that no elector could craft a receipt revealing how she voted even if she wanted to do so. The scheme in [3] is based on the Paillier cryptosystem [27] and is oriented to a large group of electors in which authorities are organized hierarchically. The votes move up in the hierarchy until its final decryption. It is relevant to note that, in this system the vote size depends on the number of zero-knowledge proofs, the size of hashed commitments and the size of the modulus used in the Paillier system.

Cramer et al. proposed in [15] a multi-authority voting system. Electors cast, encrypt and distribute shares of their votes and then post them on the bulletin board. Instead of relying on expensive zero-knowledge proofs, Cramer et al., propose a non-interactive proof of validity, reducing the cost of the zero-knowledge proof from quadratic to linear. Each vote on the bulletin is accompanied by one of this proofs. To alleviate the fact that a single authority could decrypt the vote, they use multiple authorities in a threshold system, similar to [16]. The system works for 'Yes/No' elections and it is compatible with plurality voting with minimal overhead. However, for other kinds of vote, the number of proofs could increase. The computational time cost of distributing the commitments and checking the shares is linear for each partner implied in the elections: voters require a linear effort with respect to the number of authorities and the size of the security parameter, and authorities require a linear effort with respect the number of voters and

the size of the security parameter.

In [24], Li et al. presented a multi-authority voting system based on blind signatures. To the best of our knowledge, it is the most similar protocol to our proposal. Electors cast their vote and blind it to get it signed from multiple authorities. Authorities are supposed to be conformed from multiple parties in the political spectrum to ensure a correct and honest functioning. The scheme of Li et al. implies four voting phases, four authorities, three pairs of keys for the authorities and two pairs for each vote. A consequence of the number of authorities and the processing of each vote is the high number of modular exponentiations needed. Some aspects of the public key infrastructure and the blind signature functions are not fully detailed. On the other hand, since the identification process depends partially on the authorities, they provide coercion resistance.

Porkodi et al. presented in [30] a scheme that operates in a similar way other voting schemes do: the elector encrypts her vote and post it on a public bulletin board; due to the homomorphic properties of the encryption, an encrypted tally can be anonymously computed from the bulletin board, which implicitly hidden the direction of the votes in the board; and, finally, decrypting the tally in the final stage. The decryption secret key is shared between the authorities in a threshold scheme. Each authority has to prove that he posted a commitment to its private share and each elector has to prove she encrypted a valid vote. After this, the final tally is computed and published. On the one hand, the system uses common operations on elliptic curves [26] that allow the use of smaller key sizes with the same level of security as other approaches. On the other hand, each vote must be accompanied by an expensive proof of knowledge, and each authority must proof the validity of their commitment.

Taking into account ElGamal cryptosystem and its homomorphic features, Philip, Simon and Oluremi propose their receipt-free multi-authority system in [29]. The system takes into account n authorities structured in a threshold scheme and a Trusted Center (TC). Before the elections, the TC receives the identification of the electors and provide a username and a pass code for those eligible electors. The information provided by the TC is used by the electors to get validated and register their vote. After each elector has encrypted and signed her vote and crafted its proof of correctness, the authorities use the distributed secret key to compute the final tally. Votes are re-encrypted through the voting process to provide receipt-freeness. A single vote must be encrypted, signed and accompanied by a proof of validity. Later, it needs to be re-encrypted. While the authors provide a functional system with receipt-freness, this comes at the cost of increasing the computational cost of the system.

Cobra [20] is a proof-of-concept election scheme that offers concurrent ballot authorization. The goal of this approach is to provide an immediate and efficient tally. Cobra allows each elector to generate fake ballots

in order to prevent elector coercion. Previous to the elections, credentials are distributed to the electors. Later, electors can claim fake credentials to cast fake ballots that can be later separated from real ballots without revealing the particular submissions [35]. They reduce ballot authorization to the membership problem. The discrimination function is implemented as a Bloom filter [7] in which querying and inserting operations are performed homomorphically to preserve privacy. The concurrent ballot authorization is the most expensive part of the system because the number of modular exponentiations it implies (which is quadratic with the number of submitted ballots). The honesty of all implied authorities is assumed. Despite the authors provide a small experimentation, due to the high cost of the system, it does not scale appropriately.

Thao and Khanh introduced in [37] an election scheme based on blind signatures and dynamic ballots. The elector needs to register herself using blind signatures through a chain of authorities, similarly as in [24]. To protect the privacy of votes from coercion, they employ dynamic ballots that change for each elector. A Ballot Center is responsible of providing a random permutation of the candidates to each voter. In the tallying phase, Plain-text-equivalence (PET) [21] is used to remove invalid and duplicated ballots. They provided a solid protocol that fixed most of the flaws previous voting systems had. However, the scheme employs 5 authorities, 2 bulletin boards, two types of encryption and a PET test for each identifier. All these factors result in an expensive system difficult to scale.

Yang et al. present in [40, 41] propose a ranked voting system where each ballot is coded as a square matrix and each element of the matrix is independently encrypted using ElGamal cryptosystem. In order to prove correctness of the ballot, the elector has to provide a proof of partial knowledge for every encrypted cell of the matrix and a zero-knowledge proof for the whole ballot. Homomorphic properties allow to combine the ballots' rows in order to compute the final tally for each candidate. Decryption process needs the cooperation of all authorities. An improvement of the initial version implies to encode in binary the ranks of each candidate in the ballot. As a consequence of this, the size of the matrix representing the ballot is reduced and therefore the time complexity is reduced. Nevertheless, the time complexity still remains the main drawback of this approach because the huge number of zero-knowledge and partial-knowledge proofs and modular exponentiations needed.

To our knowledge, Aziz presents in [2] the most recent electronic voting system based on blind signatures. In his work, he presents a multi-authority and coercion resistant scheme. To provide coercion resistance, he employs fake credentials [23] to provide the elector with an exit mechanism if it gets coerced. After registration, the elector anonymously requests a token to a token authority. This token contains the parameters needed to construct the ballot. This way, the ballot crafting is detached from the elector, making

impossible to craft a receipt. Then, blind signatures are employed to get the ballot signed from a registrar. The signed ballot is casted through a mix-net. After the election, a set of trustees cooperate to decrypt the votes and compute the final tally. Ballots are shuffled and separated in such way that there is no relation between the direction of the vote and the proof of validity. This prevents the elector from knowing if her vote was tallied as it was casted, but provides coercion resistance. Multiple authorities are involved in the process, but mainly there are a registrar assumed honest and a set of trustees that employ a distributed key generation protocol to disseminate the trust. Regarding the time complexity, the scheme requires to compute a token per elector (although this can be pre-computed to reduce the computational cost) and a unique pair of RSA keys per elector. These requirements: the multiple authorities and the use of mix-nets to cast a vote, elevate the final computational cost to vote.

3 Description of our proposal

Here we describe our voting protocol, TAVS, based on multiple authorities and blind signatures. Unlike most of the systems described in the previous section, we reduce the number of authorities to only 2. Thus, we simplify the voting procedure to just three steps and reduce the overall computational cost. Opposed to other approaches, the corruption of a single authority does not compromise the security of the voting protocol and the elector is able to check the honesty of the authorities. As mentioned in Section 1, we employ RSA in order to implement blind signatures; we fully disclose each computation involved on the signing scheme. We utilize the blinding scheme to address the identification of the elector. We contemplate re-blocking issues as well as disagreements with the authorities. Afterwards, after some minor remarks about the notation, TAVS is disclosed.

As mentioned before, in this paper TAVS is described regardless the coding of the vote. Therefore, any protocol implementing the proposed scheme should state clearly the way the vote is coded.

Once the elector has decided the direction of her vote, in order to obtain the ballot, the (coded) vote is combined with other numbers using two operations:

- The modular product of numbers modulus n (denoted with $a \cdot b \bmod n$). Usual and self-explained operation.
- The *concatenation* of numbers (denoted with $a \| b$), that returns the number obtained by concatenation of the binary representation of the input numbers.
- The multiplicative inverse of a number x modulus some given integer n (denoted with $x^{-1} \bmod n$).

TAVS consists on three sequential steps and is fully described in the following sections. Briefly speaking, the first step consists on the generation of the pre-ballot. This process is carried out by the elector with no need of interaction with any authority. The output of this process is an uncertified masked ballot (the pre-ballot) which cannot be disclosed without the elector participation. The second step implies the submission of the elector's identification and the (masked) pre-ballot to the IA in order to certificate the pre-ballot using an electronic signature procedure. This certification is carried out whenever the identification corresponds to an elector in the census who has not previously ask for another vote to certificate. The certification process ends when the elector acknowledges safe and correct receipt. In the last step, the elector anonymously submits the certified ballot and the information needed to unmask the vote to the RPS. The generation of the pre-ballot is such that it is unfeasible the manipulation of the certified version of the ballot in order of obtain more than one valid votes.

In order to maintain privacy and provide democracy and verification, two public bulletin boards are used, the *Revoked Board* and the elections' *Public Bulletin Board*. The use of the *Revoked Board* prevent malicious electors from using certified ballots before the end of the certification process. The elections' *Public Bulletin Board* allows the electors to confirm that their ballot has been counted in the final tally.

TAVS uses a generic hash function of T_H bits, and a signature scheme with homomorphic properties. As mentioned above, we present our proposal taking into account the Chaum's RSA blind signature procedure. Obviously, the scheme could be modified to consider any other signature procedure in the literature holding homomorphic properties.

Although the details will be provided in Section 3.1, we denote with T_S the size of the signature key (number of bits in the binary representation of the modulus n), which will be important in the following because of their role in the coding of the vote. Let us anticipate that, whichever format the vote could have, it is coded using $T_S - T_H - 1$ bits. This will guarantee correct operation in every case. Thus, the only consideration to be made here is that the values T_S and T_H must be big enough to suitably code the vote.

3.1 Pre-ballot generation

Before the voting process, it is necessary to agree the methods that will be used for hashing as well as an electronic signature procedure. Thus, before the start of the elections, the IA must generate an electronic public signature key and broadcast the public component of this key in order to allow to every member in the census to check the correct validation of the ballot. The RSA key generated by the IA (namely the private —signature— component $S_{IA} =< s >$ and the public —verification— component $V_{IA} =< n, v >$) will

allow to certificate the ballots and to check their correctness in the counting process.

Algorithm 3.1 Pre-ballot generation.

Input: The IA public component of the RSA signature key $V_{IA} =< n, v >$
Input: A hash function h of T_H bits.
Output: A pre-ballot ready to be (blindly) signed by the IA

1: **Method**
2: Let T_S be the number of bits needed to encode n.
3: Let *choice* be the $T_S - T_H - 1$ bits coding of the candidate/choice
4: Let $1 < mask < n$ be a (private) randomly-generated value such that $gcd(mask, n) = 1$
5: Compute $mask^{-1} \bmod n$
6: Let $hash = h(choice \| mask)$
7: $preballot = (choice \| hash) \cdot (mask^v \bmod n) \bmod n$
8: **return** $< preballot,\ mask >$
9: **End Method.**

Once the public component of the IA signature key has been distributed, the elector can generate a pre-ballot with her vote. This procedure is run by the elector independently and isolated from the other two authorities implied in the process. The procedure to generate the *pre-ballot* (ballot generated but not yet validated) is depicted in Algorithm 3.1 and it is designed to hold the following conditions:

- The pre-ballot must be concealed in such a way that no-one but the elector is able of finding out the direction of the vote (she is the only one that knows the mask).
- In order to prevent double vote, the mask must be linked to the vote so that it would be unfeasible the manipulation of the pre-ballot, which would allow a malicious elector to double vote.

First, the procedure codifies the direction of the elector's vote (line 3 in Algorithm 3.1), and selects a random generated mask that will be kept secret until the ballot is submitted to the RPS.

In order to prevent forgeries, the concatenation of the elector's choice and the mask is hashed. These three elements (the elector's choice, the randomly generated mask and the hash) are combined to obtain the pre-ballot (line 7). In order to avoid reblocking errors in the certification due to a pre-ballot greater than n value, the size of the elector's choice codification is such that the concatenation of the choice and the computed hash results in a number lower than n. The structure of the pre-ballot is depicted in Figure 1.

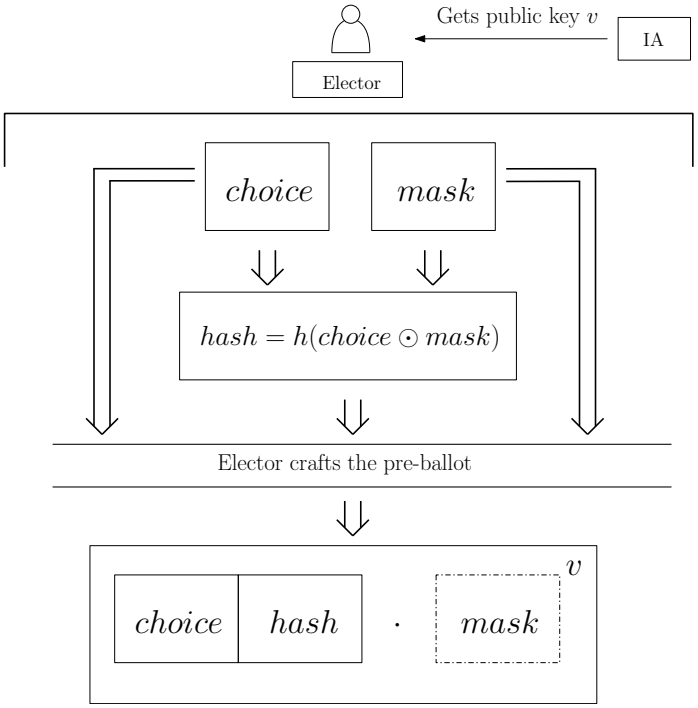

Figure 1: Pre-ballot structure. Each box represents a string. The dashed box graphically shows a modular exponentiation. The concatenation of boxes represents the concatenations of strings. The image represents the crafting and structure of the pre-ballot the elector sends to the IA (the result of applying Algorithm 3.1).

We note that the computed hash is not explicitly included in the output of Algorithm 3.1 but that it is present in an implicit way. The secrecy of the mask is important in order the system to hold the desired security properties of an electronic vote system. Once obtained the pre-ballot and the mask, the elector interacts with the IA to certificate her ballot.

3.2 Elector identification and pre-ballot certification

The second step of our system aims to the blind certification of the pre-ballot by the IA. To do so, the elector sends the (masked) pre-ballot together with her identification to the IA. The process is described in Algorithm 3.2.

We note that the IA cannot disclose the direction of the elector's vote as long the mask remains secret. This implies that the elector has total control of her vote, that if she wanted, it could even be blank or null. Once the IA has checked that the elector has not previously requested another ballot for certificate, the IA records the elector's identifier and certifies the pre-ballot signing it by using the RSA signature key publicly shared (line 8).

Algorithm 3.2 Pre-ballot certification

Input: A pre-ballot pb generated by an elector
Input: The elector identification
Input: The public $V_{IA} =< n, v >$, and private $S_{IA} =< s >$ components of IA's RSA signature key
Output: A validated ballot (pre-ballot signed by the IA) or *Forgery attempt*

1: **Method**
2: **if** the elector's identifier is already stored in the IA records **then**
3: **return** *Forgery attempt*
4: **end if**
5: Store the elector's identifier in the records
6: $endCertification = False$
7: **while not** $endCertification$ **do**
8: Compute $b = pb^s \bmod n$ //the certified ballot
9: Store b in the *Revocation Board*
10: Send b to the elector and ask for validation
11: **if** the elector's validates b **then**
12: Remove b from the Revocation Board
13: $endCertification = True$
14: **end if**
15: **end while**
16: **End Method.**

We note the effect of the certification process on the mask. We denote with pb the pre-ballot built by the elector. Because of the homomorphic properties of the modular exponentiation it follows that:

$$\begin{aligned} pb^s \bmod n &= ((choice \| hash) \cdot mask^v)^s \bmod n = \\ &= (choice \| hash)^s \cdot (mask^v)^s \bmod n = \\ &= (choice \| hash)^s \cdot mask \bmod n \end{aligned}$$

The process asks the elector to acknowledge the correct certification of the ballot, that, because of the structure of the pre-ballot, only needs the elector to compute the inverse of the mask modulus n to unmask the ballot, indeed:

$$\begin{aligned} &((choice \| hash)^s \cdot mask \bmod n) \cdot mask^{-1} \bmod n = \\ &= ((choice \| hash)^s \cdot mask \cdot mask^{-1} \bmod n = \\ &= (choice \| hash)^s \bmod n \end{aligned}$$

afterwards, it is possible to use the public component of the IA signature key to verify that the certification considered the pre-ballot previously sent

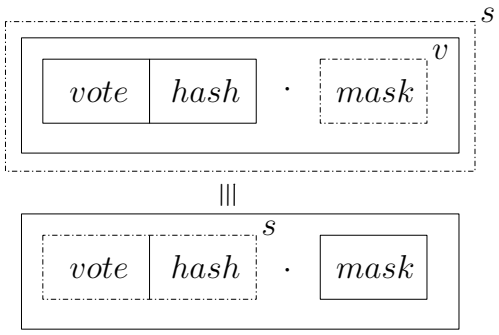

Figure 2: Certified ballot structure. The same considerations detailed in Figure 1 apply. The image graphically represents the signature process applied in Algorithm 3.2. Once validated, the IA signs the pre-ballot using its secret key. Both structures are equivalent due to the homomorphic properties of the modular exponentiation.

by the elector:

$$((choice \| hash)^s)^v \bmod n = choice \| hash.$$

If the elector acknowledges safe receipt of her ballot the certification process ends. In order to avoid a malicious elector to later claim the ballot was non-correct, which would allow double-voting, the IA stores the certified ballot in a board of revoked ballots (line 9). The ballot is removed from this board once the elector acknowledge safe and correct receipt. The certification process and the structure of the certified ballot is shown in Figure 2.

3.3 Submission and publication of the ballot

The third and last step in the voting scheme implies the submission of the certified vote to the RPS. The procedure is summarized in Algorithm 3.3.

Briefly speaking, for any valid ballot (correctly certified), an identical procedure to the one that allows the elector to verify the certification of her ballot allows the RPS to access to the masked vote (lines 5 and 6 in Algorithm 3.3). The RPS can then obtain the vote itself (line 7) and the hash that relates the vote and the mask (line 8). The computation of $h(vote \| mask)$ allows the RPS to check the integrity of the ballot (line 9). The process is depicted in Figure 3.

The assumptions we consider in the description of TAVS prevent the communication between both authorities. Nevertheless, the IA would be able to recover the direction of the vote of any elector if the public bulletin board includes some information he could link to an elector identifier. This is not possible because the hash is not available to the IA, and, therefore, its publication does not allow anyone to link the elector and her vote.

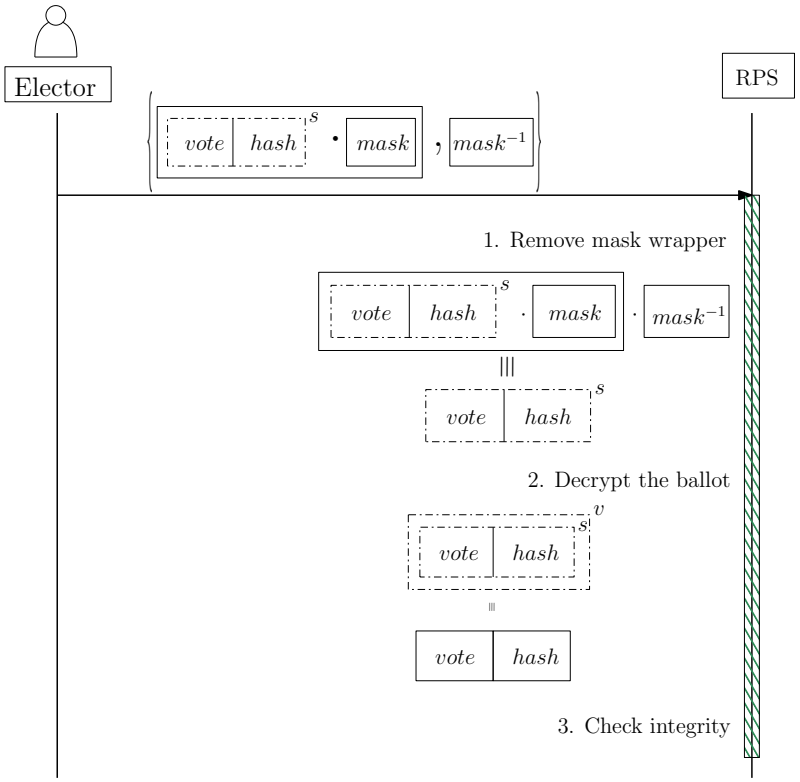

Figure 3: The image shows the submission of the ballot to the *RPS*. Once the *RPS* receives the ballot and the inverse of the mask, the Algorithm 3.3 is applied to recover the elector's vote and the hash to publish on the bulletin board.

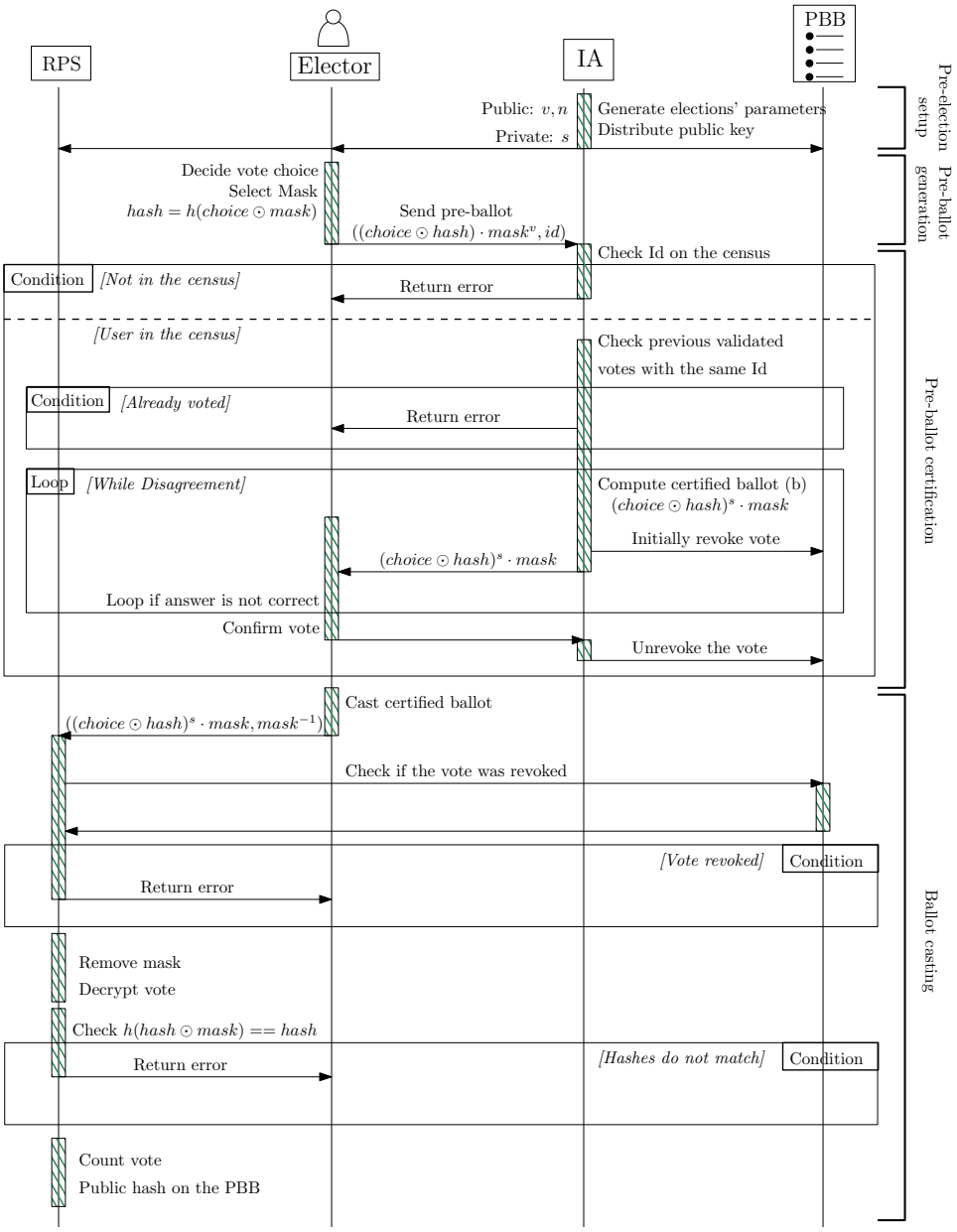

Figure 4: Timing and partners' interaction of the proposed voting scheme. The image shows the whole process an elector must complete to cast a vote. It shows the computations and the interactions needed as a time-interaction diagram.

Algorithm 3.3 Ballot casting.

Input: A certified ballot b
Input: The mask used in the generation of the pre-ballot $mask$
Input: The IA public component of the RSA signature key
Input: The agreed hash function h

1: **Method**
2: **if** b is in the *Revocation Board* **then**
3: **return** $Forgery - attempt$
4: **end if**
5: Compute $pkg = b \cdot mask^{-1} \bmod n$
6: Compute $pkg = pkg^v \bmod n$
7: Set $vote$ equal to the first $T_S - T_H - 1$ bits of pkg
8: Set $hash$ equal to the last T_H bits of pkg
9: **if** $hash == h(vote \| mask)$ **then**
10: Store the casted $vote$
11: Publish $hash$ in the elections' *Public Bulletin Board*
12: **return** $Correct$
13: **else**
14: **return** $Forgery\ attempt$
15: **end if**
16: **End Method.**

Figure 4 depicts the whole voting process: the interactions between the partners; the timing in the voting scheme; and, the conditions that trigger different consequences as well. As it can be seen, the process is initiated by the IA by making public the parameters and the public key components. Then, the elector carries out Algorithm 3.1 and sends the pre-ballot to the IA. Assuming the elector is on the census and did not vote before, the IA responds the elector with a certified ballot which is considered invalid until the elector confirms the vote. The elector can independently check the validity of the received certified ballot and contest the IA if it does not match her initial ballot. After the certification phase, the elector send her ballot to the RPS. There, it is checked if the vote was revoked by the IA. If the vote is valid, RPS follows Algorithm 3.3 to check the vote is correctly crafted. If the algorithm outputs true, the vote is counted and its hash is posted on the public bulletin.

4 Properties of the voting scheme

In this section we analyze the security of the proposed approach, and, in order to do so, we first note the reliability of RSA as well as of the many hash functions in the literature that are suitable to be used in the scheme.

We also note that most of the properties rely on the safety of blind signatures as well the fact the authorities purely perceive partial information.

We now analyze whether or not the proposed scheme holds the properties that every voting scheme should fulfill.

Democracy/Eligibility (only those included in the census can be considered as valid electors). Actually, the scheme here proposed does not prevent malicious electors to access the RPS. Nevertheless, on the one hand, we note that it is not feasible for adversaries to construct a certified ballot unless they could gain access to the IA signature private key. On the other hand, in order to impersonate an elector it is necessary to know the identification of the elector, which is assumed to be private.

Uniqueness (the electors can only vote once; double voting is not allowed). Indeed, only one ballot per elector can be certified by the IA. We now prove that it is not feasible to tamper with a certified ballot, nor to design a pre-ballot in order to achieve double voting.

Let us recall that once a ballot is certified by the IA it is of the form $(vote\|hash)^s \cdot mask$, where $hash = h(vote\|mask)$. Obtaining another valid ballot from it implies:

- That $(vote\|hash)^s$ can be considered as a mask of a (tampered) $vote'$ coded into $mask$.

  Indeed, it is feasible that there will exist the inverse of $(vote\|hash)^s$ modulus n and therefore a valid mask for $vote'$. Nevertheless, taking into account the construction described in Algorithm 3.1, to tamper with the ballot it would also be necessary that $mask$ were such that:

  $$mask = vote'\|hash',$$

  where $hash' = h(vote'\|(vote\|hash)^s)$. This is highly unfeasible taking into account that the certification (by signature) can only be carried out by the IA and the result is unknown for any elector before the certification process. The hash acts like a redundancy parameter, making the search for manipulated masks unfeasible. This is related with the third property needed for blind signatures stated by Chaum (described in Section 2). This proves the search for valid blind signatures, through tampered masks, to be impracticable.

- That the construction of the ballot considers a $mask$ that can be modified after the certification of the ballot in order to obtain a new (tampered) vote.

  Taking into account again the construction of the ballot, the computation of $hash = h(vote\|mask)$, implies that any modification of the mask results in a different hash and, therefore in a different ballot. Therefore, such process is highly unfeasible.

Taking into account the two points we just described, it is proved that the uniqueness of the voting is granted. No double voting or tampering is feasible.

Privacy (it is not possible to relate a vote with the elector who casted it). The proposed scheme forces the elector to interact with two different authorities. Nevertheless, the IA has access to the elector identifier but he cannot unmask the vote (it is necessary to know the mask to do it), and the RPS has access to the direction of the vote but the elector identifier is not sent to him. Therefore, it is impossible to relate an elector with the direction of her vote unless both authorities share information, which contradicts an assumption of the scheme.

Integrity (it is unfeasible for any partner in the system to modify a ballot without detecting the forgery). We recall previous comments on the uniqueness of the vote that takes into account the construction of the ballot described in Algorithm 3.1. A similar reasoning allows to state that it is unfeasible for any partner to modify a previously certified vote.

Correctness (the final tally considers only verified and correct ballots). Indeed, TAVS guarantees that no invalid votes are counted. We also note that the RPS has enough information to be audited. Again, without access to the IA's signature (private) key, it is unlikely to design a method able to substitute a set of valid certified votes by another set of forged ones.

Elector verifiability (any elector in the census can verify that her vote has been taken into account in the way it was cast) and *Auditability*. As it is proposed here, TAVS scheme provides the elector evidence that her vote was included in the final tally. The participation of an Audit Authority would provide the electors the confidence their votes have been counted in the direction they were cast. We note that, because the unfeasibility of tampering with the ballots, the Audit Authority is able verify the tally using only the information stored in the RPS.

Coercion resistance (an elector cannot prove how she voted or can lie about the direction of her vote). The problem of voter coercion was introduced by Juels in [23, 39] and it describes a situation where the elector might be intimidated to vote in certain way. Coercion resistance and the problem of vote selling are also closely related to the concept of receipt-freeness. Furthermore, there is a trade-off between coercion resistance and performance, because of the systems that provide receipt-freeness tend to be more complicated and sacrifice some efficiency. In this work, we do not focus on this dichotomy. We provide a system where the elector has mechanisms to be certain that her vote was considered, as well as enough information for an independent Audit Authority to anonymously check the correctness of the tally.

Similarly as Cobra system [20] does, it is interesting to explore how to give the elector the ability for generating fake ballots as a tool to mock the coercer's desires. This would imply the modification of the vote authoriza-

tion process to: first, allow the certification of more than one ballot; second, to allow the tracking of the set of (potentially more than one) duplicated valid votes that must be processed in order to select the one to be include in the tally; and, third, to guarantee that fake votes are detected and removed from the tally.

5 Time complexity analysis

We devote this section to analyze the time complexity of TAVS. We prove that our system scales linearly with the number of votes. Generally, the papers in the literature use operations modulo n (addition, multiplication, exponentiation) as the unit of computational cost, concatenation is not considered. Nevertheless, we are going to analyze the time-complexity in terms of bit operations, analyzing in detail the complexity of different operations. In the following, n denotes the number in the operations and $\log n$ denote its number of bits. It is well known that modular addition and subtraction have time complexity of $\mathcal{O}(\log n)$ bit operations, multiplication and inverse have a complexity of $\mathcal{O}(\log^2 n)$ and modular integer exponentiation is the most expensive operation with $\mathcal{O}(\log^3 n)$ time complexity. We make use of a hash function, which we recommend it to be a recognized and established one[1]. Here, we assume the complexity of applying the hash function is linear with respect to the input.

5.1 Pre-ballot generation

Once the parameters of the election have been decided, the elector must follow Algorithm 3.1:

1. The elector must select a random mask in line 5 of Algorithm 3.1, the mask must be invertible. Since we require $gcd(mask, n) = 1$ the inverse exists. To find the inverse we suggest extended Euclid's algorithm [25] which has a complexity of $\mathcal{O}(\log^2 n)$ bit operations.

2. Applying the hash function of line 6 requires linear effort with respect the input. In this case, the input is the concatenation of the choice and the mask. This results in a complexity of $\mathcal{O}(\log n - T_H - 1 + \log n) = \mathcal{O}(\log n)$ bit operations.

3. To craft the pre-ballot in line 7 of the algorithm, a multiplication and a modular exponentiation are needed.

The time complexity of Algorithm 3.1 can be expressed as:

$$\mathcal{O}(\log^2 n) + \mathcal{O}(\log n) + \mathcal{O}(\log^2 n) + \mathcal{O}(\log^3 n) = \mathcal{O}(\log^3 n)$$

[1]Hash functions as the SHA family or even a modification of DES will work. We leave this decision to the reader since standards may change.

Observe that modular integer exponentiation determines the time complexity of the algorithm.

5.2 Pre-ballot certification

For Algorithm 3.2 we assume the best case: IA is benign, the channel is secure and the elector will validate the ballot. Computing the certified ballot requires:

1. One modular multiplication and one modular exponentiation are required in line 8 to sign the ballot.
2. To check if the received certified ballot is correct, the elector needs to perform one multiplication and one modular exponentiation in line 11. It is not explicitly shown on Algorithm 3.2 but it can be seen on Figure 4.

The time complexity of Algorithm 3.2 is $\mathcal{O}(\log^2 n)+\mathcal{O}(\log^3 n) = \mathcal{O}(\log^3 n)$ bit operations for the elector and $\mathcal{O}(\log^2 n) + \mathcal{O}(\log^3 n) = \mathcal{O}(\log^3 n)$ bit operations, per ballot, for the IA. Again, modular exponentiation determines the time complexity of the algorithm.

5.3 Ballot Casting

Once the elector sends the ballot to the RPS she does not need to perform more computations. The RPS takes part on Algorithm 3.3 and performs the following to decrypt the vote:

1. A multiplication is used in line 5 to remove the mask wrapper from the ballot.
2. A modular exponentiation is needed in line 6 to recover the vote.
3. To check the integrity of the ballot the hash function must be applied in line 9 with $\mathcal{O}(\log n - T_H - 1 + \log n) = \mathcal{O}(\log n)$ bit operations.

The time complexity of Algorithm 3.3 is $\mathcal{O}(\log^2 n)+\mathcal{O}(\log^3 n)+\mathcal{O}(\log n) = \mathcal{O}(\log^3 n)$ bit operations per processed ballot. As in the previous algorithms, the complexity is determined by the modular exponentiation operation.

5.4 TAVS complexity

Here we summarize the total complexity of the system. We differentiate between the complexity for the elector to cast a vote and the complexity for the system to undertake the whole election process. The elector must perform a total of $\mathcal{O}(\log^3 n)$ bit operations to vote (Algorithms 3.1 and

3.2). TAVS system must carry out a total of $\mathcal{O}(\log^3 n)$ bit operations per processed vote (Algorithms 3.2 and 3.3).

The computation of modular exponentiations determines time complexity of our method. If this operation is considered as the atomic operation in the analysis of the complexity, as many papers in the literature do, it is clear that the elector computes a constant number of exponentiations to vote. The same applies to the involved authorities: they compute a constant number of exponentiations per ballot. Therefore, the complexity of the system scales linearly with the number of votes to be processed. Since the complexity of the system does not depend on the number of candidates, the number of authorities involved, or the encoding of the vote, we can state that TAVS can be easily extended and scaled to adapt more general situations.

5.5 Comparison with other systems

We devote this section to compare the performance of our system with those reviewed in Section 2. Let us here note that some of the papers in the literature do not provide a detailed time complexity analysis of the system they present, or do not fully detail it; some papers consider a centralized authority and the decentralization is let as a future or theoretical exercise. Here we present a comparison of the theoretical time complexity of previous methods in the literature and TAVS.

In order to state the time complexity of the methods, we consider exclusively the number of modular exponentiations (the most expensive operation). The complexity is then expressed as an asymptotic function on the number of bit operations. When the methods do not specify the protocol used (e.g: which kind of zero knowledge proof was used) we introduce a new variable in the cost analysis. For the case of interactive zero/partial-knowledge proofs, we will assume just the first iteration is carried out, although usually the interaction needs to be repeated j times to reach certain level of trust. If possible, we group the different number of authorities involved as a single variable. Multi-candidate elections scenarios are considered. Table 1 shows the results of this comparison. Appendix A explains the details of values shown in Table 1. We also differentiate between the costs assumed by the elector and the costs associated to the system to process a single vote. The work from Porkodi et al. [30] is not included since it is based on elliptic curves.

	Elector's Cost	System's Cost per vote
Chaum [11]	$\mathcal{O}(m \log^3 n)$	$\mathcal{O}(m \log^3 n)$
Cohen et al. [14]	$\mathcal{O}(\log^5 n)$	$\mathcal{O}(\log^5 n)$
Crammer et al. [16]	$\mathcal{O}(c \log^3 n)$	$\mathcal{O}(ac \log^3 n)$
Juang et al. [22]	$\mathcal{O}(m \log^3 n)$	$\mathcal{O}(a \log^3 n)$
Baudron et al. [3]	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(a \log^3 n)$
Crammer et al. [15]	$\mathcal{O}(t \log^3 n))$	$\mathcal{O}(a \log^3 n))$
Li et al. [24]	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(\log^3 n)$
Adewole et al. [29]	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(a \log^3 n)$
Essex et al. [20]	$\mathcal{O}(c \log^3 n)$	$\mathcal{O}(ac \log^3 n)$
Thao et al. [37]	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(\log^3 n)$
Yang et al. [40]	$\mathcal{O}(c(\log P) \log^3 n)$	$\mathcal{O}(ac(\log P) \log^3 n)$
Aziz [2]	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(\log^3 n)$
TAVS	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(\log^3 n)$

Table 1: Table representing the asymptotic cost of the work performed by the elector and the system in number of bit operations. In the table: m references the number of layers on a mix-net, a represents the number of authorities, c represents the number of candidates in the election, t is the number of different shares a vote is fragmented and P the number of points awarded in a ranked voting system.

In Table 1, the methods presented by Li et al., Thao et al., and Aziz have the same time-complexity as TAVS. However, there are important details to take into account because directly affect to the performance of the methods (as we mentioned on Section 5.4, modular exponentiations dominate the time-complexity functions and hide other costs of the systems such as key generation, random permutations, other modular operation etc.)

Li et al. present a method with a heavy public key infrastructure: they generate an asymmetric key for each eligible elector and a second one for each registered elector at runtime. The use of 4 authorities makes the cost of processing a vote 7 times more expensive when compared to TAVS. Their signature step is similar to our proposal, but in our case, the elector does not depend on an authority to craft the ballot.

Aziz's method needs to generate tokens for each eligible elector and keys during the election process. This makes the cost to process a vote 7 times higher than our approach. Aziz also shows that the voter must cast the vote via an anonymous channel, such as a mix-net. The time complexity of the vote casting is not measured since it is not specified on the original work, but it should be taken into account.

Thao et al. also need to generate keys and dynamic ballots at runtime for each elector. They rely on PET tests to check the elector's identifiers and need up to 5 authorities to operate the system. These factors almost

triple the cost to process a vote with respect to TAVS.

Summarizing, TAVS does not need to compute keys per user and reduces the number authorities: TAVS is the most efficient of the presented voting protocols. Also note that, contrary to other methods, the elector can independently craft her own ballot, and only needs to interact with 2 authorities along the voting process.

6 Conclusions

In this paper, we propose a two authorities voting system. Our proposal, under the assumption that the authorities are unrelated entities, provides the desired properties of any secure voting protocol. We proved our approach is correct and efficient, not relying on expensive interactive zero-knowledge proofs and being based on simple RSA primitives, which makes our method scalable and suitable for real case scenarios. Section 5 proves that the work for the elector and the complexity of the whole system is minimal. Our approach provides significant improvements with respect to some of the reviewed systems, since it not based on time-demanding interactive proofs [14, 15, 16, 40], or complex architectures [3, 22]. The vote encoding is flexible and minimal and allows multiple types of elections (yes-no polls, plurality, multi-candidate, ranked elections) with no overhead. Traffic of messages can be checked by the elector to ensure the integrity of the system and sensible information is published through public bulletins that can be later audited by a third party.

It will be interesting to study the possibility of implement the identification of the electors by using an independent, public and scrutinable data structure. This would allow to reduce the weight of the assumption of two unrelated entities. It will be also interesting to include mechanisms to allow the elector to certificate more than one ballot as a way to bypass coercers. We will address the study of both in future works. In order to empirically test our proposal with real data it is mandatory to grant a solid infrastructure which is, unfortunately, out of the reach of our group. Nevertheless, we think this experimentation is very important and we consider to tackle it as future work.

References

[1] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348, 2008.

[2] Ahsan Aziz. Coercion-resistant e-voting scheme with blind signa-

tures. In *Cybersecurity and Cyberforensics Conference, CCC 2019, Melbourne, Australia, May 8-9, 2019*, pages 143–151, 2019.

[3] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 2001, Newport, Rhode Island, USA, August 26-29, 2001*, pages 274–283, 2001.

[4] Michael Ben-Or. Probabilistic algorithms in finite fields. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 394–398. IEEE, 1981.

[5] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 37–56, 1988.

[6] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 544–553, 1994.

[7] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[8] Steven Brams and Peter C Fishburn. *Approval voting*. Springer Science & Business Media, 2007.

[9] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. Blind signatures based on the discrete logarithm problem. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 428–432, 1994.

[10] Thomas E. Carroll and Daniel Grosu. A secure and anonymous voter-controlled election scheme. *J. Network and Computer Applications*, 32(3):599–606, 2009.

[11] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[12] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203, 1982.

[13] David Chaum. Blind signature system. In *Advances in Cryptology - EUROCRYPT '84*, pages 153–153. Springer, 1984.

[14] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 372–382, 1985.

[15] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 72–83, 1996.

[16] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, 1997.

[17] Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '09, Montreal, Canada, August 10-11, 2009*, 2009.

[18] Yvo Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–458, 1994.

[19] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[20] Aleksander Essex, Jeremy Clark, and Urs Hengartner. Cobra: Toward concurrent ballot authorization for internet voting. In *2012 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '12, Bellevue, WA, USA, August 6-7, 2012*, 2012.

[21] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, pages 162–177, 2000.

[22] Wen-Shenq Juang, Chin-Laung Lei, and Horng-Twu Liaw. A verifiable multi-authority secret election allowing abstention from voting. *Comput. J.*, 45(6):672–682, 2002.

[23] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections, New Directions in Electronic Voting*, pages 37–63, 2010.

[24] Chun-Ta Li, Min-Shiang Hwang, and Yan-Chi Lai. A verifiable electronic voting scheme over the internet. In *Sixth International Conference on Information Technology: New Generations, ITNG 2009, Las Vegas, Nevada, USA, 27-29 April 2009*, pages 449–454, 2009.

[25] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[26] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pages 417–426, 1985.

[27] Pascal Paillier. Public-key cryptosystem based on discrete logarithm residues. *EUROCRYPT 1999*, 1999.

[28] Behrooz Parhami. Voting algorithms. *IEEE transactions on reliability*, 43(4):617–629, 1994.

[29] Adewole A Philip, Sodiya Adesina Simon, and Arowolo Oluremi. A receipt-free multi-authority e-voting system. *International Journal of Computer Applications*, 30(6):15–23, 2011.

[30] Chinniah Porkodi, Ramalingam Arumuganathan, and Krishnasamy Vidya. Multi-authority electronic voting scheme based on elliptic curves. *I. J. Network Security*, 12(2):84–91, 2011.

[31] Michael O Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on computing*, 9(2):273–280, 1980.

[32] Michael O. Rabin. Transaction protection by beacons. *J. Comput. Syst. Sci.*, 27(2):256–267, 1983.

[33] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[34] Donald G. Saari. *Geometry of voting*. Springer Science & Business Media, 2012.

[35] Michael Schläpfer, Rolf Haenni, Reto E. Koenig, and Oliver Spycher. Efficient vote authorization in coercion-resistant internet voting. In *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, pages 71–88, 2011.

[36] Michael I Shamos. Electronic voting-evaluating the threat. In *Third Conference on Computers, Freedom and Privacy, CPSR*, 1993.

[37] Ai Thao Nguyen Thi and Tran Khanh Dang. Enhanced security in internet voting protocol using blind signature and dynamic ballots. *Electronic Commerce Research*, 13(3):257–272, 2013.

[38] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From helios to zeus. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013*, 2013.

[39] Zhen Yu Wu, Ju-Chuan Wu, Sung-Chiang Lin, and Charlotte Wang. An electronic voting mechanism for fighting bribery and coercion. *J. Network and Computer Applications*, 40:139–150, 2014.

[40] Xuechao Yang, Xun Yi, Surya Nepal, Andrei Kelarev, and Fengling Han. A secure verifiable ranked choice online voting system based on homomorphic encryption. *IEEE Access*, 6:20506–20519, 2018.

[41] Xuechao Yang, Xun Yi, Caspar Ryan, Ron G. van Schyndel, Fengling Han, Surya Nepal, and Andy Song. A verifiable ranked choice internet voting system. In *Web Information Systems Engineering - WISE 2017 - 18th International Conference, Puschino, Russia, October 7-11, 2017, Proceedings, Part II*, pages 490–501, 2017.

[42] Xun Yi and Eiji Okamoto. Practical internet voting system. *J. Network and Computer Applications*, 36(1):378–387, 2013.

[43] José Luis Salazar, Joan Josep Piles, José Ruíz-Mas, and José María Moreno-Jiménez. Security approaches in e-cognocracy. *Computer Standards & Interfaces*, 32(5-6):256–265, 2010.

[44] Guomin Chen, Chunhui Wu, Wei Han, Xiaofeng Chen, Hyunrok Lee, and Kwangjo Kim. A new receipt-free voting scheme based on linkable ring signature for designated verifiers. In *2008 International Conference on Embedded Software and Systems Symposia*, pages 18–23. IEEE, 2008.

[45] Ahmed Ben Ayed. A conceptual secure blockchain-based electronic voting system. *International Journal of Network Security & Its Applications*, 9(3):01–09, 2017.

[46] Pavel Tarasov and Hitesh Tewari. Internet voting using zcash. *IACR Cryptology ePrint Archive*, 2017:585, 2017.

A Time complexity analysis

In this appendix we present a more detailed analysis of the complexity of the methods in the literature (results shown in Table 1). Unfortunately, not all the revised papers report the same level of detail. This is our best effort to analyze and compare their complexities. For a more complete description of the algorithms we refer the interested reader to the original papers.

- **Chaum [11]:** An elector needs to encrypt, using RSA, the vote and re-encrypt it as many times as layers m has the mix-net. The cost for the voter depends on m, $\mathcal{O}(m(log^3 n)$. Each vote needs to be encrypted $m+1$ times, shuffled and latter on, decrypted $m+1$ times. The cost can be computed as $2\log^3 n(m+1) \approx \mathcal{O}(m \log^3 n)$.

- **Cohen et al. [14]:** The authors report a 4 phase protocol. These phases include several modular exponentiations and nested iterations. The elector needs to carry out 2 of these phases while the authorities have to complete the other two. The authors expose that the total expected time required by both the authorities and the voter phases is $\mathcal{O}(\log^5 n)$.

- **Crammer et al. [16]:** To cast a vote, the elector needs to perform an encryption of the vote using ElGamal cryptosystem and a proof of partial knowledge. This encryption requires 2 modular exponentiations and the proof of validity, in an election wit c candidates, requires $4c$ exponentiations. The total cost for the voter, in bit operations, is $2 + 4c \log^3 n \approx \mathcal{O}(c \log^3 n)$. Each authority a involved needs to broadcast a proof of its private share, check the proofs of validity of the users and cooperate between them to decrypt the final tally. The total cost can be regarded as: $a + ac4 \log^3 n \approx \mathcal{O}(ac \log^3 n)$.

- **Juang et al. [22]:** The encryption of the vote requires 1 modular exponentiation. Later, each elector needs to send its encrypted vote to each administrator and construct the signature from the responses she got. This requires 6 modular exponentiations. To vote she sends the vote through an untraceable e-mail system (mix-net) requiring m more exponentiations. The final cost is $(7+m)\log^3 n \approx \mathcal{O}(m \log^3 n)$. To carry out the blinding signature scheme each authority performs 4 modular exponentiations, an special authority called the counter, performs 2 for each vote. Previous to this, an expensive preparation setup is carried out, we do not count it since it can be performed offline. The cost can be expressed as $4a \log^3 n + 2\log^3 n \approx \mathcal{O}(a \log^3 n)$ bit operations.

- **Baudron et al. [3]:** This approach defines a hierarchical arrangement of authorities, the original paper uses three levels and we denote

the number of levels as l. The elector needs to encrypt his vote for an authority in each level, requiring l modular exponentiations. She also needs to provide zero-knowledge proofs for every encryption, and an extra proof to show all the encryptions contain he same vote. This results in $l+1$ zero-knowledge proofs, each requiring 6 modular exponentiations. Assuming l is a small constant, the cost can be computed as $(l+6(l+1))\log^3 n = (7l+1)\log^3 n \approx \mathcal{O}(\log^3 n)$. Authorities need to interact with the voter for the zero-knowledge proof, this requires 3 modular exponentiations. Each authority needs to compute its local result and forward it to its immediate superior authority. Authorities a in the same level must cooperate to partially decrypt the results, they also must provide a partial proof of decryption. The whole system cost can be expressed as $3a + aP_p \log^3 n \approx \mathcal{O}(a \log^3 n)$, being P_p the cost of the partial proof.

- **Crammer et al. [15]:** The elector needs to encrypt her vote, using 2 modular exponentiations and the proof of validity, which requires 4 exponentiations. The vote is split in t shares, the shares are sent to t different authorities together with a commitment. Each commitment requires 2 modular exponentiations. The cost for the voter is $(2+4+2t)\log^3 n \approx \mathcal{O}(t \log^3 n)$. Each authority checks the proofs of validity of the users, this requires of 4 modular exponentiations. Each tallying authority checks the t shares posted by an authority employing $a_2 t$ modular exponentiations. The final cost can be computed as $(4a_1 + a_2 t,$ being $a_1)\log^3 n$ the authorities and a_2 the tallying authorities. Since t depends on the number votes, and $a1, a_2$ can be grouped in a authorities the cost can be expressed as $\mathcal{O}(a \log^3 n)$.

- **Li et al. [24]:** The elector must interact multiple times with the different authorities to get verified, get a blind signature and cast her encrypted vote. This results in a total of 15 modular exponentiations to cast a vote: $15 \log^3 n \approx \mathcal{O}(\log^3 n)$. Authorities must generate a pair of public keys for each possible elector, but since this can be pre-computed we will ignore it in the cost function. Authorities must also generate another pair of keys for each registered voter, we will count the cost of finding the inverse in the RSA[2] which is $\mathcal{O}(\log^2 n)$ as we saw when analyzing TAVS. Furthermore the authorities nedd to perform 14 modular exponentiations to decrypt, verify, sign and re-encrypt each ballot. The cost of the system can be expressed as $\mathcal{O}(\log^2 n) + 14 \log^3 n \approx \mathcal{O}(\log^3 n)$.

- **Adewole et al. [29]:** Each voter needs to encrypt and sign her

[2]There are also other costs when computing RSA keys such as primality tests and the totient function, however we prefer to keep it simple for the comparison. We refer to the original work for more information about the voting protocol.

vote using ElGamal encryption. In addition to this, they also need to craft a zero knowledge proof for the vote. These processes require 5 modular exponentiations, the total cost for the voter remains constant $5\log^3 n \approx \mathcal{O}(\log^3 n)$. Each authority a needs to provide a commitment of his share of the secret key, this uses 1 modular exponentiation. Each re-encrypted vote needs to be decrypted, requiring 2 modular exponentiations. The zero-knowledge proofs must be checked by each authority. Finally, the authorities cooperate to decrypt the valid votes, this process demands 2 modular exponentiations per vote. The total cost can be expressed as $(a + 2 + a + 2)\log^3 n) \approx \mathcal{O}(a\log^3 n))$.

- **Essex et al. [20]:** They report an exhaustive analysis of their system. We ignore the elevated number of modular exponentiations in the registration phase since it can be performed before the elections. To cast a vote and submitting a ballot and its credential, the elector needs to perform $(8c + 4)\log^3 n \approx \mathcal{O}(c\log^3 n)$ bit operations. The number of ballots b can be larger than the number of electors since they can generate fake ballots. The most expensive part of the system is the ballot authorization process. The total cost, in number of bit operations, can be expressed as $((4c + 4)b + (280a + 12ca + 19a + 2)b + 3ac)\log^3 n) \approx \mathcal{O}(ac\log^3 n)$. Being a the number of authorities and c the number of candidates.

- **Thao et al. [37]:** The elector has to communicate multiple times with the authorities to get identified and get her dynamic ballot. These communications are encrypted using public key cryptography. This results in 10 modular exponentiantions for the user, with a cost of $10\log^3 n \approx \mathcal{O}(\log^3 n)$ bit operations. The cost of the system is defined by the checks and signatures of certificates (5 modular exponentiations) and the generation of keys for each registered elector. The total cost, in number of bit operations can be expressed as $5\log^3 n + 2\log^2 n \approx \mathcal{O}(\log^3 n)$.

- **Yang et al. [40]:** In this ranked voting system, the elector assigns P points between c candidates. The vote is constructed as a matrix with c rows and $\log P$ columns. The elector must encrypt all the cells in the matrix, requiring 3 modular exponentiations each encryption. She also needs to provide a partial-knowledge proof for each cell, and an extra zero-knowledge ´proof for the whole ballot. Partial-knowledge proofs require 8 modular exponentiations each and the zero-knowledge proof requires 5 modular exponentiations. Finally, the ballot must be signed, requiring 2 modular exponentiations. The work done by the elector can be summed up as $(8c(\log P) + 5 + 2)\log^3 n \approx \mathcal{O}(c(\log P)\log^3 n)$. The proofs posted by the voters must be checked. The cost for partial-knowledge proofs is 6 modular exponentiations,

and 4 for zero-knowledge proofs. In addition, all authorities must broadcast a commitment of their private key share, this requires 1 modular exponentiation. The performance of the whole system can be expressed as $(6c(\log P) + 4 + a)\log^3 n \approx \mathcal{O}(ac(\log P)\log^3 n)$.

- **Aziz [2]:** The elector has to request a token, check the signature of the ballot and cast her vote. This results in 5 modular exponentiations with a cost of $5\log^3 n \approx \mathcal{O}(\log^3 n)$ bit operations. The authorities must generate a token and generate a public/private key pair for each elector, and perform the blind signature. The total cost of the system can be expressed, in terms of bit operations, as $2\log^2 n + 14\log^3 n \approx \mathcal{O}(\log^3 n)$.