mathematics

Article

NetVote: A Strict-Coercion Resistance Re-Voting Based Internet Voting Scheme with Linear Filtering †

Iñigo Querejeta-Azurmendi [1,2,*], David Arroyo Guardeño [2], Jorge L. Hernández-Ardieta [1] and Luis Hernández Encinas [2]

1. Departamento de Informática, Universidad Carlos III de Madrid, 28911 Leganés, Spain; jlhernan@inf.uc3m.es
2. Instituto de Tecnologías Físicas y de la Información (ITEFI), Consejo Superior de Investigaciones Cientificas (CSIC), 28006 Madrid, Spain; david.arroyo@csic.es (D.A.G.); luis@iec.csic.es (L.H.E.)

* Correspondence: azinig@querejeta.me

† This paper is an extended of: Querejeta-Azurmendi, I.; Hernández Encinas, L.; Arroyo Guardeño, D.; Hernandez-Ardieta, J.L. An internet voting proposal towards improving usability and coercion resistance. Proceedings of the International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019), Seville, Spain, 13–15 May 2019.

Received: 7 August 2020; Accepted: 14 September 2020; Published: 18 September 2020

Abstract: This paper proposes NetVote, an internet voting protocol where usability and ease in deployment are a priority. We introduce the notion of strict coercion resistance, to distinguish between vote-buying and coercion resistance. We propose a protocol with ballot secrecy, practical everlasting privacy, verifiability and strict coercion resistance in the re-voting setting. Coercion is mitigated via a random dummy vote padding strategy to hide voting patterns and make re-voting deniable. This allows us to build a filtering phase with linear complexity, based on zero knowledge proofs to ensure correctness while maintaining privacy of the process. Voting tokens are formed by anonymous credentials and pseudorandom identifiers, achieving practical everlasting privacy, where even if dealing with a future computationally unbounded adversary, vote intention is still hidden. It is not assumed for voters to own cryptographic keys prior to the election, nor store cryptographic material during the election. This property allows voters not only to vote multiple times, but also from different devices each time, granting the voter a vote-from-anywhere experience. This paper builds on top of the paper published in CISIS'19. In this version, we modify the filtering. Moreover, we formally define the padding technique, which allows us to perform the linear filtering scheme. Similarly we provide more details on the protocol itself and include a section of the security analysis, where we include the formal definitions of strict coercion resistance and a game based definition of practical everlasting privacy. Finally, we prove that NetVote satisfies them all.

Keywords: internet voting; coercion-resistance; data privacy; zero knowledge; homomorphic encryption; usability

1. Introduction

Democracy is one of the biggest achievements of our society with its main pillar being elections, and that is why any change in the electoral process needs a very detailed study. The digitalisation of polls, while still going slower than any other field of society, is starting to become a developed trend, and even if some countries have drawn back lately for fear of not having the ability to have high levels of auditability [1,2], the list of countries using electronic devices to assist in the ballot cast or tallying process keeps growing, with special focus in the developing world [3]. This is known as electronic voting (e-voting), which comprises not only internet voting (i-voting) schemes, but also presence schemes that use electronic means for voting, tallying or verification purposes.

Traditional presence elections have the control in the environment where the voters cast a ballot, ensuring privacy of vote cast and allowing auditability of the vote cast method, which makes the process of digitalisation of presence elections faster in comparison with the one of remote elections. The latter has several differences compared to the former. Firstly, the voter may cast the vote under any circumstances, resulting in a high possibility of coercion or vote-selling. Secondly, the vote cast will go through an uncontrolled channel. Thirdly, remote authentication increases the chances of impersonation, and finally, in the specific case of i-voting, the voter casts a vote from their own device, making it difficult to do a massive-scale, effective security assessment.

The next natural step for absentee voters is i-voting and whether it is because the number of expats grows, people living abroad want to get more politically involved, or simply because voters want to cast their vote from their homes, the number of remote voters is growing [4,5]. However, many countries (e.g., Germany, Great Britain, Spain, Mexico) are reluctant to use i-voting, and prefer, on the contrary, postal voting. A proposal of the former offering security, usability, low deployment requirements and low complexity has still not been presented, and this pushes countries to keep using postal voting as a remote voting system, even if properties such as privacy may be compromised. For example, in some countries [6–8], to cast an accepted postal vote, one has to send their sealed vote together with a certificate (which contains identification of the voter) validating their right to vote together in an envelope. This creates a direct relation between identity and vote choice, making it possible for a compromise of ballot privacy. Furthermore, a full chain-of-custody cannot be made, trust has to be given to external entities (postal office) and coercion can easily happen. Hence, i-voting to replace remote paper based elections would not only offer a reduction of tallying time, but mainly, could offer more security while improving usability towards the voter. Current literature presents many solutions offering ballot privacy, coercion resistance, verifiability, or everlasting privacy (these concepts will become clear in Section 3) with very low trust requirements. However, most of these solutions are not usable, either by the complication towards the voter or by the complexity of the tallying process. The existing state of the art providing a coercion resistant internet voting scheme under reasonable assumptions proposes a poly-logarithmic tallying stage. We further reduce the filtering process in this work to linear.

Remote authentication is another main topic in remote voting schemes, both by the difficulty of a proper solution and by its importance in a remote election. Digital authentication still remains a fragile topic, however, it is tenable that using cryptographic certificates is the most secure way of digital authentication. A cryptosystem of 128 bits security is, as specified by the U.S. National Institute of Standards and Technology (NIST), considered to be acceptable as of 2031 and beyond [9]. Some countries (such as Spain, Estonia, Luxembourg or Belgium), have a Public Key Infrastructure (PKI) for national identity cards. Such a key storage mechanism allows easy mobility of cryptographic keys to the user and requires only to memorize a password in order to use its cryptographic key, a key that (usually) never leaves the smartcard. Nevertheless, smartcards have disadvantages. While it may seem that it allows user mobility, it is still necessary to use it with a smartcard reader, limiting its usability. There is a high cost in the deployment of the card itself, which increases once we consider training the electorate or the Total Cost of Ownership (TCO). It is clear that the digitalisation of identification and authentication has to be included in governmental actions; however, it is not a straightforward project. The cost of deploying the Spanish electronic identity card was expected to be of 220 million Euro [10] and in 2014 (eight years after its introduction), it was used only in 0.02% of the electronic procedures [11]. In the case of Germany, apart from a delay in its deployment, it cost the government a total of 1.5 billion Euro [12] to deploy smartcards for eHealth. This shows that deploying a smartcard-based Public Key Infrastructure (PKI) in short notice cannot be a requirement for i-voting. Moreover, attacks to smartcard-based PKIs were shown [13], with two well known cases; first with the known vulnerability of Taiwan's smartcards [14], followed with the vulnerability of a chip distributor [15], which affected, among others, Estonia, a country considered as the pioneer of Europe's digitalisation.

Our goal is to present a protocol that can be used both by developing countries, and more technologically mature countries. We believe that it is important not to base our protocol on the requirement of the ownership of cryptographic key pairs by the users. Not only is it expensive, but as we have seen, several weaknesses have been found. Hence, if a requirement to deploy an online voting system is to first deploy individual cryptographic keys per user, several governments/companies/institutions might oppose.

Related work is presented, later, in Section 2, followed by our contributions. In Section 3, we provide the preliminaries before introducing the protocol; Section 3.1 introduces the parties of the protocol and the threat model. Section 3.2 introduces the cryptographic tools that we use to build our scheme. In Section 4, we describe NetVote and we follow with a description of the padding strategy in Section 5. A security analysis of the proposed construction is provided in Section 6. Section 7 presents a discussion on the assumptions made in the construction of the protocol, and we conclude our work and present possible lines of improvement in Section 8.

2. Related Work and Contributions

In this section we introduce the related work followed by our contributions.

2.1. Related Work

Coercion resistance comes at a high cost, as can be seen in the existing literature. In the present moment there are two methods to achieve coercion resistance in remote elections. First comes the use of fake credentials (or fake passwords) introduced in the work by Juels et al. [16] (referred to as the JCJ protocol), and used in several new constructions [17–22]. In these schemes, during the registration phase, the voter receives a bunch of credentials out of which some are correct and cast a valid vote, while others are invalid, and cast a non countable vote. When a voter is coerced, it uses a fake credential to cast a vote, making this vote invalid during the tallying phase. In a moment where the coercer is absent, the voter can cast a vote using the correct credential. These type of solutions assume the coercer to be absent during registration and at a given moment throughout the election and for the user to have access to an Annonymous Channel (AC). Moreover, they require the voter to privately and securely store cryptographic material (being able to hide it from the coercer) and to lie convincingly under the pressure of the coercer, which may indeed be a challenge for some. Finally, in order to provide coercion resistance, they do not provide feedback to the user of whether the vote was cast with a valid credential, resulting in a high dependence of the human memory and usage of the correct credentials at the right time.

Secondly, coercion can be mitigated by the use of re-voting. Voters can cast multiple votes, and the last vote is counted. Contrary to the previous approach, this solution requires the voter to be able to cast a vote after being coerced and before the election closes. However, there is no registration process where the coercer must be absent, the user may not necessarily need to store cryptographic material, and the voter can suffer from 'perfect coercion': the coercer may indicate exactly how the user must act, without the latter needing to lie about its actions while coercion is taking place.

We choose the latter solution as we believe that its core assumptions are more realistic for real-world scenarios. In Section 7 we give an intuition of why these assumptions are more realistic than the ones assumed in fake credentials based solutions. Re-voting has been used in several constructions proposed in the current literature. The main challenge that one finds when allowing multiple voting for coercion resistance is filtering. In order to mitigate coercion attacks, voting must be deniable, meaning that the adversary must not be able to determine whether a specific voter re-cast her ballot, or more generally, not be able to know which of the voters re-voted. Concurrently, auditability that the process happened as expected must be provided. We now present the state of the art of current filtering procedures.

The JCJ protocol [16] allows multiple voting, and the filtering stage is not deniable, i.e., one can determine whether a given vote has been filtered (note that this is not how JCJ achieves coercion resistance, but with fake credentials). They achieve this by using a cryptographic tool allowing an entity to compare two ciphertexts without the need of decrypting any of them; Plaintext Equivalence Texts (PETs). With this tool, they are capable of comparing every pair of credentials used to cast a vote. If two votes are cast by the same voter, they take the last. The complexity of comparing each pair of credentials results in a computation of $\mathcal{O}(N^2)$ PETs, with N being the total number of votes cast, making it an unusable scheme even for small scale elections. This complexity was later reduced by Araújo et al. [17], however, still not offering filter deniability. Rønne et al. [22] presents a JCJ-based linear filtering phase using fully homomorphic encryption. However, the type of computations needed to perform over the encrypted text, and the current state of the art of fully homomorphic encryption schemes, make some of the operations, as stated in their paper, "not satisfactory and remain the subject of ongoing research and future improvements".

While this work achieves similar asymptotic complexity as our work, their scheme serves as an initial research of a quantum safe scheme, and is not production ready. The dependencies in fully homomorphic encryption, and "not satisfactory" state of the zero knowledge proofs used, make it an unusable scheme.

The work presented by Spycher et al. [23] allows the voter to cast a vote in an electoral school, and therefore overwrite any previously cast votes. However, this results in the requirement of presence accessibility of the voter. Another used method is to do the filtering as a black box protocol. Trust is given to a server which will filter all votes and publish all re-encrypted votes [24]. This avoids any tracking or knowledge of which votes have been filtered, but the verifiability is totally dispensed. Dimitriu [25] proposes a solution based on trusted hardware capable of generating zero knowledge proofs. On top of that, it assumes that the coercer is not present at the time of casting, making it an unrealistic solution for our scenario.

To the best of our knowledge, current filtering schemes that offer a trustless deniable voting scheme which, at the same time have public verifiability, are the ones proposed by Achenbach et al. [26], Locher et al. [27], Locher et al. [28] and Lueks et al. [29]. The first three use similar solutions. In a protocol where a Public Bulletin Board (PBB) is used in order to allow verifiability, the filtering process must be done after the mixing, otherwise, a voter (and thus the coercer) would know whether their vote was filtered or not. However, after the mixing, it is not longer possible to know the order of the votes, and therefore, before inserting the votes in the mix-net there must be some kind of reference of their order. This solution faces this by performing Encrypted PETs (EPET) on the credentials for all votes against all lately cast votes before mixing. This consists of performing a PET, the output of which is encrypted, and if any of the comparisons among the credentials is equivalent, then the output of the EPET hides a random number (alternatively, the encrypted number is a one). Votes are then included in a mix-net. The filtering stage happens after mixing by decrypting the EPET, and all votes which output a random number are filtered out. This achieves deniability with no trust in any external entity. However, there is a high increase in the complexity as the EPETs have to be performed for each pair of votes, resulting in a complexity of $\mathcal{O}(N^2)$ distributed (among several servers) EPET calculations prior to the mixing, and in $\mathcal{O}(N)$ distributed decryptions and zero knowledge proofs of correct decryption during the filtering. Again, this makes these solutions unusable even for elections of hundreds of thousands of voters.

Note that all these filtering schemes presented above are implemented by comparing the voting credentials, which is necessary for voters to maintain a cryptographic state throughout the election.

The recent work presented by Lueks et al. [29] presents a similar protocol to that presented in this work. In their scheme, the trust in the tallying server reduces the existing complexity of the tallying phase from quadratic to polynomial-logarithmic. The use a deterministic padding strategy makes their solution not depend on probability distributions. Our scheme, on the other hand, achieves a linear filtering phase by depending on a probability distribution, making an attack of vote buying more likely than negligible. Note, however, that if a user wants to escape coercion, an adversary can only detect it with negligible probability. In Table 1 we present a comparison of the existing schemes with respect to NetVote.

Everlasting privacy is a concept that was introduced by Moran et al. [30]. In this scheme, perfectly hiding commitment schemes are used to hide vote intention, and the hiding values of the commitment schemes are exchanged through private channels. This idea can be used in any scheme based in homomorphic tallying, and by consequence in ours. In [21], an everlasting privacy scheme is presented in the JCJ setting, giving to users the burden of having to handle several credentials, but with the improvement on previous schemes that it only assumes the existence of private channels. Locher et al. [27] present an everlasting privacy scheme in an information-theoretical sense, with the main drawback of having a quadratic proposal, which makes it unusable even for medium sized elections.

Table 1. Comparison of existing schemes. NetVote constitutes the first re-voting scheme with linear complexity, which is verifiable and does not depend on immature cryptography (IC). We introduce the new notion of strict coercion resistance to achieve linear filtering.

	Filtering				**Coercion**		
	Deniable	**Verifiable**	**Complexity**	**Crypto State**	**Assumption**	**Strict**	**IC**
JCJ [16,18,20]	No	Yes	n^2	yes	k-out-of-t + AC	No	No
Rønne et al. [22]	No	Yes	n	yes	k-out-of-t + AC	No	Yes
Blackbox [24]	TTP	No	n	Yes	TTP	No	No
Achenbach et al. [26]	k-out-of-t	Yes	n^2	Yes	k-out-of-t + AC	No	No
Locher et al. [27]	k-out-of-t	Yes	n^2	Yes	k-out-of-t + AC	No	No
Lueks et al. [29]	TTP	Yes	$n \log n$	No	TTP	No	No
NetVote	TTP	Yes	n	No	TTP	Yes	No

2.2. Our Contribution

In this paper we present an i-voting scheme, for which the trade-offs made converts it in an interesting choice to be used as a remote voting scheme for large scale elections. Our construction is based on a creation of ephemeral anonymous certificates making the voting procedure private even against future adversaries. We mitigate coercion by allowing re-voting. Our construction not only is deniable and verifiable, but it presents a method with reduced complexity in comparison with existing proposals [26–29]. We base our construction in well known and studied cryptographic protocols. We present a novel probabilistic dummy voter generation procedure, which reduces the filtering complexity of the solution to linear in the number of cast votes, making it suitable for large scale elections. Moreover, our scheme does not rely in the user keeping a cryptographic state, allowing the later to vote from any device.

In this paper we introduce a game based definition for practical everlasting privacy. Moreover, we introduce the notion of strict coercion resistance, which models only the attack of coercion, and separates the act of vote buying from the property. Current definitions model both the attacks of coercion and vote buying under the same property, namely coercion resistance. Our model satisfies ballot privacy, verifiability, practical everlasting privacy and strict coercion resistance without losing usability. We formally define these properties and prove that our model satisfies them.

This text is the full work of our shorter version published at CISIS19 [31]. Several results only appear in this long version. Namely: security proofs, formal definitions, details on the protocol and an analysis of the probabilistic dummy vote addition technique.

3. Parties and Building Blocks

This section introduces the parties involved and the building blocks that we use for the construction of the voting protocol. Building blocks are referenced to their original construction and used 'as is' during the rest of the paper.

3.1. Parties and Threat Model

The parties involved throughout the protocol are the electorate, formed by n_e voters, of which we only consider the subset that takes part in the election, say n voters, with $n \leq n_e$ identified as $V = \{V_1, V_2, \ldots, V_n\}$. The t trustees $\mathrm{T} = \{\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_t\}$, each holding a share, sk_{T_i}, of the voting key, pk_{T}. Certificate authority, CA, which handles the generation of voting credentials. The voting server, VS, which validates and posts the votes and the tallying server, TS, which performs the filtering and the counting of votes. In general, given an entity X, pk_X and sk_X will denote the public and private keys of X, respectively.

We informally introduce the security properties that our protocol satisfies. A more formal definition appears in Section 6.

Ballot privacy guarantees that an adversary cannot determine more information on the vote of a specific voter than the one given by the final results. In our model, ballot secrecy is achieved assuming a subset of the tellers is honest.
Practical everlasting privacy assures that ballot secrecy will be maintained with no limit in time. This is, even considering a computationally unbounded adversary, ballot secrecy is not broken. Assuming the certification authority follows the protocol honestly, our construction satisfies, in addition, practical everlasting privacy.
Verifiability allows any third party to verify that the last ballot per voter is tallied, the adversary cannot include more votes than the number of voters it controls, and that honest ballots cannot be replaced. Assuming that CA is honest, the protocol provides verifiability. This assumption is required to grant voters voting certificates. The existence of a trusted entity creating and assigning authentication credentials to eligible voters is intrinsic in the remote voting scenario.
Strict coercion resistance allows voters to escape coercion without the coercer noticing. This is, if the voter escapes coercion resistance, the adversary is incapable of detecting it. NetVote satisfies strict coercion given that the tallying server is honest. Note that this assumption is not required for verifiability. However, to reduce the filtering complexity from quadratic to linear, we let TS learn which are the valid votes.

3.2. Building Blocks

Let ϵ be a security parameter. Let $\mathbb{G}$ be a cyclic group of prime order p generated by generator g and let $\mathbb{Z}_p$ be the set of integers modulo p. We write $a \leftarrow_\$ \mathbb{Z}_p$ to denote that a is chosen uniformly at random from the set $\mathbb{Z}_p$.

NetVote uses the ElGamal's encryption scheme given by: the key generation algorithm $\mathsf{KeyGen}(\mathbb{G}, g, p)$ which generates the key-pair $(pk = g^{sk}, sk)$ for $sk \leftarrow_\$ \mathbb{Z}_p$; the encryption function $\mathsf{Enc}(pk, m)$ which given a public key pk and a message $m \in \mathbb{G}$ returns a ciphertext $c = (c_1, c_2) = (g^r, m \cdot pk^r)$ for $r \leftarrow_\$ \mathbb{Z}_p$; and the decryption algorithm $\mathsf{Dec}(sk, c)$ which given a secret key, sk, related to the public key used to encrypt the ciphertext, $pk = g^{sk}$, returns the message $m = c_2 \cdot c_1^{-sk}$. NetVote leverages the additive homomorphic property that this scheme offers. Let $l \in \mathbb{Z}_p$ be a value that one wants to encrypt. To leverage the homomorphic property, we encode it as a group element $m = g^l$. Then, given a key-pair $(pk, sk) = \mathsf{KeyGen}(\mathbb{G}, g, p)$ and two messages, $m_1 = g^{l_1}, m_2 = g^{l_2} \in \mathbb{G}$, applying the binary operation that defines the group, $\cdot$, over the corresponding encryptions results in the encryption of the added messages. More precisely,

$$\begin{aligned}\mathsf{Enc}(pk, m_1) \cdot \mathsf{Enc}(pk, m_2) = (g^{r_1}, m_1 \cdot pk^{r_1}) \cdot (g^{r_2}, m_2 \cdot pk^{r_2}) = \\ (g^{r_1+r_2}, g^{l_1+l_2} \cdot pk^{r_1+r_2}) = \mathsf{Enc}(pk, m_1 \cdot m_2) = \mathsf{Enc}(pk, m_{12})\end{aligned}$$

with $r_1, r_2 \leftarrow_\$ \mathbb{Z}_p$ and $m_{12} = g^{l_1+l_2}$ the encoding of the addition of the values l_1, l_2. We use this to perform a homomorphic tally of the votes without requiring to decrypt each individual ballot, but only the result.

We leverage the homomorphic property to randomise ciphertexts, by adding a ciphertext with an encryption of zero. We denote the randomisation of a ciphertext c with randomness r by $(\Pi_R, c') = \mathsf{Rand}(c, r)$.

To distribute the trust among the different tellers, we use threshold ElGamal encryption. For this, the trustees jointly run the $\mathsf{VoteKeyGen}(1^\epsilon, k, t, n_C)$ protocol where ϵ is the security parameter, k is the number of trustees needed to decrypt ciphertexts, t the total number of trustees, and n_C the number of candidates. They follow the protocol presented in [32]. Such a protocol allows a set of trustees to compute a public key pair where the public key is directly computed from the different shares of the private key, meaning that the 'full' private key is never computed. This protocol outputs a public encryption key pk_T and each trustee T_i obtains a private decryption key sk_{T_i}. To encrypt her vote for candidate C, a voter calls $(\mathcal{V}, \Pi) = \mathsf{VoteEnc}(pk_\mathrm{T}, C)$ to obtain an encrypted vote $\mathcal{V}$ and proof Π that $\mathcal{V}$ encrypts a choice for a valid candidate.

We say that a probability P is negligible with respect to ϵ if for every positive integer c there exists an integer N_c such that for all $x > N_c$, $P < 1/x^c$.

We denote the encryption of the zero candidate (i.e., no candidate) with explicit randomiser $r \leftarrow_\$ \mathbb{Z}_p$ by $\mathsf{VoteZEnc}(pk_\mathrm{T}; r)$.

The algorithm $\mathsf{VoteVerify}(pk_\mathrm{T}, \mathcal{V}, \Pi)$ outputs $\top$ if the encrypted vote $\mathcal{V}$ is correct, and $\perp$ otherwise. To decrypt a ciphertext, c, the trustees jointly run the $(z, \Pi_z) \leftarrow \mathsf{VoteDec}(pk_\mathrm{T}, c)$ protocol to compute the election result z and a proof of correctness Π_z.

NetVote uses deterministic encryption (with randomness zero) as a cheap verifiable 'encoding' for the dummy ballots. This allows the TS to include dummy ballots in a cheap verifiable way.

We use a traditional signature scheme given by: the signing algorithm $s = \mathsf{Sign}(sk, m)$ that signs messages $m \in \{0,1\}^*$; and a verification algorithm $\mathsf{SignVerify}(pk, s, m)$ that outputs $\top$ if s is a valid signature on m and $\perp$ otherwise.

We use verifiable re-encryption shuffles [33,34] to support coercion resistance in a privacy preserving and verifiable way. These enable an entity to verifiably shuffle a list of homomorphic ciphertexts in such a way that it is infeasible for a computationally bounded adversary to match input and output ciphertexts. These are defined by a function $\mathsf{Shuffle}(A) = (\Pi_s, A')$, which inputs a list of ciphertexts, A, outputs a shuffled list of ciphertexts, A', and a proof of shuffle, Π_s.

NetVote uses standard zero-knowledge proofs [35] to prove correct behaviour of the different parties. We use the Fiat–Shamir heuristic [36] to convert them into non-interactive proofs of knowledge. We adopt the Camenisch–Stadler notation [37] to denote such proofs and write, for example,

$$SPK\{(sk) : pk = g^{sk} \wedge m = \mathsf{Dec}(sk, c)\},$$

to denote the non-interactive proof of knowledge that the prover knows the private key sk corresponding to pk and that c decrypts to m under sk.

In particular, the TS, to prove correctness of filtering, it needs to prove that an encrypted counter, $\mathsf{Enc}(pk, \mathsf{Counter}_1)$ is greater than another encrypted counter, $\mathsf{Enc}(pk, \mathsf{Counter}_2)$, without disclosing any information about the counters. For this it uses the homomorphic property of the encryption scheme to subtract both encryptions:

$$\mathsf{Counter}_S = \mathsf{Enc}(pk, \mathsf{Counter}_1)/\mathsf{Enc}(pk, \mathsf{Counter}_2) = \mathsf{Enc}(pk, \mathsf{Counter}_1 - \mathsf{Counter}_2).$$

If $\mathsf{Counter}_1$ is greater than $\mathsf{Counter}_2$, then $\mathsf{Counter}_S$ will be greater than zero. However, as we are working over finite fields, even if $\mathsf{Counter}_2$ is greater, the subtraction will be positive. Given that the counters will be numbers of at most 32 bits, then we know that if $\mathsf{Counter}_1$ is greater, then $\mathsf{Counter}_S$ will also be a number of at most 32 bits. In the opposite scenario, $\mathsf{Counter}_S$ will be a much bigger number. It suffices then to prove that $\mathsf{Counter}_S$ is a number of at most 32 bits. For this, we use the range proof presented by Bünz et al. [38]. To denote the proof that a number is greater than another, we use Π_{GT}.

NetVote uses anonymous credentials during the registration phase and vote cast. The only requirement of these credentials is that they certify certain attributes, which are used to group voters by electoral colleges and filter votes cast by the same voter. Several constructions exist in literature, [39–41]. We instantiate them by the use of three algorithms: The request, $\mathsf{ReqCred}(\mathsf{auth})$, where the user authenticates to the credential authority and requests an anonymous credential; the generation, $\mathsf{Cert}(\{\mathsf{attr}\}_{i=1}^{n}) = \mathsf{CredGen}(\{\mathsf{attr}\}_{i=1}^{n})$, where upon receipt of a certificate request, the certificate authority generates one with the attributes, attr, assigned to the user; and the verification of the certification, $\mathsf{CertVerify}(pk_{\mathrm{CA}}, \mathsf{Cert}(\{\mathsf{attr}\}_{i=1}^{n}))$, where with input of a certificate and the public key of the certificate authority, verifies the correctness of the certificate. It outputs $\top$ if the verification succeeds, and $\perp$ otherwise. While any type of attributes can be added in these certificates, throughout the paper we only consider a unique anonymous identifier per voter, and leave additional attributes optional to electoral runs.

Finally, we use an append-only PBB where votes, proofs (re-encryption, shuffle, decryption) and all intermediate steps until the tally step are posted. A specification of a possible construction is explained in [42]. Another interesting approach for a PBB is the use of blockchain technologies, as presented in [43].

4. Description of Our Protocol

This section presents an overview of the protocol in order to introduce the basic ideas of our construction, followed with a more exhaustive explanation of each of the blocks.

4.1. Overview

The protocol can be divided in three phases: pre-election, election and tallying phases. During the pre-election phase all server keys are generated. VS randomly generates n_e voter identifiers, VId_i for $1 \leq i \leq n_e$. It then commits to them by encrypting them, $w_i = \mathsf{Enc}(pk_{\mathrm{VS}}, VId_i)$, and posting them in the PBB. In the election-phase the voters first obtain an anonymous certificate, which proves that they are eligible voters. This certificate is ephemeral, and is generated each time that a voter wants to cast a vote. Each certificate generated for the same voter contains the same w_i in order to uniquely identify each vote cast by the same voter. When the voter wants to cast a vote, they sign the encrypted ballot with the ephemeral certified key to prove, on the one hand, that they are an eligible voter, and on the other hand, to link the vote to the w_i. They then send their vote to VS, which verifies the correctness of the latter and publishes the encrypted ballot in PBB. The voter verifies that her vote was recorded as cast. These two steps are presented in Figure 1. During the tallying phase, the VS generates a random number of 'dummy' certificates for each of the voters and casts 'dummy' votes (in order to hide the number of votes cast by each voter) in a verifiable manner. Next, the votes are filtered by VS in a verifiable and private manner. Finally, the tellers, T, proceed with a complete tally and decryption of the votes. This phase is presented in Figure 2.

4.2. Pre-Election Phase

During the pre-election phase, the different parties generate their key pairs and publish the public key in PBB. Similarly, PBB publishes the list of candidates. Then, VS generates the random identifiers and commits to them by publishing their encryption.

Procedure 1 (Setup)**.** During the setup procedure, the different entities run $\mathsf{Setup}(\epsilon, \mathcal{C}, k, t)$. They pick a group $\mathbb{G}$ with primer order p and generator g. Then they proceed as follows:

1. PBB publishes the list of candidates, $\mathcal{C} = \{C_1, \ldots, C_{n_C}\}$, and initialises a counter $\mathsf{Counter} = 0$.
2. CA, VS and TS generate their key-pairs $(pk_{\mathrm{CA}}, sk_{\mathrm{CA}}), (pk_{\mathrm{VS}}, sk_{\mathrm{VS}})$ and $(pk_{\mathrm{TS}}, sk_{\mathrm{TS}})$ respectively, by calling $\mathsf{KeyGen}(\mathbb{G}, g, p)$. They proceed by publishing their public keys in PBB.

3. The trustees distributively run $\mathsf{VoteKeyGen}(1^\epsilon, k, t)$, where the voting key, pk_{T}, is generated, and each trustee owns a share of the private key sk_{T_i}. They proceed by publishing the voting key, pk_{T} in PBB.
4. CA takes as input the total number of voters, n_e, and generates random and distinct voting identifiers, $VId_i \leftarrow_{\$} \mathbb{Z}_p$ for $1 \leq i \leq n_e$ with $VId_i \neq VId_j$ for $i \neq j$. It keeps the relation between VId and voter private. Next, it encrypts all identifiers:

$$w_i := \mathsf{Enc}(pk_{\mathrm{TS}}, VId_i) \text{ for } 1 \leq i \leq n_e, \text{ with } r_i \leftarrow_{\$} \mathbb{Z}_p.$$

Next, it signs each encrypted identifier, w_i:

$$\sigma_i = \mathsf{Sign}(sk_{\mathrm{CA}}, w_i).$$

Finally, it commits to these values by publishing the pair (w_i, σ_i) in PBB.

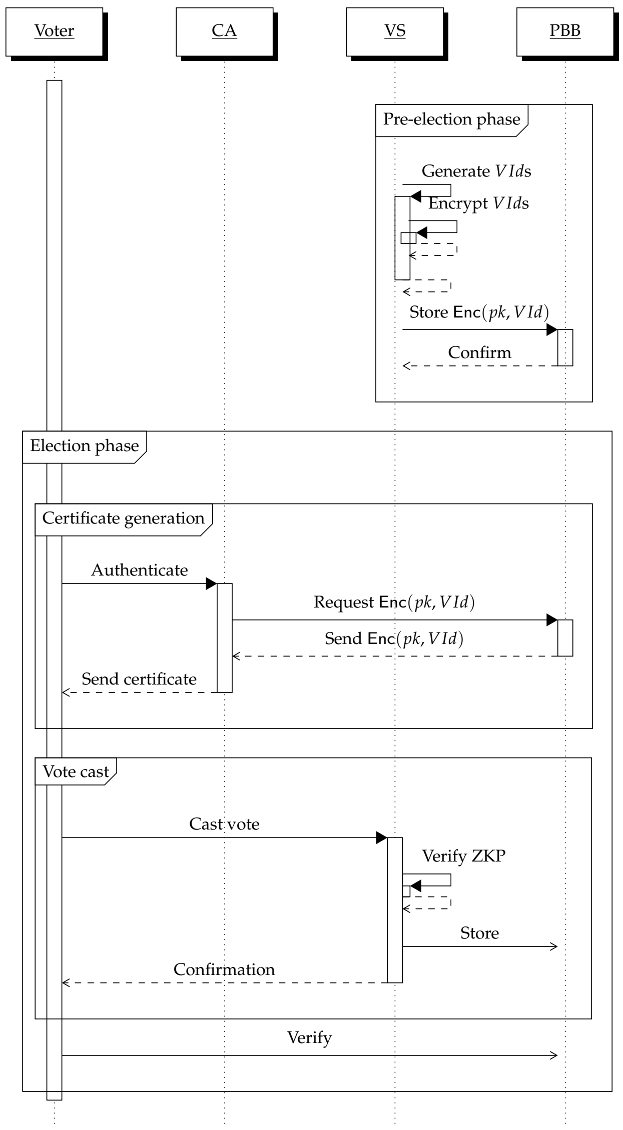

Figure 1. Diagram presenting a summary of the pre-election and election phases involving the voters, certificate authority, voting server and public bulletin board.

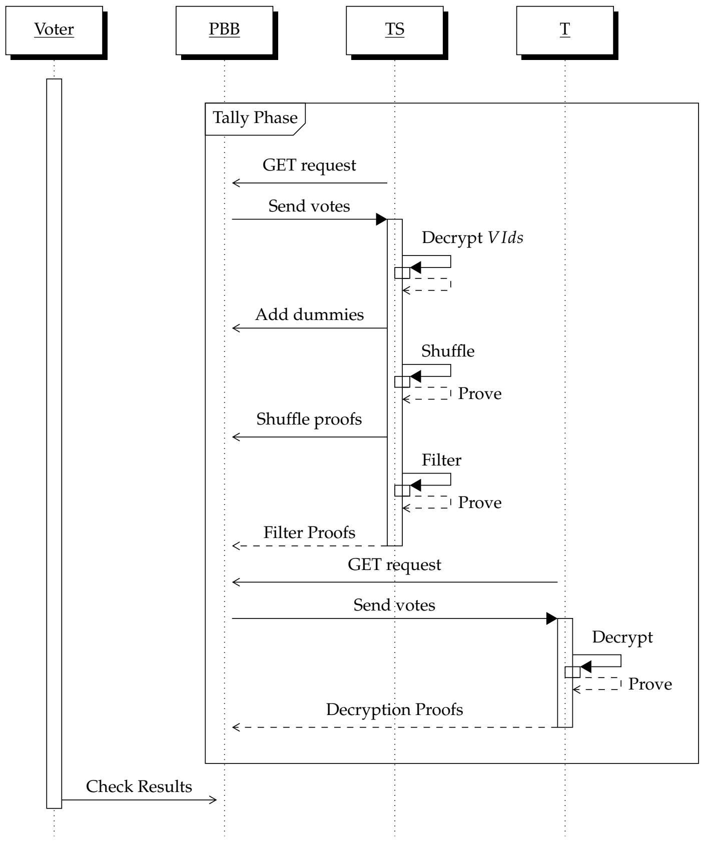

Figure 2. Diagram showing a summary of the tallying phase, involving the voters, the public bulletin board, the tallying server and the tellers.

4.3. Election Phase

This phase comprises all steps that are taken while the election process is open. Note that a voter needs to follow a certification phase for each vote cast, which allows to avoid coercion without a high increase in complexity whilst simplifying the task for voters to cast votes multiple times from different devices.

Procedure 2 (Certificate generation)**.** The voter authenticates to the certification authority using their inalienable means of authentication, $auth$, and requests an anonymous credential. As a response, they receive a one-time use anonymous certificate with a re-encryption, w_i', of the respective w_i as an attribute. Together with the certificate, $\mathsf{Cert}(w_i')$, CA includes a proof of correct re-encryption of w_i.

1. The voter authenticates to CA and requests an anonymous certificate generation $\mathsf{ReqCred}(auth)$.
2. CA selects the corresponding encrypted voter identifier, w_i, an randomises the ciphertext by levering the homomorphic property:

$$(\Pi_R, w_i') = \mathsf{Rand}(w_i, r_i),$$

for $r_i \leftarrow_\$ \mathbb{Z}_p$, to generate a randomisation of the encrypted identifier, w_i', and a proof of correct randomisation, Π_R.
3. CA generates an anonymous certificate, $\mathsf{Cert}(w_i') = \mathsf{CredGen}(\{w_i'\})$, with w_i' as an attribute, and sends it to the user together with w_i and the proof of re-encryption Π_R,

$$\left(\mathsf{Cert}(w_i'), w_i, \Pi_R\right).$$

4. The user verifies that the attribute of the certificate is an encryption of w_i by verifying Π_R. If it is not the first time it casts a vote, it may also verify that it has received the same w_i during both vote cast phases.

Procedure 3 (Vote cast)**.** The procedure of casting a vote proceeds as a 'regular vote-cast protocol'. Voter selects a candidate, encrypts it and proves correctness. It includes her encrypted identifier for a further filtering phase. More precisely,

1. The voter encrypts the chosen candidate, $C \in \mathcal{C}$ and generates a proof of correctness by calling
$$(\mathcal{V}, \Pi) = \mathsf{VoteEnc}(pk_{\mathrm{T}}, C)$$
and signs it using their ephemeral certificate, Cert. Let sk_C be the private key related to Cert, then the user computes
$$\sigma = \mathsf{Sign}(sk_C, \mathcal{V}),$$
Finally, they send it to VS,
$$\left(\mathsf{Cert}(w_i'), \mathcal{V}, \Pi, \sigma\right).$$
2. VS verifies that the certificate was issued by the CA, the signature is a valid signature by Cert, and the proofs ensures that the encrypted vote corresponds to a valid candidate:
$$\mathsf{CertVerify}(pk_{\mathrm{CA}}, \mathsf{Cert}(w_i')) = \top \wedge \mathsf{SignVerify}(pk_C, \sigma) = \top \wedge \mathsf{VoteVerify}() = \top.$$
If everything verifies correctly, it sends the vote to PBB and sends an acknowledgement to the voter.
3. PBB augments the counter $\mathsf{Counter} = \mathsf{Counter} + 1$ and publishes the vote in the board
$$\left(\mathsf{Counter}, \mathsf{Cert}(w_i'), \mathcal{V}, \Pi, \sigma\right).$$

Procedure 4 (Vote verification)**.** The voter, upon receiving the acknowledge, can verify that the vote was recorded as cast by viewing the PBB. Moreover, any third party is able to check that all votes recorded in the PBB come from a certified voter, the w_i' is related to the certificate and votes have a correct format.

Note that voters may repeat Procedures 2 and 3 as many times as they wish while the election is open, from different devices without needing to store or move the credentials.

4.4. Tallying Phase

At this point, the election process is closed, and all votes from the voters that took place in the election (possibly more than one vote from some voters) are stored. Before proceeding to the tallying, the system needs to perform the filtering, i.e., keeping the last vote of each voter. To this end, we make use of a proof determining whether $a > b$, with a, b being the counters of the objects in the PBB, without giving any other information of a, b.

Procedure 5 (Public filtering)**.** Our interest is to do the public filtering in a manner that a voter (or coercer) cannot know which of their votes has been accepted. However, it must be provable that only the last vote cast by each individual voter passed the filtering. In order to do this, we use Bulletproofs [38] to prove that an encrypted counter is greater than another.

1. TS begins stripping the information from the ballot, keeping only the necessary information to proceed the filtering phase, mainly the counter, the encrypted identifier and the encrypted vote, $(\mathsf{Counter}, w_i', \mathcal{V})$ and adds them to the PBB.
2. Next, the TS decrypts the voter identifier and ads a random number of dummy voters per voter. We describe how the TS chooses the number of voters to add per voter in Section 5. We want

the counter to be always less than the counters of honest voters, and hence, TS always adds a dummy with a zero counter. Moreover, every added dummy will have the zero vector and zero randomness, $\mathsf{VoteZEnc}(pk_{\mathrm{T}}, 0)$,

$$(0, w_i'', \mathsf{VoteZEnc}(pk_{\mathrm{T}}, 0)).$$

where w_i'' is another randomisation of w_i distinct to w_i'. This allows anyone to verify the correctness of this process, while maintaining private to which voter each dummy vote is assigned.

3. Next, TS encrypts each of the counters with randomness zero for verification purposes $\mathsf{EncCounter} = \mathsf{Enc}(pk_{\mathrm{TS}}, i)$ for every counter i, resulting in each entry of PBB as follows:

$$(\mathsf{EncCounter}_i, \mathcal{V}_i, w_i').$$

4. Now, TS proceeds by performing a verifiable shuffle to all the entries in the bulletin board. Let $B = [B_1, \ldots, B_L]$ be the stripped ballots with the encrypted counter. It uses a provable shuffle to obtain a shuffled list of ballots, $B' = [B_1', \ldots, B_L']$ and a proof of correctness Π_s. It appends B' and Π_s to PBB.
5. Next, it proceeds to group encrypted votes cast by the same voters. To achieve this, it proceeds by decrypting each identifier, $VId_i = \mathsf{Dec}(sk_{\mathrm{TS}}, w_i')$ and adding a proof of correct decryption, Π_d,

$$(\mathsf{EncCounter}_i, \mathcal{V}_i, VId_i, \Pi_d).$$

6. Finally, TS filters the votes by taking the encrypted vote with the higher counter. To do this, it locally decrypts every counter and selects the one with the highest counter. It proceeds by publishing the filtered votes, and together with S_i^G proves that the counter is greater than all the other counters related to votes cast by the same voter, where S_i^G is the number of votes cast by voter V_i. It publishes it in the bulletin board

$$(\mathsf{EncCounter}_i, \mathcal{V}_i, VId_i, (\Pi_{GT,j})_{j \in S_i}),$$

where S_i are the groups formed by votes cast with the same identifier VId_i.

Procedure 6 (Tallying)**.** Finally, the TS calculates the full encrypted result by performing an addition of all ciphertexts accepted after the filtering

$$\mathsf{Enc}(pk_{\mathrm{T}}, v_{\mathtt{Final}}) = \prod_{i=1}^{n} \mathsf{Enc}(pk_{\mathrm{T}}, v_i) = \mathsf{Enc}\left(pk_{\mathrm{T}}, \sum_{i=1}^{n} v_i\right).$$

Then, the result goes through the group of trustees T holding the different shares of the private key. They decrypt and generate the proof of correct decryption, Π_z. Moreover, any auditor or third party can calculate the sum of all the ciphertexts, and then verify that the result is a proper decryption of such a product.

5. Including Dummy Votes

The inclusion of dummy votes allows us to mitigate the 1009 attack in an elegant and simple manner. Undoubtedly, complexity increases, as we will be including more votes in the shuffle and the filtering stage. However, the number of dummy votes does not need to be big, and therefore complexity will only be increased by a small constant. In this section, we describe how the TS includes dummy votes once the election is closed.

This strategy is designed to mitigate what is known as the '1009 attack' [44]. In this attack, the coercer tells the user to cast an unusual number of votes (e.g., 1009). Then, during the filtering phase, the coercer looks at the public information posted in the PBB, and looks for a voter that has cast 1009 votes. If there is such a group, then the coercer knows that the voter submitted to coercion. On the other hand, if there is no such group, then the coercer knows that the voter escaped coercion.

A naive way to solve this problem would be to filter the votes in a black-box manner [24]. However, such a scheme provides no verifiability.

In order to hide the number of votes that each voter cast, we include a random number of votes per eligible voter. In this section we show how we hide voting patterns by individual voters by using dummy ballots. To see how it is not trivial to determine the number of dummies to add per voter, consider the following:

- It is straightforward to see why a fixed number of dummies for all voters would not serve the purpose here. So say that a random number of dummies for voter V_i, $n_{i,dum}$, is added. If the set where we take n_{dum} from is $\mathbb{Z}$, then the overhead would become prohibitive, as there is no upperbound. However, if $n_{dum} \leftarrow_{\$} \mathbb{Z}_u$, where u is the upperbound, then with probability $1/u$, $n_{i,dum} = u$ which would blow the wistle if voter V_i decided to re-vote to escape coercion (as now a total of $u + 1$ votes would have been added, where not all could have been dummies).

Then, it seems necessary to have no upperbound, but instead of choosing $n_{i,dum}$ uniformly at random from $\mathbb{Z}_u$, we could use a distribution where $n_{i,dum} = n$ with probability $1/2^{n+1}$. With this distribution there is no uppebound but it is very unlikely to add a big number of dummies for voter V_i. The drawback of this mechanism is that there is a lower bound. Moreover, with probability $1/2$, zero dummies will be added. This is not a problem for a voter who wants to escape coercion, as re-voting would not reveal anything to the coercer. However, a voter that wants to submit would be able to prove so with probability $1/2$, and therefore be able to sell its vote with high probability.

- However, while we want to hide the groups of votes with unusual group sizes (e.g., 1009), we do not need to add an overhead to small groups (which are not prone to receive the 1009 attack). To this end, our solution adds random votes to voters depending on the votes they have cast, following the negative binomial distribution, defined as:

$$f(\mu; \rho, p) := \Pr[X = \mu] = \binom{\mu + \rho - 1}{\mu} (1 - p)^{\rho} p^{\mu}.$$

where ρ is the number of votes cast by the voter in question, and μ is the number of dummy votes to add for that user.

The choice of this probability distribution is made clear in Figure 3, left. This distribution results in adding, with high probability, a low number of votes for voters that cast a small number of votes, and with very low probability adding zero dummies for votes who cast an unusual number of votes. This behaviour is exactly what we want to achieve for hiding voting patterns.

We present in Figure 3, left, the different distributions of the number of dummy votes added depending on how many votes a voter cast. In both top and bottom (left) we see the probability distribution of the number of dummies to add given a number of cast votes. In the top, the probability of success is 0.8, while on the bottom it is of 0.2.

In order to understand how big the overhead is in the filtering phase, we present the overall overhead assuming that a subset of users re-voted.

To define how the population of voters votes, we use the Poisson distribution to determine the number of votes cast by each voter. This distribution is , defined by

$$P(k; \lambda) := \Pr[X = k] = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Again, this distribution fits our goal with $\lambda = 1$, as we expect most users to vote one or two times, and consider the activity of voting a lot improbable. Note that, with $\lambda = 1$, the probability of casting a high number of votes is low. However, we need to consider a subset of voters that casts lots of votes, either as an attack to the system or simply by users who are suffering coercion. To this end, we consider that a given percentage, say l, cast a very high number of votes, and therefore the number of re-votes are chosen at random from the Poisson distribution with $\lambda = 200$. In this case, the probability of casting a lot of votes is high. In Figure 3 right, we plot the overhead caused by dummies ((number of votes + number of dummies)/number of votes) with a varying l between 0.1–5% of such users. Top and bottom represent the overhead using probability of success of 0.8 and 0.2, respectively.

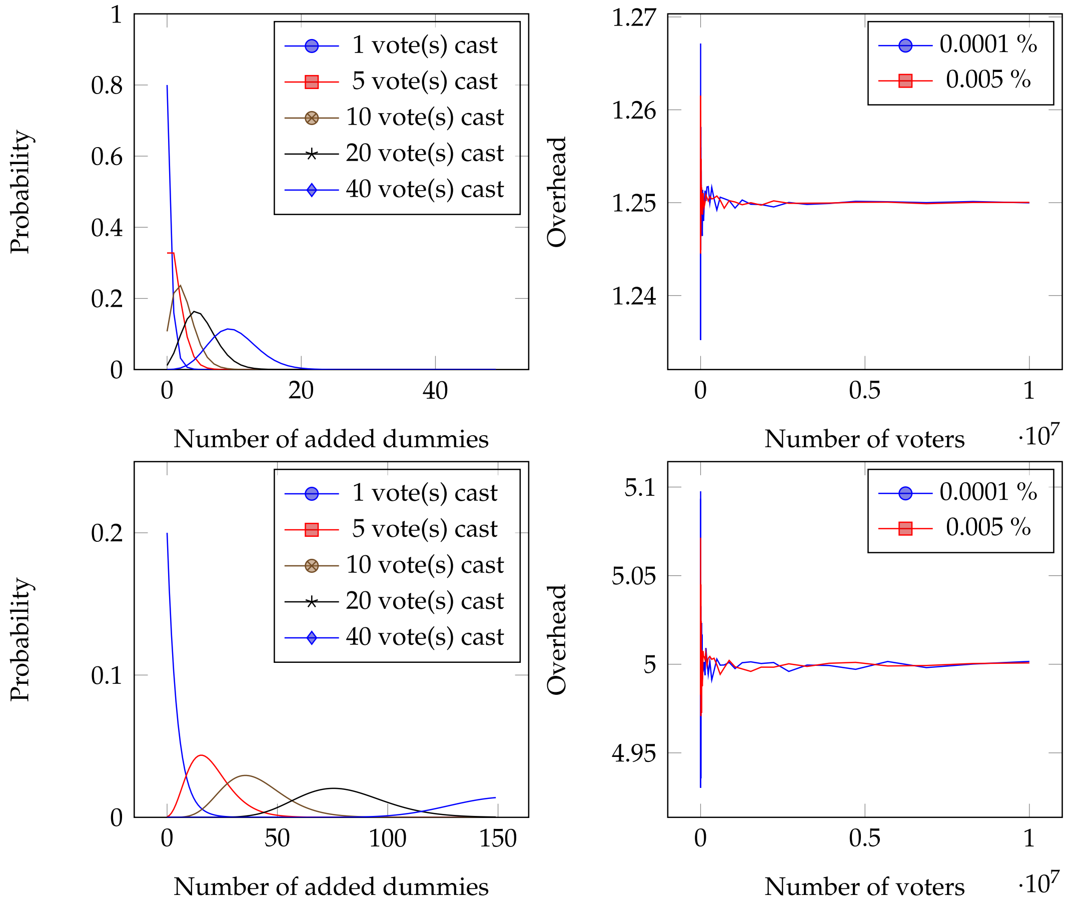

Figure 3. Negative binomial distribution (**left**) with probability of success 0.8 (**top**) and 0.2 (**bottom**). This distribution is used to select the number of dummies to add per voter, depending on how many votes a voter cast. Overhead caused by the dummy addition (**right**), having used the corresponding distributions (**top** and **bottom**), where the overhead is counted as $((\#votes + \#dummies)/\#votes)$.

We can see that the increase of the number of votes to process is at most times five. Compared with the state of the art scheme by Lueks et al. [29] with a complexity of $O(n \log n)$, our scheme benefits a considerable increase complexity with $O(n)$ where the factor is five for the example case presented above. This linear increase can easily be countered with an increase in the machines used for the filtering stage.

6. Security Analysis

In order to analyse the security properties of the scheme, we define them using a general, protocol independent, syntax. In this section we begin introducing the used syntax and we proceed with the formal definitions of ballot privacy, practical everlasting privacy, verifiability and strict coercion resistance. Next, we prove that our scheme, as defined in Section 4, satisfies these generically defined properties.

A voting scheme consists of seven protocols: Setup, Register, CastVote, VoteVerify, Valid, Tally and Verify:

- $\mathsf{Setup}(\mathcal{E}, \mathcal{C})$. In the pre-election phase, the scheme runs Setup to prepare the voting scheme for voting. This protocol takes as input the electoral roll $\mathcal{E}$, the list of all eligible voters, and the list of candidates $\mathcal{C}$.
- $\mathsf{Register}(i)$. Before casting a vote, voter V_i run $\mathsf{Register}(i)$ to obtain a token τ that allows them to cast a vote.
- $\mathsf{CastVote}(\tau, C)$. After registering, voters use their token τ and selected candidate C to cast their vote using the $\mathsf{CastVote}(\tau, C)$ protocol. The user produces a ballot β containing their choice, and interacts with the voting scheme. If the user's ballot β is accepted by the voting scheme, the ballot β is added to the public bulletin board.
- $\mathsf{VoteVerify}(\beta, PBB)$. After casting a vote, voters can verify that the ballot was recorded as cast, i.e., they verify that the vote was successfully stored in PBB.
- $\mathsf{Valid}(\beta, PBB)$. Once the scheme receives a vote, it verifies that it is valid.
- $\mathsf{Tally}(\mathcal{B})$. After the voting phase, the voting scheme runs Tally. This protocol takes as input the set $\mathcal{B}$ of ballots on the bulletin board. It filters the votes and outputs an election result, z, and a proof of correctness Π of the election results.
- $\mathsf{Verify}(\mathcal{B}, z, \Pi)$. Finally, any third party can run Verify. This protocol takes the set $\mathcal{B}$ of ballots, the result, z, the proof of correct tally, Π, and checks its correctness.

6.1. Ballot Privacy

The following game between an adversary, $\mathcal{A}$, and a challenger, $\mathcal{D}$, based on Bernhard et al. [45], formalises ballot privacy. $\mathcal{A}$ wins the game if it manages to differentiate between a real and fake world. To model these two worlds, the game, $\mathtt{Exp}^{\mathtt{bpriv},}_{\mathcal{A},\mathcal{V}}$, tracks two bulletin boards, PBB_0 and PBB_1 for each world respectively. To model ballot privacy, we allow the adversary to control the certificate authority, CA, the voting server, VS, and the tallying server, TS. During the game, the adversary can make calls to the following oracles:

- $\mathcal{O}\mathtt{board}()$ which allows the adversary to see the information posted until that moment in the bulletin board. It can call this oracle at any point of the game.
- $\mathcal{O}\mathtt{LRvote}(i, C_0, C_1)$ where $\mathcal{A}$ selects two possible candidates, C_0, C_1 for voter V_i. The challenger produces voting tokens τ and generates one ballot for each candidate, β_0, β_1. It then places β_0 and β_1 in PBB_0 and PBB_1 respectively. It can call this oracle at any point of the game.
- $\mathcal{O}\mathtt{cast}(\beta)$ where $\mathcal{A}$ has the ability to cast a vote for any voter. The same ballot, β, is generated for both bulletin boards. It can call this oracle at any point of the game.
- $\mathcal{O}\mathtt{tally}()$, which allows $\mathcal{A}$ to request the result of the election. To avoid information leakage of the tally result, the result is always counted on PBB_0, so in the experiment 1, the results and proofs are simulated. It can call this oracle once, and after receiving the answer, $\mathcal{A}$ must output a guess of the bit (representing the world the game is happening in).

We denote the calls to the oracles by $\mathcal{A}^{\mathcal{O}}$. At the end of the game, the adversary needs to output b', guessing which of the two worlds (real or fake) it is seeing. If it guesses correctly with non negligible probability, the adversary has won the game. We formally describe the experiment, $\mathtt{Exp}^{\mathtt{bpriv},}_{\mathcal{A},\mathcal{V}}$, in Algorithm 1.

Algorithm 1 In the ballot privacy experiment $\texttt{Exp}^{\texttt{bpriv},b}_{\mathcal{A},\mathcal{V}}$, the adversary $\mathcal{A}$ has access to the oracles $O = \{O\texttt{LRvote}, O\texttt{cast}, O\texttt{board}, O\texttt{tally}\}$. The adversary controls the tallying server (TS), the voting server (VS) and the certificate authority (CA). It can call $O\texttt{tally}$ only once.

```
1:  $\texttt{Exp}^{\texttt{bpriv},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$:
2:      $(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}}, sk_{\mathrm{T}}) \leftarrow \mathsf{Setup}(1^\epsilon, \mathcal{E}, \mathcal{C})$
3:      $b \leftarrow \mathcal{A}^O(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}})$
4:      Output $b'$
5:  $O\texttt{LRvote}(\tau, C_0, C_1)$:
6:      Let $\beta_0 = \mathsf{CastVote}(\tau, C_0)$ and $\beta_1 = \mathsf{CastVote}(\tau, C_1)$
7:      If $\mathsf{Valid}(\mathrm{PBB}_b, \beta_b) = \perp$ return $\perp$
8:      Else $\mathrm{PBB}_0 \leftarrow \mathrm{PBB}_0 \parallel \beta_0$ and $\mathrm{PBB}_1 \leftarrow \mathrm{PBB}_1 \parallel \beta_1$
9:  $O\texttt{cast}(\beta)$:
10:     If $\mathsf{Valid}(\mathrm{PBB}_b, \beta) = \perp$ return $\perp$
11:     Else $\mathrm{PBB}_0 \leftarrow \mathrm{PBB}_0 \parallel \beta$ and $\mathrm{PBB}_1 \leftarrow \mathrm{PBB}_1 \parallel \beta$
12: $O\texttt{board}()$:
13:     return $\mathrm{PBB}_b$
14: $O\texttt{tally}()$
15:     $(z, \Pi_0) \leftarrow \mathsf{Tally}(\mathrm{PBB}_0, sk_{\mathrm{T}})$
16:     $\Pi_1 = \texttt{SimTally}(\mathrm{PBB}_1, z)$
17:     return $(z, \Pi_b)$
```

Definition 1. *Consider a voting scheme $\mathcal{V}$ = (Setup, Register, CastVote, VoteVerify, Valid, Tally, Verify). We say the scheme has ballot privacy if there exists an algorithm* $\texttt{SimTally}$ *such that for all probabilistic polynomial time adversaries $\mathcal{A}$*

$$\left|\Pr\left[\mathit{Exp}^{\mathit{bpriv},0}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right] - \Pr\left[\mathit{Exp}^{\mathit{bpriv},1}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right]\right|$$

is negligible with respect to ϵ.

Theorem 1. *NetVote provides ballot privacy under the Decisional Diffie–Hellman (DDH) assumption in the random oracle model.*

Proof. We provide a similar proof than the one used to prove that Helios [46] has ballot privacy presented in [45] by using a sequence of games. We initialise the argument with $\texttt{Exp}^{\texttt{bpriv},0}_{\mathcal{A},\mathcal{V}}$ and go step by step until reaching a game equivalent to $\texttt{Exp}^{\texttt{bpriv},1}_{\mathcal{A},\mathcal{V}}$. By showing that each of the transitions between the steps is indistinguishable, we conclude that the two experiments are indistinguishable and hence NetVote satisfies ballot privacy.

Game G_0: Let G_0 be $\texttt{Exp}^{\texttt{bpriv},0}_{\mathcal{A},\mathcal{V}}$ as defined in Algorithm 1 where the adversary has access to the bulletin board PBB_0.

Game G_1: G_1 is defined exactly as G_0 with the exception that the tally proof is simulated. This is, the result is still computed from the votes in PBB_0, but the proof of tally is simulated. The proofs to be simulated are the shuffle proof in Step 4, the proofs of correct decryption in Step 5, and the proofs of greater or equal relation in Step 6, of Procedure 5. Given that all these proofs are zero-knowledge proofs, they require (in order to have the zero-knowledge property) the existence of a simulator algorithm that generates simulations of the proofs that are indistinguishable from real proofs. We use the random oracle to describe as such our SimTally algorithm.

Let L be the number of votes published in the bulletin board. Note that, by how the oracles are defined, this number is equivalent in PBB_0 and PBB_1. We denote β_i the ballots posted in PBB_i for $i \in \{0,1\}$. Next, we proceed with a series of L games where we change (one by one) each entry of posted votes. Let $G_1^0 = G_1$. For $i \in \{1,\ldots,L\}$, we do the following:

Game G_1^i**:** The difference between G_1^i and G_1^{i-1} is only one. If ballot β_0^i of PBB_0 differs from ballot β_1^i of PBB_1, it exchanges β_0^i by β_1^i.

Game G_2**:** We define G_2 as G_1^L. Note that this view is equivalent to the view of $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{bpriv},1}$. Hence, all that remains to prove is that this set of transitions of games G_1^i are indistinguishable among each other.

In order to prove that the transitions made between games G_1^i are indistinguishable, we use the non-malleability under chosen plaintext attack (NM-CPA) property of the ElGamal cryptosystem [47]. This property of ElGamal ensures that an adversary that chooses two plaintexts, and has a challenger encrypt them, is capable of distinguishing which ciphertext encrypts which plaintext with negligible probability. Recall that the ballots are formed by $(\mathsf{Cert}(w_i'), \mathcal{V}, \Pi, \sigma)$, where the identifier and vote are encrypted. However, the process of changing each of the ballots will occur after TS has stripped them, as defined in Step 1 of Procedure 5, mainly, $(\mathsf{Counter}, w_i', \mathcal{V})$.

Hence, when changing two ballots between games G_1^i and G_1^{i+1}, we only need to change the encrypted vote $\mathcal{V}$. We piggyback on this property of ElGamal encryption, and deduce that if an adversary is capable of distinguishing between two consecutive games G_1^i with non-negligible probability, then it is capable of breaking NP-CPA security of ElGamal.

This completes the proof that an adversary can distinguish between $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{bpriv},0}$ and $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{bpriv},1}$ with negligible probability. □

6.2. Practical Everlasting Privacy

We prove how our scheme provides practical everlasting privacy as introduced by Arapinis et al. [48]. A more recent game-based definition of everlasting privacy was presented by Grontas et al. [49], which allows the future adversary to control the electoral entities during the election. Our scheme does not satisfy this stronger model of everlasting privacy as we assume that the information generated by the certificate authority during the election is unreachable to the future adversary. In the definition of Arapinis et al., it is assumed that the adversary can only get information that was posted in the PBB during the election. This is, all information that was exchanged during the election is not accessible to the adversary (such as timing attacks or tokens requests). We propose a game based definition, $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},b}$, similar to $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{bpriv},b}$. Again, $\mathcal{A}$ wins the game if it manages to differentiate between a real and fake world. To model these two worlds, the game tracks two bulletin boards, PBB_0 and PBB_1 for each world respectively. To model such a scenario, we allow the adversary to call on runs of the voting protocol, with electoral roll $\mathcal{E} = V_0, V_1$, and candidate list $\mathcal{C} = C_0, C_1$. Hence, the adversary can make calls to the following oracle:

- $\mathcal{O}\mathtt{RunElection}(V_0, V_1, C_0, C_1)$ where the adversary chooses two voters and two candidates and requests the challenger to run the election. The challenger runs the election, by first running the Setup protocol, generating keys for all parties, and distinct random identifiers to each of the voters. It proceeds with the $\mathsf{Register}(i)$ protocol for each voter, generating voting credentials for each of the voters, then proceeds by casting votes for both voters in both worlds, and, finally, it runs the tally protocol.

Then, $\mathcal{A}$ needs to guess which world it is interacting with. We formally describe the game, $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},b}$, in Algorithm 2. Note that the result is independent of the game bit b, as it will always be one vote for C_0 and one vote for C_1.

Algorithm 2 In the practical everlasting privacy game, $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},b}()$, the adversary $\mathcal{A}$ has access to the oracle $O = \{\mathtt{ORunElection}\}$. In this scenario, the setup needs to happen inside the oracle, so that the voter identifiers are 'reset'.

```
1:  $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},b}(\epsilon, \mathcal{E}, \mathcal{C})$:
2:      $b \leftarrow \mathcal{A}^{O}(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}}, sk_{\mathrm{T}})$
3:      Output $b''$
4:  $\mathtt{ORunElection}(\epsilon, \mathcal{E}, \mathcal{C})$:
5:      $(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}}, sk_{\mathrm{T}}) \leftarrow \mathsf{Setup}(1^{\epsilon}, \mathcal{E}, \mathcal{C})$
6:      Let $\tau_0 = \mathsf{Register}(V_0)$ and $\tau_1 = \mathsf{Register}(V_1)$
7:      Let $\beta_0^b = \mathsf{CastVote}(\tau_0, C_b)$ and $\beta_1^b = \mathsf{CastVote}(\tau_1, C_{1-b})$.
8:      $\mathrm{PBB}_0 \leftarrow \mathrm{PBB}_0 \parallel \beta_0^0 \parallel \beta_1^0$ and $\mathrm{PBB}_1 \leftarrow \mathrm{PBB}_1 \parallel \beta_1^1 \parallel \beta_0^1$
9:      $(z_0, \Pi_0) \leftarrow \mathsf{Tally}(\mathrm{PBB}_0, sk_{\mathrm{T}})$
10:     $(z_1, \Pi_1) \leftarrow \mathsf{Tally}(\mathrm{PBB}_1, sk_{\mathrm{T}})$
11:     return $(z_b, \Pi_b)$
```

Definition 2. *Consider a voting scheme $\mathcal{V} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{CastVote}, \mathsf{VoteVerify}, \mathsf{Valid}, \mathsf{Tally}, \mathsf{Verify})$. We say the scheme has practical everlasting privacy if for a computationally unbounded adversary $\mathcal{A}$*

$$|\Pr\left[\mathit{Exp}_{\mathcal{A},\mathcal{V}}^{\mathit{everbpriv},0}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right] - \Pr\left[\mathit{Exp}_{\mathcal{A},\mathcal{V}}^{\mathit{everbpriv},1}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right]|$$

is negligible with respect to ϵ.

Theorem 2. *NetVote provides practical everlasting privacy.*

Proof. As in the proof of Theorem 1, we begin defining a game corresponding to $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},0}$, and proceed with indistinguishable changes until a game corresponding to $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},1}$, therefore completing the proof.

Game G_0: This is defined as a run of $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},0}$ as defined in Algorithm 2, where the adversary has access to the bulletin board PBB_0.

Game G_1: This game is defined exactly as G_0 with the sole exception that now, we change the register phase, and instead run: $\tau_1 = \mathsf{Register}(V_0)$ and $\tau_0 = \mathsf{Register}(V_1)$

Note that G_1 already corresponds to $\mathtt{Exp}_{\mathcal{A},\mathcal{V}}^{\mathtt{everbpriv},1}$. In order to prove that these two events are statistically (and not computationally) indistinguishable, we recall how the Register phase is defined in NetVote. A voter V_i uses its authentication credentials (e.g., usr/pwd) to request a voting certificate, Cert() to CA. This certificate contains an encrypted voting identifier w_i unique to the voter. However, these identifiers are chosen at random during the Setup protocol, and the link between voter and VId is private to CA (i.e., it is not published in the PBB). Hence, while the adversary is capable of decrypting the identifier in each of the credentials, it is not capable of determining whether a given VId corresponds to V_0 or V_1, as they are chosen at random at each oracle call.

This completes the proof that NetVote provides practical everlasting privacy. □

6.3. Verifiability

There exist several concepts of verifiability in the e-voting scenario. First we have the property known as cast as intended, which allows a voter to verify that the encrypted vote sent to the PBB indeed contains the intended vote. This property aims at detecting attacks performed by the voting hardware. Secondly, we have recorded as cast, allowing voters to verify that the encrypted vote generated by their voting device has been correctly recorded. Finally, the property counted as recorded allows voters to verify that the ballot registered in the bulletin board has been correctly counted in the final tally.

Moreover, voting schemes based on different paradigms require different verifiability definitions. For the case of re-voting schemes, the verifiability game needs to take into account all votes cast by the same voter. As an example, take a voter that casts vote one and verifies it was recorded as cast, but then decides to cast another vote without verifying it was properly recorded. In the re-voting scenario, such cases must be modelled. Several analyses fail to study such corner cases, such as Achenbach et al.'s [26] and Juels et al.'s [16] models.

To this end we base our security analysis on the game presented by Lueks et al. [29], which we introduce below (using the same notation as in the original paper), which is in turn an extension of the work by Cortier et al. [50]. This model excludes the cast as intended property. We follow these lines in our modelling, and consider voting hardware security to be an orthogonal problem to our construction.

We assume that the CA is honest (as it is the sole entity generating the voting credentials), but the adversary controls PBB, VS, TS and the trustees. The game presented in [29] and reproduced in Algorithm 3, $\mathtt{Exp}^{\mathtt{verif},b}_{\mathcal{A},\mathcal{V}}(\epsilon,\mathcal{E},\mathcal{C})$, tracks when voters are corrupted in list $\mathfrak{C}$, as well as the honest voters in $\mathfrak{H}$. The adversary has access to two oracles:

- $O\mathtt{register}(i)$ to get a token for voter V_i.
- $O\mathtt{cast}(i, C)$, to make voter V_i cast a vote for candidate C.

Every voter V_i for which the adversary calls $O\mathtt{register}(i)$ is considered corrupted until it casts an honest vote. Particularly, the game divides voters into three different groups: (i) the corrupted as described above, `Corrupted`, and then honest voters are divided in two groups: (ii) the ones that check that a ballot cast after coercion has been recorded as cast, `Checked`, and (iii) the ones that do not check their ballots were recorded as cast, `Unchecked`, and have not been coerced. Similarly, the game tracks which are the candidates that the system is allowed to exchange for each voter, `AllowedCandidates`. In particular, this list contains the candidates cast during or after the last checked vote cast. In other words, the final tally must contain the last checked vote or a later one.

At the end of the game, $\mathcal{A}$ returns the state of the PBB, the result of the election and the proof of correctness. The adversary wins if the result and corresponding proofs verify, but manages to cheat the system, i.e., (i) a higher number of corrupted votes than the number of voters it corrupted, $n_{\mathfrak{C}}$, are counted, (ii) for some voter that verified a ballot, the result includes a candidate cast only before that event (i.e., a candidate chosen only before verifying), or (iii) the result includes a candidate not cast at any point during the election by a voter that did not check its submissions.

In the game experiment, the tally is produced individually by groups (as introduced above), and therefore requires the scheme to support partial tallying, for example,

$$\mathsf{Tally}(\mathtt{Corrupted} \cup \mathtt{Checked}) = \mathsf{Tally}(\mathtt{Corrupted}) \star \mathsf{Tally}(\mathtt{Checked}),$$

where $\star\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is a binary function that ads both of the partial tallies. Note that our tally function performs the homomorphic addition of votes, and therefore supports partial tallying. We formally describe the game, $\mathtt{Exp}^{\mathtt{verif},b}_{\mathcal{A},\mathcal{V}}(\epsilon,\mathcal{E},\mathcal{C})$, and corresponding oracles, in Algorithm 3.

Definition 3. *Consider a voting scheme $\mathcal{V}$ = (Setup, Register, CastVote, VoteVerify, Valid, Tally, Verify). We say the scheme is verifiable if for all probabilistic polynomial time adversaries $\mathcal{A}$*

$$\left|\Pr\left[Exp^{verif,0}_{\mathcal{A},\mathcal{V}}(\epsilon,\mathcal{E},\mathcal{C}) = 1\right] - \Pr\left[Exp^{verif,1}_{\mathcal{A},\mathcal{V}}(\epsilon,\mathcal{E},\mathcal{C}) = 1\right]\right|$$

is negligible with respect to ϵ.

Theorem 3. *NetVote is verifiable under the DDH assumption in the random oracle model.*

Proof. We provide a similar proof to the one presented by Lueks et al. [29], by showing that dummy votes are not counted in the final tally and that at least a cast vote has been counted (one later than the last verification). At the end of the election, the adversary outputs z, Π. Because Π validates, we know that the result z is the addition of the filtered votes from the PBB. Now it remains to show that the filtered votes do indeed satisfy the conditions imposed by the game.

Let B be the stripped ballots in PBB once the election closes. We know that all these votes originate from a valid ballot cast by a voter, and hence are accompanied by a proof that they contain an encryption of a valid candidate.

Let n be the number of different voters that cast at least a ballot. We argue that the number of ballots included in the tally equals n. Because of the correctness of the shuffle, Π_s, we know that these same ballots will be present after the shuffle, and hence votes of voters $1, \ldots, n$ are present in B'. The filtering procedure groups votes cast by the same voter, and takes only the vote with the highest counter (where the counter is unique per PBB entry starting at one). Because of the validity of the decryption proof, Π_d, we know that the filtering is applied to votes cast by the same voter (recall that we assume the honesty of CA for verifiability). Moreover, given the correctness of the greater than proofs, $\Pi_{GT,i}$ for i ranging all indexes of votes cast by the same voter, we know that only the ballot with higher index was counted for each voter. Finally, all dummy votes are cast with non encrypted zero counter, and therefore it is impossible that a dummy vote supersedes a real vote cast by a voter.

Algorithm 3 Verifiability game presented by Lueks et al. [29]. In the verifiability game experiment $\mathrm{Exp}^{\texttt{verif},b}_{\mathcal{A},\mathcal{V}}$, the adversary $\mathcal{A}$ has access to the oracles $O = \{\mathcal{O}\texttt{register}, \mathcal{O}\texttt{cast}\}$.

```
1:  $\mathrm{Exp}^{\texttt{verif},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$:
2:    $(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}}, sk_{\mathrm{T}}) \leftarrow \mathsf{Setup}(1^{\epsilon}, \mathcal{E}, \mathcal{C})$
3:    Set $\mathfrak{H} \leftarrow \emptyset$ and $\mathfrak{C} \leftarrow \emptyset$
4:    $(\mathrm{PBB}, z, \Pi) \leftarrow \mathcal{A}^{O}(pk, sk_{\mathrm{TS}}, sk_{\mathrm{T}})$
5:    If $\mathsf{Verify}(\mathrm{PBB}, z, \Pi) = \perp$ return 0
6:    Let $\texttt{Verified} = \{(i_1, \mathsf{Counter}_1), \ldots, (i_{n_v}, \mathsf{Counter}_{n_v})\}$ correspond to checked ballots.
7:    Let $\texttt{Corrupted} = \{i \mid (i, \mathsf{Counter}) \in \mathfrak{C} \wedge \forall (i, \mathsf{Counter}') \in \texttt{Verified} : \mathsf{Counter}' < \mathsf{Counter}\}$
8:    Let $\texttt{Checked} = \{i \mid (i, \_) \in \texttt{Verified}\} \setminus \texttt{Corrupted}$
9:    Let $\texttt{Unchecked} = \{i \mid (i, \_, \_) \in \mathfrak{H} \wedge (i, \_) \notin \mathfrak{C}\} \setminus \texttt{Checked}$
10:   Let $\texttt{AllowedCandidates}[i] = \{C \mid (i, \mathsf{Counter}, C) \in \mathfrak{H}$ s.t. $\forall (i, \mathsf{Counter}') \in \texttt{Verified} : \mathsf{Counter} \geq \mathsf{Counter}'\}$
11:   If $\exists\, C^V_1, \ldots, C^V_{n_V}$ s.t. $C^V_j \in \texttt{AllowedCandidates}[i^V_j]$ where $\texttt{Checked} = \{i^V_1, \ldots, i^V_{n_V}\}$
12:     $\exists\, (i^U_1, C^U_1), \ldots, (i_{n_U}, C^U_{n_U})$ s.t. $i^U_j \in \mathsf{Unchecked}, C^U_j \in \mathsf{AllowedVotes}[i^U_j]$, $i^U_j$ distinct
13:     $\exists\, C^B_1, \ldots, C^B_{n_B} \in \mathcal{C}$ s.t. $0 \leq n_B \leq |\mathsf{Corrupted}|$
14:     s.t. $z = \mathsf{Tally}(\{C^V_i\}_{i=1}^{n_V}) \star \mathsf{Tally}(\{C^U_i\}_{i=1}^{n_U}) \star \mathsf{Tally}(\{C^B_i\}_{i=1}^{n_B})$
15:   Then return 0, otherwise return 1
16: $\mathcal{O}\texttt{cast}(i, C)$:
17:   $\tau = \mathsf{Register}(i)$
18:   Add $(i, \texttt{\#tokens}(i), C)$ to $\mathfrak{H}$
19:   Return $\mathsf{CastVote}(\tau, C)$
20: $\mathcal{O}\texttt{register}(i)$:
21:   Let $\tau = \mathsf{Register}(i)$
22:   Add $(i, \texttt{\#tokens}(i))$ to $\mathfrak{C}$
23:   return $\tau$
```

Hence, we know that votes counted in the final tally are the last votes recorded in the PBB by each voter that took part in the election. These voters are part of one of the three groups of our definition. What remains is to prove that the conditions are met for each of these three groups.

First, we show that the last checked vote or one after is counted for each voter in $\texttt{Checked}$. Consider voter $V_i \in \texttt{Checked}$, and let $\mathsf{Counter}_v$ be the counter of the last ballot it checked was recorded as cast. We know that the ballot with counter $\mathsf{Counter}_v$ was added to PBB, therefore in the grouping after the shuffle, we know that the group for voter V_i will exist, with at least the ballot with counter $\mathsf{Counter}_v$. Given that the selected ballot of each group is accompanied with a proof that the counter is greatest among the group, the selected ballot must be the one with counter $\mathsf{Counter}_v$ or one cast afterwards. Given the proof of decryption of the homomorphic tally among all selected votes, we know that either ballot $\mathsf{Counter}_v$ or a later ballot by voter V_i is counted in the final tally.

Now consider a voter $V_i \in \texttt{Unchecked}$. Then, by the same argument as above, we know that the tally either drops all ballots, or counts one of the ballots cast by the voter. In other words, the tally cannot add a vote not cast by voter V_i.

Finally, all remaining voters correspond to group $\texttt{Corrupted}$. Notice that by the arguments above, the tally procedure cannot include any votes by voters who did not cast a vote. Moreover, only one vote per grouped votes is selected. Hence, it follows that the size of this group is at most the number of corrupted voters, $n_{\mathfrak{C}}$. $\square$

6.4. Strict Coercion Resistance

The advantage that our padding strategy gives us in the linear filtering procedure against current literature, obliges us to weaken the coercion resistance property that our scheme satisfies. In the work by Lueks et al. [29], a new coercion resistance definition is presented for the re-voting setting. However, their quasi linear deterministic padding strategy ensures that the same number of ballots is added regardless of how voters voted. This strategy has the downside of producing a filtering stage with complexity $O(N \log N)$, with N being the number of voters. However, it facilitates the task of modelling coercion resistance. To see this in comparison with our construction, consider an election where two voters, V_1, V_2, vote. V_1 casts one vote, while V_2 casts 1009 votes. Then, if the coercer obliged V_1 to cast one vote, then with high probability (0.2 in the case of the negative binomial distribution with probability of success 0.2) no vote will be added. However, we expect an election to hide patterns of voters casting a small amount of votes, as the expected behaviour of voters is to cast only once. Nonetheless, patterns of voters which are forced to cast a higher number of votes, for example, 1009, are not expected to be hidden by other voters. In this scenario, if voter V_2 wants to sell its vote and not escape coercion it can prove so with non negligible probability. While the probability remains small, it is still bounded by the probability distribution that we use for adding dummies, and hence non-negligible. Note, however, that if the coerced voter wants to escape coercion, the coercer cannot determine whether the voter really escaped or simply dummy votes were added. To model such a difference we need to differentiate between coercion-resistance and vote-buying. To this end we need to briefly modify the definition of Lueks et al. [29].

Strict Coercion Resistance: To offer a linear filtering phase, we propose a new game-based coercion-resistance definition, $\texttt{Exp}^{\texttt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$, that we name strict coercion-resistance, inspired by Lueks et al.'s coercion resistance definition. The game tracks two bulletin boards, PBB_0, PBB_1, of which the adversary has access to only one. In our definition we do not require the adversary to not be able to distinguish between a voter resisting or submitting to coercion. We only require that, if the voter escapes coercion, the coercer cannot determine whether it decided to resist coercion and cast another vote, or if on the contrary, the voter decided to submit to coercion. The goal of the adversary is to determine which run of the experiment it is seeing (see Algorithm 4). To model this, we provide the adversary with five oracles that it can call throughout the game:

- $\texttt{OvoteDR}(i_0, C_0, i_1, C_1)$, to make voter V_{i_0} cast a vote for candidate C_0, and voter V_{i_1} cast a dummy vote for a dummy candidate, 0, in PBB_0, and to make voter V_{i_1} cast a vote for candidate C_1, and voter V_{i_0} cast a dummy vote for a dummy candidate, 0, in PBB_1. We use, $\mathsf{RegisterDummy}(i)$ to denote the dummy registration of voter V_i. Moreover, we denote by $\mathsf{CastVoteDummy}(\tau, C)$ the dummy vote cast for candidate C using token τ. The adversary is allowed to make this call multiple times.

Note that regardless of the call, two votes are added to both bulletin boards. This oracle models the situation we want to protect our voters from. If they wish to escape coercion, the adversary will not be able to distinguish that situation with one where a dummy vote is added. Note that in our scheme, the dummy votes are added once the election is closed. However, in the game we model the dummy vote addition in parallel to the voting stage. The rest of the oracles are modelled as in the work by Lueks et al., with the difference that our scheme does not keep state in the voting tokens.

- $\mathcal{O}\mathtt{register}(i)$, allows the adversary to register and obtain a token, τ, for voter V_i.
- $\mathcal{O}\mathtt{cast}(\tau, C)$, using a token τ, the adversary can call this oracle to cast a vote for candidate C in PBB_0 and PBB_1.
- $\mathcal{O}\mathtt{board}()$ allows the adversary to see the bulletin board.
- $\mathcal{O}\mathtt{tally}()$ allows the adversary to compute the tally of the election. It can call this oracle only once.

Algorithm 4 In the strict coercion resistance experiment, $\mathtt{Exp}^{\mathtt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$, adversary $\mathcal{A}$ has access to oracles $O = \{\mathcal{O}\mathtt{voteDR}, \mathcal{O}\mathtt{cast}, \mathcal{O}\mathtt{board}, , \mathcal{O}\mathtt{register}, \mathcal{O}\mathtt{tally}\}$. It can call $\mathcal{O}\mathtt{tally}$ only once.

```
1:  $\mathtt{Exp}^{\mathtt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$:
2:    $(pk, sk_{\mathrm{CA}}, sk_{\mathrm{TS}}, sk_{\mathrm{T}}) \leftarrow \mathsf{Setup}(1^{\epsilon}, \mathcal{E}, \mathcal{C})$
3:    $b' \leftarrow \mathcal{A}^{O}(pk, pk_{\mathrm{CA}})$
4:    Output $b'$
5:  $\mathcal{O}\mathtt{voteDR}(i_0, C_0, i_1, C_1)$:
6:    Let $\tau_0 \leftarrow \mathsf{Register}(i_0)$ and $\tau'_1 \leftarrow \mathsf{RegisterDummy}(i_1)$
7:      $\tau_1 \leftarrow \mathsf{Register}(i_1)$ and $\tau'_0 \leftarrow \mathsf{RegisterDummy}(i_0)$
8:    Let $\beta_0 = \mathsf{CastVote}(\tau_0, C_0)$ and $\beta'_1 = \mathsf{CastVoteDummy}(\tau'_1, 0)$
9:      $\beta_1 = \mathsf{CastVote}(\tau_1, C_1)$ and $\beta'_0 = \mathsf{CastVoteDummy}(\tau'_0, 0)$
10:   If $\mathsf{Valid}(\mathrm{PBB}_b, \beta_b) = \perp$ or
11:     $\mathsf{Valid}(\mathrm{PBB}_b, \beta'_{1-b}) = \perp$ return $\perp$
12:   Else $\mathrm{PBB}_0 \leftarrow \mathrm{PBB}_0 \parallel \beta_0 \parallel \beta'_1$ and
13:     $\mathrm{PBB}_1 \leftarrow \mathrm{PBB}_1 \parallel \beta_1 \parallel \beta'_0$
14: $\mathcal{O}\mathtt{register}(i)$:
15:   Let $\tau \leftarrow \mathsf{Register}(i)$
16:   return $\tau$
17: $\mathcal{O}\mathtt{cast}(\tau, C)$:
18:   If $\mathsf{Valid}(\mathrm{PBB}_0, \beta) = \perp$ or
19:     $\mathsf{Valid}(\mathrm{PBB}_1, \beta) = \perp$ return $\perp$
20:   Else $\mathrm{PBB}_0 \leftarrow \mathrm{PBB}_0 \parallel \beta$ and
21:     $\mathrm{PBB}_1 \leftarrow \mathrm{PBB}_1 \parallel \beta$
22: $\mathcal{O}\mathtt{board}()$:
23:   return $\mathrm{PBB}_b$
24: $\mathcal{O}\mathtt{tally}()$
25:   Let $(z, \Pi_0) \leftarrow \mathsf{Tally}(\mathrm{PBB}_0, sk_{\mathrm{T}})$
26:   $\Pi_1 = \mathtt{SimTally}(\mathrm{PBB}_1, z)$
27:   return $(z, \Pi_b)$
```

Again, as in the ballot privacy game, the result is always computed from the same bulletin board, to avoid leakage of information given by the result. To this end, the game simulates the tally and proofs of correctness in case the game is using PBB_1. We stress that this game models the 1009 attack only when the voter decides to escape coercion. The vote-selling where the voter decides to submit to the adversary and prove that it did not resubmit is not hereby modelled. At the end of the game, the adversary needs to output a guess, b'. If it guesses correctly with non-negligible probability, it wins the game. We formally define $\texttt{Exp}^{\texttt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$ in Algorithm 4.

Definition 4. *Consider a voting scheme $\mathcal{V} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{CastVote}, \mathsf{VoteVerify}, \mathsf{Valid}, \mathsf{Tally}, \mathsf{Verify})$. We say the scheme has strict coercion resistance if there exists algorithm* `SimTally` *such that for all probabilistic polynomial time adversaries $\mathcal{A}$*

$$\left|\Pr\left[\mathit{Exp}^{\mathit{coer},0}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right] - \Pr\left[\mathit{Exp}^{\mathit{coer},1}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C}) = 1\right]\right|$$

Theorem 4. *$\mathsf{NetVote}$ has strict coercion resistance under the DDH assumption in the random oracle model.*

Proof. As in the proof of Theorem 1, we construct our `SimTally` algorithm by leveraging the zero-knowledge property of the zero-knowledge proofs used in the tally. Namely, `SimTally` simulates the proofs of shuffle, decryption, greater than, and finally, the proof of decryption of the added votes (Procedures 5 and 6).

We follow a similar proof than that presented by Lueks et al., which proceeds by a series of games, replacing all the ballots that depend on the bit b. If all steps used throughout the replacement of these ballots are indistinguishable, strict coercion resistance follows.

Game G_1: This game is defined as $\texttt{Exp}^{\texttt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$ of Algorithm 4. Note that we differ from the proof of ballot privacy in that we do not start with a fixed value of b.

Game G_2: Later, we are going to replace all votes by random votes. To this end, in this game, we compute the result taking the decrypted ballots of PBB_0. Consider the stripped ballots of Step 3 of Procedure 5, $(\mathsf{EncCounter}, \mathcal{V}, w')$. It computes the tally of these ballots, $\mathsf{Tally}((\mathsf{EncCounter}, \mathcal{V}, w')_{i=1}^{n_l})$, where n_l is the total number of votes cast. It does so by first decrypting $\mathsf{EncCounter}, \mathcal{V}, w'$ using $sk_{\text{TS}}, sk_{\text{T}}$ and sk_{TS} respectively. Now, it proceeds by computing the final result by taking the last vote cast per VId_i. Note that, as the result is always taken from PBB_0 and the game does not publish these decrypted values, game G_2 is indistinguishable from game G_1.

Game G_3: Same as the previous game, but now all zero-knowledge proofs, regardless of b, are replaced by simulations. This is, the proof of shuffle, Π_s, of all votes (including the dummies), the proof of correct decryption, Π_d, of each identifier, VId_i, the greater than proofs used to filter all but the last vote, $\Pi_{GT,i}$, and finally, the proof of decryption of the final result, Π_z.

Due to the zero-knowledge property, we can use the random oracle to simulate this step, making the difference between game G_2 and G_3 indistinguishable.
Our goal now is to exchange all identifying ciphertexts (mainly w', $\mathsf{EncCounter}$, and the corresponding values after the shuffle) by random ciphertexts. Note that the proofs are simulated, and the result (filter and tally) is calculated from the decrypted votes of game G_2, so the decryption, shuffle, and tally are now independent of the actual values of the encrypted votes.

Game G_4: We define G_4 the same as G_3 with the exception that we exchange all ciphertexts w' in the certificates by random ciphertexts. Note that due to the changes of G_2, the result is correctly calculated from the votes initially in PBB_0 and hence this does not affect the filtering stage. A hybrid argument reduces the indistinguishability of games G_4 and G_3 to the CPA security of ElGamal encryption. Note that this reduction is possible as we no longer need to decrypt the ciphertexts in the tallying.

Game G_5: We follow by replacing all votes cast by $O\texttt{voteDR}()$ by zero votes. Again, following the lines of the ballot privacy proof, the indistinguishability of this step is reduced to the NM-CPA security of ElGamal.

Game G_6: To ensure that the filter does not leak information to the coercer, we also replace all ciphertexts generated thereafter. We replace the encryption of the counters, EncCounter, by random ciphertexts. We do the same with all ciphertexts that exit the shuffle. Namely, we replace the shuffled encrypted counters, encrypted votes and encrypted VIds. This exchange is possible as we do not need to decrypt the ciphertexts (as of game G_2). Moreover, the indistinguishability of this step follows from the simulation of the zero-knowledge proofs (of game G_3) and the NM-CPA security of the encryption scheme.

The resulting game is clearly independent of b. Due to indistinguishable changes that we made to achieve this last game, we conclude that game G_1, and therefore $\texttt{Exp}^{\texttt{coer},b}_{\mathcal{A},\mathcal{V}}(\epsilon, \mathcal{E}, \mathcal{C})$, are independent of the game bit b, and hence strict coercion resistance follows. □

7. Discussion on the Assumptions of Fake Credentials vs. Re-Voting

Coercion in remote elections is a hard problem that requires strong assumptions regarding the adversary and the user interaction with the election. As presented in Section 2, current solutions mitigate coercion either by the use of fake credentials, or by allowing voters to re-vote. In this section we give an intuition of the motivation of choosing the latter. We do not analyse specific constructions, but rather study the intrinsic limitations of each approach.

For the fake credentials the assumptions are the following:

1. User needs inalienable means of authentication.
2. User needs to lie convincingly while being coerced.
3. User needs to store cryptographic material securely and privately.
4. Coercer needs to be absent during the registration and at some point during the election.

In the case of the re-voting elections, the assumptions are as follows:

1. User needs inalienable means of authentication.
2. Coercer needs to be absent at the end of the election.

Current work has failed to present a solution to remove the assumption on inalienable means of authentication, and it seems that it is an intrinsic problem to remote voting. A voter, either to register or to cast a vote, will need to authenticate itself to prove that it is an eligible voter. If these authentication means can be removed by an adversary, then coercion is inevitable.

In our opinion, fake credential schemes seem to have stronger assumptions. Lying convincingly to an adversary while voting can be a challenge to some. However, this is not the only limitation. It is clear from how much money it has been lost in the cryptocurrency world (because of losing keys) that storing cryptographic credentials securely and privately is not obvious [51]. Moreover, having to store the cryptographic material in a device opens attack vectors for a coercer to impede re-voting, simply by removing the device where these keys are stored. Last but not least, a study by Neto et al. [52] concluded that more than 90% of the participants did not understand the concept of casting fake votes, and were uncomfortable with the fact of not being able to distinguish between real or fake votes at the time of casting. This study puts the usability of fake credential systems for the common public under question.

Finally, we argue on the difference of the strength of the assumptions regarding the absence of the adversary. Both approaches require that the coercer is absent during a fixed period. In the re-voting setting the coercer must not allow to cast a vote to the coerced voter at the end of the election. As long as the voter is left alone enough time to cast a vote (say 5 min), the voter will be able to escape coercion. In the other case, the coercer needs to be absent during registration and some time during the election.

In general, a registration process can happen across several hours or days (in Spain it is 24 days for citizens living abroad [53]).

Hence, in the 'registration-scenario' it would not suffice for the adversary to be absent for five minutes. Clearly, both limitations allow an adversary to successfully perform a targeted attack. However, this intuitively shows that producing a large scale coercion attack in a time frame of 5 min is harder than performing one during the registration phase. This motivates our choice of a re-voting scenario rather than fake credentials.

8. Conclusions and Future Work

The core requirement of elections has not changed since it was introduced by the Greeks: there must be proof of the final result. Whether it was putting pebbles in an urn, using marked paper in a ballot box or using cryptography to prove the results. While the transparency of the counting (and the possibility to repeat it to consolidate the results) is unarguable, the guarantee of some other properties, such as coercion-resistance, ballot secrecy or voter eligibility, has not always received the same importance. Technology now offers us the possibility of doing the tally in a provable and reproducible way, but producing a scheme with the required properties that does not use paper at all, and that is usable, is something that is still under research. It is clear that the migration from standard voting to electronic voting will have to consider trade-off's, at least in the point of time we are now, so the problem that has to be solved in the present is which trade-off's are to be done in which situations. For instance, presence voting can highly be improved by hybrid schemes, where trust is removed, and a dual system using paper and cryptography improves the verifiability and election experience. For the case of remote elections, it depends on requirements, as if a presence registration process is to be considered, high levels of eligibility verification and coercion-resistance can be reached. However, for the case where the whole process is to be done remotely, we have seen how proposals allow a voter to mitigate coercion and guarantee ballot secrecy by sacrificing the usability and implementability of the scheme.

We have presented a protocol that stands in between. The requirements for deployment are low, the usability is very high and we offer verifiability, ballot secrecy, practical everlasting privacy, and strict coercion-resistance. The complexity is also improved in comparison to existing schemes, and allows an organisation to employ remote voting elections in a secure, deployable and verifiable manner without the requirement of voters having cryptographic keys.

Our current lines of investigation are directed towards diminishing the trust assumptions of our model, and we are studying what are the reaches of our scheme if we assume the existence of voter cryptographic keys. Moreover, future work will study the exact probability of success of vote buying under these circumstances, and try to prove coercion resistance under stronger existing definitions. An evaluation study will proceed, with an implementation of the system, to study the execution times and resources required for a large scale election, and a user study to present usability results.

Author Contributions: Conceptualisation, investigation, methodology, and writing—review and editing, I.Q.-A., L.H.E., J.L.H.-A. and D.A.G.; formal analysis and writing—original draft preparation, I.Q.-A.; resources, I.Q.-A., L.H.E. and J.L.H.-A.; supervision, project administration and funding acquisition, L.H.E. and J.L.H.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially supported by Ministerio de Economía, Industria y Competitividad (MINECO), Agencia Estatal de Investigación (AEI), and European Regional Development Fund (ERDF, EU), through project COPCIS, grant number TIN2017-84844-C2-1-R, and by Comunidad de Madrid (Spain) through project CYNAMON, grant number P2018/TCS-4566-CM, co-funded along with ERDF.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DDH	Decisional Diffie–Hellman
CPA	Chosen Plaintext Attack
NM-CPA	Non-Malleability under Chosen Plaintext Attack
V	Voters
T	Trustees
CA	Certificate Authority
VS	Voting Server
TS	Tallying Server
PBB	Public Bulletin Board

References

1. The Guardian. Dutch Will Count All Election Ballots by Hand to Thwart Hacking. May 2018. Available online: https://www.theguardian.com/world/2017/feb/02/dutch-will-count-all-election-ballots-by-hand-to-thwart-cyber-hacking (accessed on 3 May 2020).
2. Reuters. France Drops Electronic Voting for Citizens Abroad Over Cybersecurity Fears. 2017. Available online: http://www.reuters.com/article/us-france-election-cyber-idUSKBN16D233 (accessed on 3 May 2020).
3. Hapsara, M.; Imran, A.; Turner, T. E-Voting in Developing Countries. *Lect. Notes Comput. Sci.* **2017**, *10141*, 36–55. [CrossRef]
4. Electoral-Comission. The Administration of the June 2017 UK General Election. May 2018. Available online: https://www.electoralcommission.org.uk/__data/assets/pdf_file/0003/238044/The-administration-of-the-June-2017-UK-general-election.pdf (accessed on 3 May 2020).
5. Chase, J. German Election Could Be Won by Early Voting. May 2018. Available online: http://www.dw.com/en/german-election-could-be-won-by-early-voting/a-40296550 (accessed on 3 May 2020).
6. Ministerio del Interior. Voto Desde Fuera de España. 2013. Available online: http://www.infoelectoral.mir.es/voto-desde-fuera-de-espana (accessed on 3 May 2020).
7. Ministère de l'Europe et des Affaires étrangères. Vote par Correspondance. May 2018. Available online: https://www.diplomatie.gouv.fr/fr/services-aux-francais/voter-a-l-etranger/modalites-de-vote/vote-par-correspondance/ (accessed on 3 May 2020).
8. United Kingdom Government. Completing and Returning Your Postal Vote. May 2018. Available online: https://www.gov.uk/voting-in-the-uk/postal-voting (accessed on 3 May 2020).
9. Barker, E.B. SP 800-57. Recommendation for Key Management, Part 1: General; Available online: http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4 (accessed on 3 May 2020).
10. Ministerio del Interior. La Subsecretaria Soledad López Explica en el Senado las Características del DNI Electrónico. 2006. Available online: https://goo.gl/r6MVYJ (accessed on 3 May 2020).
11. Gimeno, M.; Villamía-Uriarte, B.; Suárez-Saa, V. eEspaña—Informe Anual Sobre el Desarrollo de la Sociedad de la Información en España. 2014. Available online: https://www.proyectosfundacionorange.es/docs/eE2014/Informe_eE2014.pdf (accessed on 3 May 2020).
12. Cripps, H.; Standing, C.; Prijatelj, V. Smart health case cards: Are they applicable in the australian context? In Proceedings of the 25th Bled eConference eDependability: Reliable and Trustworthy eStructures, eProcesses, eOperations and eServices for the Future, Bled, Slovenia, 17–20 June 2012.
13. Hernandez-Ardieta, J.L.; Gonzalez-Tablas, A.I.; De Fuentes, J.M.; Ramos, B. A Taxonomy and Survey of Attacks on Digital Signatures. *Comput. Secur.* **2013**, *34*, 67–112. [CrossRef]
14. Bernstein, D.J.; Chang, Y.A.; Cheng, C.M.; Chou, L.P.; Heninger, N.; Lange, T.; van Someren, N. Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild. Advances in Cryptology—ASIACRYPT'2013. In Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, 1–5 December 2013; Sako, K., Sarkar, P., Eds.; pp. 341–360. [CrossRef]
15. Nemec, M.; Sys, M.; Svenda, P.; Klinec, D.; Matyas, V. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS'2017), Dallas, TX, USA, 30 October–3 November 2017; ACM: New York, NY, USA, 2017; pp. 1631–1648. [CrossRef]

16. Juels, A.; Catalano, D.; Jakobsson, M. Coercion-resistant Electronic Elections. In Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society; WPES '05, Alexandria, VA, USA, 7 November 2005; ACM: New York, NY, USA, 2005; pp. 61–70. [CrossRef]
17. Araújo, R.; Barki, A.; Brunet, S.; Traoré, J. Remote electronic voting can be efficient, verifiable and coercion-resistant. In *Financial Cryptography and Data Security—2016 International Workshops, BITCOIN, VOTING, and WAHC*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9604, pp. 224–232. [CrossRef]
18. Bursuc, S.; Grewal, G.S.; Ryan, M.D., Trivitas: Voters Directly Verifying Votes. In *E-Voting and Identity: Proceedings of the Third International Conference, VoteID 2011, Tallinn, Estonia, 28–30 September 2011*; Revised Selected Papers; Kiayias, A., Lipmaa, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 190–207. [CrossRef]
19. Clark, J.; Hengartner, U. Selections: Internet voting with over-the-shoulder coercion-resistance. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 47–61. [CrossRef]
20. Myers, A.C.; Clarkson, M.; Chong, S. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*; IEEE: Piscataway, NJ, USA, 2008; pp. 354–368. [CrossRef]
21. Grontas, P.; Pagourtzis, A.; Zacharakis, A.; Zhang, B. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In *Financial Cryptography and Data Security*; Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 210–231. [CrossRef]
22. Rønne, P.B.; Atashpendar, A.; Gjøsteen, K.; Ryan, P.Y.A. Short paper: Coercion-resistant voting in linear time via fully homomorphic encryption. In *Financial Cryptography and Data Security*; Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 289–298.
23. Spycher, O.; Haenni, R.; Dubuis, E. Coercion-resistant hybrid voting systems. In Proceedings of the Electronic Voting 2010, EVOTE 2010, 4th International Conference, Bregenz, Austria, 21– 24 July 2010; pp. 269–282.
24. Gjøsteen, K. Analysis of an Internet Voting Protocol; Technical Report. Available online: https://eprint.iacr.org/2010/380.pdf (accessed on 3 May 2020).
25. Dimitriou, T. Efficient, Coercion-free and Universally Verifiable Blockchain-based Voting. *Comput. Netw.* **2020**, *174*, 107234. [CrossRef]
26. Achenbach, D.; Kempka, C.; Löwe, B.; Müller-Quade, J. Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting. *USENIX J. Elect. Technol. Syst. JETS* **2015**, *3*, 26–45.
27. Locher, P.; Haenni, R.; Koenig, R.E. Coercion-resistant internet voting with everlasting privacy. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 161–175. [CrossRef]
28. Locher, P.; Haenni, R. Verifiable Internet Elections with Everlasting Privacy and Minimal Trust. In *E-Voting and Identity*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 74–91. [CrossRef]
29. Lueks, W.; Querejeta-Azurmendi, I.; Troncoso, C. VoteAgain: A scalable coercion-resistant voting system. In Proceedings of 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 1553–1570
30. Moran, T.; Naor, M. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In Proceedings of the 26th Annual International Conference on Advances in Cryptology; CRYPTO'06, Santa Barbara, CA, USA, 20–24 August 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 373–392. [CrossRef]
31. Querejeta-Azurmendi, I.; Hernández Encinas, L.; Arroyo Guardeño, D.; Hernandez-Ardieta, J.L. An internet voting proposal towards improving usability and coercion resistance. In *Advances in Intelligent Systems and Computing, Proceedings of the International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019), Seville, Spain, 13–15 May 2019*; Advances in Intelligent Systems and Computing; Martínez-Álvarez, F., Lora, A.T., Muñoz, J.A.S., Quintián, H., Corchado, E., Eds.; Springer: Cham, Switzerland, 2019; Volume 951, pp. 155–164. [CrossRef]
32. Pedersen, T.P. A Threshold Cryptosystem without a Trusted Party. In *Advances in Cryptology—EUROCRYPT'91*; Springer: Berlin/Heidelberg, Germany, 1991; Volume 547, pp. 522–526. [CrossRef]
33. Groth, J. A Verifiable Secret Shuffle of Homomorphic Encryptions. *J. Cryptol.* **2010**, *23*, 546–579. [CrossRef]
34. Bayer, S.; Groth, J. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. Available online: http://www0.cs.ucl.ac.uk/staff/J.Groth/MinimalShuffle.pdf (accessed on 3 May 2020).

35. Goldwasser, S.; Micali, S.; Rackoff, C. The Knowledge Complexity of Interactive Proof-Systems. In *STOC '85: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*; Sedgewick, R., Ed.; Association for Computing Machinery: New York, NY, USA, 1985; pp. 291–304. [CrossRef]
36. Fiat, A.; Shamir, A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings of the Conference on Advances in Cryptology—CRYPTO '86*; Springer: Berlin/Heidelberg, Germany, 1987; pp. 186–194. [CrossRef]
37. Camenisch, J.; Stadler, M. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Proceedings of the Conference 17th Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 1997. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.1954&rep=rep1&type=pdf (accessed on 3 May 2020)
38. Bünz, B.; Bootle, J.; Boneh, D.; Poelstra, A.; Wuille, P.; Maxwell, G. Bulletproofs: Short Proofs for Confidential Transactions and More. In Proceedings of the 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, San Francisco, CA, USA, 21–23 May 2018; IEEE Computer Society: Piscataway, NJ, USA, 2018; pp. 315–334. [CrossRef]
39. Camenisch, J.; Lysyanskaya, A., A Signature Scheme with Efficient Protocols. In *Security in Communication Networks: Proceedings of the Third International Conference, SCN 2002 Amalfi, Italy, 11–13 September 2002*; Revised Papers; Cimato, S., Persiano, G., Galdi, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 268–289. [CrossRef]
40. Brands, S.A. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*; MIT Press: Cambridge, MA, USA, 2000.
41. Park, S.; Park, H.; Won, Y.; Lee, J.; Kent, S. Traceable Anonymous Certificate. Available online: https://tools.ietf.org/pdf/rfc5636 (accessed on 3 May 2020).
42. Heather, J.; Lundin, D. The append-only web bulletin board. In *Formal Aspects in Security and Trust*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 242–256, doi:10.1007/978-3-642-01465-9_16. [CrossRef]
43. McCorry, P.; Shahandashti, S.F.; Hao, F. A smart contract for boardroom voting with maximum voter privacy. In *Financial Cryptography and Data Security*; LNCS; Springer: Berlin/Heidelberg, Germany, 2017; pp. 357–375. Available online: https://dblp.org/rec/bib/conf/fc/McCorrySH17 (accessed on 3 May 2020). [CrossRef]
44. Smith, W. New cryptographic election protocol with best-known theoretical properties. In Proceedings of the ECRYPT-Frontiers in Electronic Elections (FEE), Milan, Italy, 15–16 September 2005.
45. Bernhard, D.; Cortier, V.; Galindo, D.; Pereira, O.; Warinschi, B. SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, 17–21 May 2015; IEEE Computer Society: Piscataway, NJ, USA, 2015; pp. 499–516. [CrossRef]
46. Adida, B. Helios: Web-based Open-audit Voting. In Proceedings of the 17th Conference on Security Symposium, SS'08; USENIX, Boston, MA, USA, 22–27 June 2008; Association: Berkeley, CA, USA, 2008; pp. 335–348.
47. Bernhard, D.; Pereira, O.; Warinschi, B. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. Available online: https://eprint.iacr.org/2016/771.pdf (accessed on 3 May 2020).
48. Arapinis, M.; Cortier, V.; Kremer, S.; Ryan, M. Practical everlasting privacy. In *Principles of Security and Trust—Proceedings of the Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, 16–24 March 2013*; Lecture Notes in Computer Science; Basin, D.A., Mitchell, J.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7796, pp. 21–40. [CrossRef]
49. Grontas, P.; Pagourtzis, A.; Zacharakis, A. Security Models for Everlasting Privacy. Available online: https://eprint.iacr.org/2019/1193.pdf (accessed on 3 May 2020).
50. Cortier, V.; Galindo, D.; Küsters, R.; Müller, J.; Truderung, T. SoK: Verifiability Notions for E-Voting Protocols. In Proceedings of the IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, 22–26 May 2016; IEEE Computer Society: Piscataway, NJ, USA, 2016; pp. 779–798. [CrossRef]
51. CryptoZ. Chainanalysis—3.8 Million Bitcoin Is Lost Forever. Steemit. 2018. Available online: https://steemit.com/cryptocurrency/@crypto-z/chainanalysis-3-8-million-bitcoin-is-lost-forever (accessed on 3 May 2020).

52. Neto, A.S.; Leite, M.; Araújo, R.; Mota, M.P.; Neto, N.C.S.; Traoré, J. Usability Considerations For Coercion-Resistant Election Systems. In Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems (IHC 2018), Belém, Brazil, 22–26 October 2018. [CrossRef]
53. Marea Granate. Calendario Electoral Voto Exterior 2019. Available online: https://mareagranate.org/2019/02/calendario-electoral-voto-exterior-2019/ (accessed on 3 May 2020).