Amun: Securing E-Voting Against Over-the-Shoulder Coercion

Riccardo Longo[1][0000−0002−8739−3091] and Chiara Spadafora[2][0000−0003−3352−9210]

[1] riccardolongomath@gmail.com
[2] chiara.spadafora@unitn.it
Department of Mathematics, University Of Trento, 38123 Povo, Trento, Italy

Abstract. In an election where each voter may express P preferences among M possible choices, the Amun protocol allows to secure vote casting against over-the-shoulder adversaries, retaining privacy, fairness, end-to-end verifiability, and correctness.
Before the election, each voter receives a ballot containing valid and decoy tokens: only valid tokens contribute in the final tally, but they remain indistinguishable from the decoys. Since the voter is the only one who knows which tokens are valid (without being able to prove it to a coercer), over-the-shoulder attacks are thwarted.
We prove the security of the construction under the standard Decisional Diffie Hellman assumption in the random oracle model.

Keywords: E-Voting · Over-the-Shoulder Attack · Coercion-Resistance · End-to-End Verifiability · Diffie-Hellman Assumption · Cryptography · Formal Proof of Security.

1 Introduction

Remote voting, thanks to advanced cryptographic techniques, may guarantee higher levels of security with respect to classical paper-based voting. However, being able to vote from home is a double-edged sword: on one hand it may improve turnout by easing the voting process, on the other hand it comes at the expense of the loss of privacy that only a voting booth can guarantee.

Although many systems protect against various adversaries who try to bribe electors, we found that it is more difficult to counter opponents that closely monitor voters during the voting phase (over-the-shoulder attacks). The main mitigation technique against coercion is the usage of fake credentials [19], which are indistinguishable from real ones but that do not produce valid votes. However, if the adversary keeps under control the voter until the end of the voting period, it becomes impossible to re-vote with the valid credential.

Here we present the Amun[3] protocol, which hides the real choice expressed by a ballot even if an adversary is physically monitoring the elector during

[3] Amun was a major ancient Egyptian deity. The name Amun meant something like "the hidden one" or "invisible".

vote casting. This feature protects the elector against over-the-shoulder attacks without the need to re-vote. The Amun protocol aims to achieve end-to-end verifiability, universal verifiability, privacy, correctness, fairness, and coercion resistance.

In [35], the authors propose a a remote e-voting protocol in an election with two candidates, based on blockchain technology. The basic idea is that every voter owns two voting tokens (*v-tokens*): one is valid, the other is a decoy, but only the voter knows which is which. When voting, every voter expresses their preference by assigning their valid *v-token* to the chosen candidate and their decoy *v-token* to the other candidate. The voter gets a vote receipt on which both transactions are be displayed. In the final tally the decoy *v-tokens* do not contribute to the count and the whole process is publicly auditable. Amun is a generalization of this protocol, which adds support for multiple candidates and more than one choice (which can be exploited to express blank or partial ballots), and forsakes the blockchain infrastructure in favor of a traditional bulletin board.

In Amun, three authorities share the administration of the election: they setup the parameters, manage voters' registration, and compute the final tally at the end of the voting phase. Privacy is preserved even if an attacker colludes with one authority, limiting their power. As in [35], votes are cast by assigning some "voting tokens", generated during the registration, to the candidates. Among these tokens, only a few are valid and express the real preference of the voter, but they are indistinguishable from the other, decoy, tokens. This trick disguises the actual choice made by the elector, even if the adversary is watching.

1.1 Related Work

Protocols for electronic election systems have been abundantly proposed in recent years. Many have addressed the problem of coercion resistance, giving a plurality of definitions [13,15,19,21,22].

Civitas [7], which derives from *JCJ* [19], deals with coercion by allowing voters to vote multiple times via a mechanism of *real* and *fake* credentials.

Selene [32] associates to every vote a unique tracker: the idea is that, in case of an attack, every voter is able to open up its commitment to a fake tracker in order to deceive the attacker.

Bingo Voting [4] is an e-voting protocol that relies on a trusted random-number generator. Fake votes are generated for all the candidates the voter did not for. Every voter then receives a receipt which includes every candidate, concealing which ones they actually did vote for. Fake votes are eliminated in tallying.

Belenios [9] itself is not coercion resistant: voters can keep the randomness used to encrypt the ballot to prove how they voted. This limitation has been overcome with the deployment of *BeleniosRF* [6,8].

Caveat Coercitor [14] is a unique voting system that, instead of preventing coercion, allows it, while recording unforgeable evidence of said coercions. Observers can decide whether or not the outcome is valid based on the number of suspicious ballots.

1.2 Organization

We present some preliminaries in Section 2, in particular in Section 2.4 we describe what we mean by the term *bulletin board*. We describe our protocol in Section 3 and we provide a proof of security in Section 5. Finally, in Section 6 we draw some conclusions.

2 Preliminaries

The algebraic preliminaries we need to build the protcol are the *Decisional Diffie-Hellmann Assumption*, the *Equality of discrete logarithms ZKP* and a commitment scheme formalized in [35]. For the sake of compactness we use the following notation for the indexes: $[n] = \{i \in \mathbb{N} : 1 \leq i \leq n\}$, $(t_j)_{j \in [m]} = (t_1, \ldots, t_m)$.

2.1 Decisional Diffie-Hellman Assumption

We adopt the definition of the Decisional Diffie–Hellman (DDH) problem and the relative hardness assumption given in [25].

Let p be a prime. Let a, b, $\xi \in \mathbb{Z}_p^*$ be chosen at random and g be a generator of a cyclic group $\mathbb{G}$ of order p. The DDH problem consists in constructing an algorithm

$$\mathbb{B}\left(g, A = g^a, B = g^b, \Xi\right) \rightarrow \{0, 1\} \tag{1}$$

to distinguish between the tuples $\left(g, A, B, g^{ab}\right)$ and $\left(g, A, B, g^{\xi}\right)$, outputting respectively 1 and 0. The advantage of $\mathbb{B}$ in this case is written as:

$$Adv_{\mathbb{B}} = \left|\mathbb{P}\left[\mathbb{B}\left(g, A, B, g^{ab}\right) = 1\right] - \mathbb{P}\left[\mathbb{B}\left(g, A, B, g^{\xi}\right) = 1\right]\right|, \tag{2}$$

where the probability is taken over the random choice of the generator g, of a, b, $\xi \in \mathbb{Z}_p^*$, and the random bits possibly consumed by $\mathbb{B}$ to compute the response.

Definition 1 (DDH Assumption). *The Decisional Diffie-Hellman assumption holds if no probabilistic polynomial-time algorithm $\mathbb{B}$ has a non-negligible advantage in solving the DDH problem.*

2.2 Zero-Knowledge Proofs

A Zero-Knowledge proof (ZKP) is a cryptographic proof which allows one party (the prover $\mathcal{P}$) to convince another party (the verifier $\mathcal{V}$) about the truth of some statement, without revealing anything else to the verifier.

Given a language $\mathcal{L}$ and a common input x then the three basic properties of a ZKP are:

Definition 2 (Completeness). *If $x \in \mathcal{L}$ (i.e. the prover is honest) then the verifier should accept the proof with probability* 1.

Definition 3 (Soundness). *If $x \notin \mathcal{L}$ (i.e. the prover wants to convince the verifier to know something that it does not know or the validity of a property that is actually false) then the verifier should only accept with negligible probability.*

Definition 4 (Zero-Knowledge). *For every verifier $\mathcal{V}$ there exists an efficient simulator that can generate transcripts that are indistinguishable from real interaction between a real prover and $\mathcal{V}$.*

The third property guarantees that the verifier learns nothing from the interaction, except that $x \in \mathcal{L}$.

Equality of discrete logarithms We report here a proof of the equality of two discrete logarithms [35], which is a variation of the Schnorr interactive protocol [33,34].

Protocol 1. *Let $\mathbb{G}$ be a cyclic group of prime order p, let $u, \bar{u}$ be generators of $\mathbb{G}$, and let $z, \bar{z} \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:*

$$u^{\omega} = z \quad \text{and} \quad \bar{u}^{\omega} = \bar{z}, \tag{3}$$

without disclosing ω. The values of u, z, $\bar{u}$ and $\bar{z}$ are publicly known.

1. *The prover generates a random r and computes the commitments $t = u^r$ and $\bar{t} = \bar{u}^r$, then sends $(t, \bar{t})$ to the verifier.*
2. *The verifier generates a challenge $c \in \mathbb{Z}_p$ and sends it to the prover*[4]*.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = z^c \cdot t$, $\bar{u}^s = \bar{z}^c \cdot \bar{t}$. If the check fails the proof fails and the protocols aborts.*

The classical interactive ZKP is obtained if in step 2 c is chosen uniformly at random in $\{0, 1\}$, and the protocol is repeated $\tau = \text{poly}(\log_2(p))$ times (the number of repetitions is polynomial in the length of p, which is the security parameter). The completeness, soundness, and zero-knowledge properties of this protocol are proven in [35], where it also described as to simulate a proof and to extract the secret discrete logarithm from an adversary that we can rewind.

Further discussion on Zero-Knowledge proofs and simulations can be found in [24].

Non-Interactive proofs. In many contexts, such as public verifiability in e-voting protocols, it is necessary to prove a statement (in zero-knwoledge) to many parties, so an interactive protocol becomes quite inconvenient. In these cases it is much preferable if the prover can publish some sort of evidence of the truthfulness of the statement that can be independently verified by all the relevant parties later on. Such sort of ZKP is called Non-Interactive Zero-Knowledge

[4] Depending on how this challenge is generated, different types of ZKP can be instantiated, see comments below and the next section (Section 2.2)

Proof (NIZKP), and the Fiat-Shamir technique [17] can be used to transform an interactive sigma protocol into a NIZKP by exploiting a hash function modeled as a random oracle. The technique requires to derive the challenge value c deterministically by hashing all the public values involved in the ZKP. This method assures that $\mathcal{P}$ cannot choose the challenge before the commitments, so a single-round ZKP can be transformed in a NIZKP.

In particular, the non-interactive version of Protocol 1 proceeds as follows:

- $\mathcal{P}$ performs the first step as in the ZKP, derives $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t})$, then computes s as in the third step of the ZKP, and publishes $(u, \bar{u}, z, \bar{z}, t, \bar{t}, s)$;
- $\mathcal{V}$ computes $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t})$, then performs the checks as in the last step of the ZKP.

Designated-Verifier proofs. Designated-Verifier Non-Interactive ZKP systems (DVNIZKPs [18]) are protocols which retain most of the security properties of a NIZKP, but are not publicly verifiable: only the owner of some secret information (the designated verifier) can check the proof. This property is useful in the context of e-voting to achieve end-to-end verifiability while still preventing the voter from transferring some proofs (and thus preventing coercion through plausible deniability).

A method that can be used to build a DVNIZKP is to prove either the knowledge of a secret key or that $x \in \mathcal{L}$, with a NIZKP that assures that one of these two statements is true without revealing which one. Given two NIZKPs for the languages $\mathcal{L}_0$ and $\mathcal{L}_1$, with a challenge $c \in \mathbb{Z}_p$, the Cramer-Damgård-Schoenmakers technique [10] allows to build a NIZKP for the disjunction $\mathcal{L}_0 \vee$ $mathcal L_1$. The method exploits the ability of the prover to simulate the proof if c is known in advance, and the fact that given $c \in \mathbb{Z}_p$ you can freely choose $c_0 \in \mathbb{Z}_p$ and in consequence fix $c_1 \in \mathbb{Z}_p$ such that $c = c_0 + c_1$.

Protocol 2. *Let $\mathbb{G}$ be a cyclic group of prime order p, let $u, \bar{u}$ be generators of $\mathbb{G}$, and let $z, \bar{z} \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. Let $e \in \mathbb{Z}_p$ be the secret key of $\mathcal{V}$ and $D = u^e \in \mathbb{G}$ be the corresponding public key. As in Protocol 1, the prover knows ω and wants to convince the verifier that:*

$$u^{\omega} = z \quad \text{and} \quad \bar{u}^{\omega} = \bar{z}, \tag{4}$$

without disclosing ω. We also want $\mathcal{V}$ to be able to exploit the knowledge of e to forge such a proof for any value of $z, \bar{z}$ without knowing ω (such an ω may also not exist).

The values of u, z, $\bar{u}$, $\bar{z}$, and D are publicly known.

1. *$\mathcal{P}$ computes $t, \bar{t} \in \mathbb{G}$ as in Protocol 1;*
2. *$\mathcal{P}$ chooses uniformly at random $s_0, c_0 \in \mathbb{Z}_p$ and computes $t_0 = u^{s_0} \cdot D^{-c_0}$;*
3. *$\mathcal{P}$ computes $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D)$ and $c_1 = c - c_0$;*
4. *$\mathcal{P}$ computes $s_1 = r + c_1 \cdot \omega$ and publishes the DVNIZKP:*

$$(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D, s_0, s_1, c_0, c_1)\,.$$

$\mathcal{V}$ checks that $u^{s_1} = z^{c_1} \cdot t$, $\bar{u}^{s_1} = \bar{z}^{c_1} \cdot \bar{t}$, $u^{s_0} = D^{c_0} \cdot t_0$, and $c_0 + c_1 = c$ with $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D)$. If the check fails the proof is rejected.

$\mathcal{V}$, who knows e, can forge a proof for any $z, \bar{z} \in \mathbb{G}$ in the following way:

1. $\mathcal{V}$ chooses $r_0 \in \mathbb{Z}_p$ uniformly at random and computes $t_0 = u^{r_0}$;
2. $\mathcal{V}$ chooses uniformly at random $s_1, c_1 \in \mathbb{Z}_p$ and computes $t = u^{s_1} \cdot z^{-c_1}$, $\bar{t} = \bar{u}^{s_1} \cdot \bar{z}^{-c_1}$;
3. $\mathcal{V}$ computes $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D)$ and $c_0 = c - c_1$;
4. $\mathcal{V}$ computes $s_0 = r_0 + c_0 \cdot e$ and obtains the forged DVNIZKP:

$$(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D, s_0, s_1, c_0, c_1)\,.$$

2.3 Commitment Scheme

A commitment scheme [5] is composed by two algorithms:

- `Commit`(m, r)**:** takes the message m to commit with some random value r as input and outputs the commitment c and a decommitment value d.
- `Verify`(c, m, d)**:** takes the commitment c, the message m and the decommitment value d and outputs true if the verification succeeds, false otherwise.

A commitment scheme must have the following two properties:

- **Binding:** it is infeasible to find $m' \neq m$ and d, d' such that `Verify`$(c, m, d) =$ `Verify`$(c, m', d') =$ `true`.
- **Hiding:** Let $[c_1, d_1] =$ `Commit`(m_1, r_1) and $[c_2, d_2] =$ `Commit`(m_2, r_2) with $m_1 \neq m_2$, then it is infeasible for an attacker having only c_1, c_2, m_1 and m_2 to distinguish which c_i corresponds to which m_i.

In our construction we use commitments to prevent some possible malicious choice of parameters, specifically we want that the authorities choose their values independently. However, this kind of suspicious behavior does not affect Vote-Indistinguishability (see Definition 11) thanks to the hardness of DLOG problem, so in our analysis commitments are not directly involved in the proof of Theorem 1. For this reason we do not specify the meaning of *infeasibility* in the aforementioned security properties, noting that a commitment scheme can achieve *perfect* (information theoretic) security in only one of the two properties, while the other is at most *computationally* secure.

2.4 Bulletin Board

The concept of *(Web) Bulletin Board* (BB [20]) is well established in literature, as its use in e-voting.

A BB is a log [7] service that implements publicly readable, insert-only storage. It is often managed by the administrator of the election and relies on some security assumptions:

- it is not possible to forge messages,

– attempts to present different views of log contents to different readers should be detected.

A secure voting system should protect against a malicious administrator or bulletin board which tries to forge or unduly redact data (e.g. tries to insert arbitrary ballots or reject valid ballots). A more detailed discussion on bulletin boards can be found in [16].

2.5 General requirements for remote voting systems

A trustworthy e-voting protocol has to satisfy conflicting requirements: it should preserve both *integrity* of election results and *confidentiality* of votes. In this section we define the properties that a trustworthy e-voting protocol should fulfill. We will prove that our proposed protocol satisfies them in Section 5.2.

Definition 5 (Correctness). *Correctness [19] requires that an adversary cannot preempt, alter, or cancel the votes of honest voters, and cannot cause voters to cast ballots resulting in double voting.*

Definition 6 (Fairness). *Fairness [28] requires that no information about how many votes each candidate has received can be learned until the voting results are published. Any participant cannot gain knowledge of the voting results before its final publication.*

Definition 7 (Privacy). *Privacy ([29,23,3]) is defined as the inability of the adversary to distinguish, given two candidates C_1, C_2, whether voter V_i voted for C_1 or C_2.*

Definition 8 (Verifiability). *Verifiability [1,19,6] requires that the results of tabulation cannot be different than if all votes were announced and tabulated publicly (even if an adversary tries to change the election result). Verifiability can be divided [1] into:*

- *Universal Verifiability: the correctness of elections results can be verified by all observers;*
- *Individual Verifiability: every voter can check that their vote has been cast correctly and has been accurately counted.*
  - *Cast-as-intended verifiability [11]: every voter can check that their vote was correctly cast.*
  - *Recorded-as-cast verifiability [27]: every voter can check that their vote was recorded as it was cast.*
  - *Tallied-as-recorded verifiability [30]: anyone can check that cast votes were correctly tallied.*

  In [2], the combination of cast-as-intended, recorded-as-cast, and tallied-as-recorded, is called End-to-end*.*

Coercion resistance ([19,15]) requires that an adversary cannot learn any additional information about the votes other than what is revealed by the results of tabulation. In other words, voters cannot prove whether or how they voted, even if they can interact with the adversary while voting. In [15] there is a critical analysis of various definitions.

The Amun protocol protects against coercers that wish to sway elections towards specific candidates, but is not very effective against the more subtle randomization and forced abstention attacks. In this simplified model, we adapt the definition of Coercion Resistance as follows:

Definition 9 (Vote-Coercion Resistance). *Let $\mathcal{A}$ be a coercer, V_c the set of coerced voters, and $(C_{i,1}, \ldots, C_{i,P})$ the choices that $\mathcal{A}$ wants to impose to the voter corresponding to $v_i \in V_c$. Let Ψ_1 be the scenario in which $\mathcal{A}$ has access only to the final tally. Let Ψ_2 be the scenario in which $\mathcal{A}$ has access to the whole Bulletin Board, and can see all the actions performed by the voters in V_c, with the exception of the ones carried out in a protected environment (or through an untappable channel). A voting protocol is* Vote-Coercion Resistant *if the probability of $\mathcal{A}$ detecting that a voter in V_c has not followed its instruction is the same in Ψ_1 and Ψ_2.*

3 Multi-Candidate Voting System

This section presents our proposal for a remote e-voting protocol that manages an election with N voters, where each one expresses P preferences among M candidates (obviously $P < M$).

The basic idea is that every voter owns M voting tokens (*v-tokens*): P are valid, the others are a decoy, but only the voter knows which is which. When voting, voters express their preferences assigning the valid *v-tokens* to the chosen candidates and the decoy ones to the others.

The protocol allows for re-voting, before tallying duplicate ballots (i.e. ballots with the same *v-tokens* regardless of their order) are discarded, keeping only the most recent. After the voting phase, when counting the votes, the decoy *v-tokens* do not contribute to the tally, so only valid *v-tokens* are counted. The whole process is publicly auditable and fully verifiable, and preserves privacy as long as at most one authority is corrupt.
The protocol is divided into four phases:

- **Setup.** Three authorities, knowing a list of eligible voters, generate the values for the creation of both the *v-tokens* and the masks associated to the candidates. These masks guarantee the voters' privacy, and prevent early tallying.
- **Registrar Phase.** In this phase, the three authorities engage in a 5-step protocol (see Figure 1) to create M indistinguishable *v-tokens* (P are valid and $M - P$ are a decoy) employing masking and shuffling so that at the end the authorities will not be able to identify which tokens are valid. The voter can check the validity of these *v-tokens* thanks to DVNIZKPs issued by the

authorities. These proofs are worthless for a coercer because the voter can forge them.

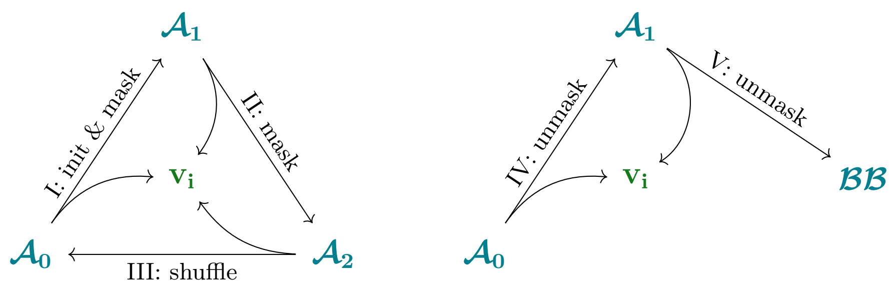

Fig. 1. The main steps of the ballot generation procedure, which correspond to steps 3-7 of the Registrar Phase as described in Section 3.1.

- **Voting Phase.** During this phase the voter express their preferences by assigning each of their M *v-tokens* to the candidates. All *v-tokens* of a voter must be assigned together, each to a distinct candidate. After the *v-tokens* have been assigned, the voter gets a transcript that reports the assignment of the *v-tokens* to the candidates. This transcript is worthless for a coercer since the *v-tokens* are indistinguishable. Here we assume that every candidate receives at least one legitimate vote (with a valid *v-token*), otherwise it is trivial to discern the validity of some tokens from the election results.
- **Tallying.** The *v-tokens* are processed (see Figure 2), removing the candidate masks, which allows to count the number of valid and decoy tokens assigned to each candidate. The results and the intermediate computation steps are published, alongside a set of NIZKPs that allow anyone to check that the results are correct and there has not been any manipulation of the ballots. Every voter can also check, by examining the bulletin board, that their *v-tokens* have been cast and counted correctly.

3.1 Protocol Description

The key components involved in the protocol are:

1. a finite set of voters $V = \{v_i\}_{i\in[N]}$ (where v_i is a pseudonymous id), with $N \in \mathbb{N}$ the number of eligible voters;
2. a finite set of candidates $C = \{c_\ell\}_{\ell\in[M]}$ with $M \in \mathbb{N}$ the number of candidates;
3. three trusted authorities[5] $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$.

[5] We use a weak concept of trust here, since the conduct of these authorities can be checked by voters.

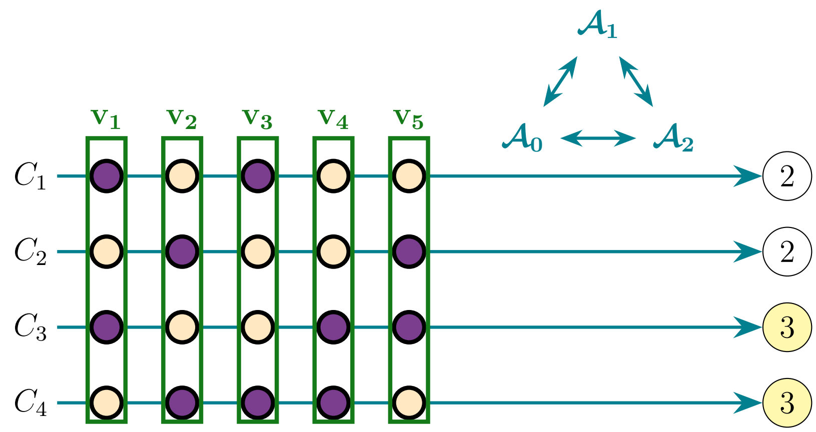

Fig. 2. Example of voting and tallying. Each voter has two valid tokens ● and two decoy tokens ●. After the tallying it is revealed that candidates C_3 and C_4 are elected having received more preferences (3) with respect to the other two candidates (who received only 2).

4. one ballot b_i (comprising M *v-tokens*) for every $i \in [N]$, i.e. one for each eligible voter.

Throughout the protocol we implicitly assume that every public value (including a description of the key components presented above) are published in the BB. The protocol is divided into four phases:

Setup The authority $\mathcal{A}_0$ selects and publishes:

1. a secure group $\mathbb{G}$ of prime order p in which the DDH assumption (see Definition 1) holds;
2. a generator $g \in \mathbb{G}$;
3. a commitment scheme (see Section 2.3) `Comm` to be used to commit to the values computed before publishing them, in order to improve security.

Then $\mathcal{A}_0$ performs the following operations:

1. chooses uniformly at random two values k and λ in $\mathbb{Z}_p^*$. $\mathcal{A}_0$ knows that the *v-tokens* computed using k are valid, while the ones computed using λ are decoys, but this information is kept secret;
2. chooses uniformly at random $N \cdot M$ distinct values $\bar{z}_{i,\ell} \in \mathbb{Z}_p^*$, with $i \in [N]$, $\ell \in [M]$;
3. finally, $\mathcal{A}_0$ commits to the values g^k, g^λ, and, for every $i \in [N]$, it commits to $\left(v_i, (g^{\bar{z}_{i,\ell}})_{\ell \in [M]}\right)$.

An honest authority $\mathcal{A}_0$ is supposed to keep private all the values $\bar{z}_{i,\ell}$, k, λ.

The authority $\mathcal{A}_1$ performs the following operations:

1. chooses uniformly at random M distinct values $\alpha'_\ell \in \mathbb{Z}_p^*$, with $\ell \in [M]$, these will be the first half of the candidates' masks;
2. chooses uniformly at random N distinct values $x'_i \in \mathbb{Z}_p^*$, with $i \in [N]$;
3. chooses uniformly at random two sets of $N \cdot M$ distinct values $z'_{i,\ell}, y'_{i,\ell} \in \mathbb{Z}_p^*$, with $i \in [N], \ell \in [M]$;
4. finally, $\mathcal{A}_1$ commits to the values $g^{\alpha'_\ell}$, $\forall \ell \in [M]$, and for every $i \in [N]$ it commits to the tuple $\left(v_i, g^{x'_i}, (g^{z'_{i,\ell}})_{\ell\in[M]}, (g^{y'_{i,\ell}})_{\ell\in[M]}\right)$.

An honest authority $\mathcal{A}_1$ is supposed to keep private all the values α'_ℓ, x'_i, $z'_{i,\ell}$, $y'_{i,\ell}$.

The authority $\mathcal{A}_2$ performs the following operations:

1. chooses uniformly at random M distinct values $\alpha''_\ell \in \mathbb{Z}_p^*$, with $\ell \in [M]$, these will be the second half of the candidates' masks;
2. chooses uniformly at random N distinct values $x''_i \in \mathbb{Z}_p^*$, with $i \in [N]$;
3. chooses uniformly at random $N \cdot M$ distinct values $y''_{i,\ell} \in \mathbb{Z}_p^*$, with $i \in [N]$, $\ell \in [M]$;
4. Finally $\mathcal{A}_2$ commits to the values $g^{\alpha''_\ell}$, $\forall \ell \in [M]$, and for every $i \in [N]$ it commits to the tuple $\left(v_i, g^{x''_i}, (g^{y''_{i,\ell}})_{\ell\in[M]}\right)$.

An honest authority $\mathcal{A}_2$ is supposed to keep private all the values α''_ℓ, x''_i, $y''_{i,\ell}$.

Once that all the commitments have been published, the authorities can decommit the values:

- $\mathcal{A}_0$ publishes the decommitments for the values g^k, g^λ, alongside all the tuples $\left(v_i, (g^{\bar{z}_{i,\ell}})_{\ell\in[M]}\right) \forall i \in [N]$;
- $\mathcal{A}_1$ publishes the decommitments for the values $g^{\alpha'_\ell} \forall \ell \in [M]$, and the tuples $\left(v_i, g^{x'_i}, (g^{z'_{i,\ell}})_{\ell\in[M]}, (g^{y'_{i,\ell}})_{\ell\in[M]}\right) \forall i \in [N]$;
- $\mathcal{A}_2$ publishes the decommitments for the values $g^{\alpha''_\ell} \forall \ell \in [M]$, and the tuples $\left(v_i, g^{x''_i}, (g^{y''_{i,\ell}})_{\ell\in[M]}\right) \forall i \in [N]$.

All these published values are accompanied by NIZKPs which prove that the authority who published them knows the corresponding secret exponents. These NIZKPs can be constructed using the Schnorr protocol [33] and the Fiat-Shamir transformation [17] just like in Section 2.2.

To simplify notation we introduce the following definitions for aggregate values for all $i \in [N]$ and $\ell \in [M]$:

$$x_i = x'_i + x''_i, \qquad \alpha_\ell = \alpha'_\ell \cdot \alpha''_\ell, \qquad z_{i,\ell} = \bar{z}_{i,\ell} \cdot z'_{i,\ell}, \qquad y_{i,\ell} = y'_{i,\ell} \cdot y''_{i,\ell}. \quad (5)$$

Registrar Phase For every pseudonymous id $v_i \in V$ the following steps are performed:

1. Let Alice be the person associated to the pseudonymous id v_i, note that the authorities do not need to know this association. She goes in a safe and controlled environment (see Section 6 for further discussion on this requirement) where she is identified and authenticated as the eligible and not yet registered pseudonymous id v_i. In this environment she can interact with all three authorities without fear that any adversary can eavesdrop or interfere.
2. Alice creates a signing key-pair (s_i, K_i), a designated verifier key-pair (e_i, D_i), and gives K_i, D_i to the authorities proving the knowledge of s_i (e.g. by signing a challenge message), and of e_i via a NIZKP (which includes the challenge message among the public values). The authorities associate K_i, D_i to v_i in their respective voters lists.
3. $\mathcal{A}_0$ performs the following steps:
   (a) $\mathcal{A}_0$ chooses, for every $i \in [N]$, a random subset $V_i \subset [M]$ such that its cardinality is exactly P, then sets:

$$\sigma_{i,\ell} = \begin{cases} k \iff \ell \in V_i \\ \lambda \iff \ell \notin V_i \end{cases} \tag{6}$$

   i.e. the random choice of the V_i determines which tokens will be valid and which will be a decoy;
   (b) $\mathcal{A}_0$ takes the (publicly available) values $g^{x'_i}$ and $g^{x''_i}$ and creates the step 0 of the ballot $\bar{b}_{0,i} = (\bar{b}_{0,i,\ell})_{\ell \in [M]}$ where:

$$\bar{b}_{0,i,\ell} = \left(g^{\sigma_{i,\ell}} \cdot g^{x'_i} \cdot g^{x''_i}\right)^{\bar{z}_{i,\ell}} = g^{\bar{z}_{i,\ell}(\sigma_{i,\ell}+x_i)} \qquad \forall \ell \in [M]; \tag{7}$$

   (c) $\mathcal{A}_0$ sends to $\mathcal{A}_1$ the initial ballot $\bar{b}_{0,i}$ and sends to Alice $\bar{b}_{0,i}$ and V_i;
   (d) $\mathcal{A}_0$ proves the correctness of computations with multiple instances of the DVNIZKP of Protocol 2:
      i. $\mathcal{A}_0$ proves that the $g^{\bar{z}_{i,\ell}\sigma_{i,\ell}}$ are correct (using $\sigma_{i,\ell} = k$ or $\sigma_{i,\ell} = \lambda$) with:

$$\omega = k, \quad u = g, \quad z = g^k, \quad \bar{u} = g^{\bar{z}_{i,\ell}}, \quad \bar{z} = g^{\bar{z}_{i,\ell}k} \qquad \forall \ell \in V_i, \tag{8}$$

$$\omega = \lambda, \quad u = g, \quad z = g^\lambda, \quad \bar{u} = g^{\bar{z}_{i,\ell}}, \quad \bar{z} = g^{\bar{z}_{i,\ell}\lambda} \qquad \forall \ell \in [M] \setminus V_i, \tag{9}$$

      ii. then $\mathcal{A}_0$ proves that the $\bar{b}_{0,i,\ell}$ are correct using for all $\ell \in [M]$:

$$\omega = \bar{z}_{i,\ell}, \quad u = g, \quad z = g^{\bar{z}_{i,\ell}}, \quad \bar{u} = g^{\sigma_{i,\ell}} \cdot g^{x'_i} \cdot g^{x''_i}, \quad \bar{z} = \bar{b}_{0,i,\ell}. \tag{10}$$

4. $\mathcal{A}_1$ computes the step 1 of the ballot $\bar{b}_{1,i} = (\bar{b}_{1,i,\ell})_{\ell \in [M]}$ where:

$$\bar{b}_{1,i,\ell} = \left(\bar{b}_{0,i,\ell}\right)^{z'_{i,\ell}} = g^{z_{i,\ell}(\sigma_{i,\ell}+x_i)} \qquad \forall \ell \in [M] \tag{11}$$

   and sends it to Alice and to $\mathcal{A}_2$. Then $\mathcal{A}_1$ proves that the $\bar{b}_{1,i,\ell}$ are correct with the DVNIZKP of Protocol 2, using:

$$\omega = z'_{i,\ell}, \quad u = g, \quad z = g^{z'_{i,\ell}}, \quad \bar{u} = \bar{b}_{0,i,\ell}, \quad \bar{z} = \bar{b}_{1,i,\ell} \qquad \forall \ell \in [M]. \tag{12}$$

5. $\mathcal{A}_2$ chooses uniformly at random a permutation $\pi_i \in \mathrm{Sym}([M])$ and computes the step 2 of the ballot $\bar{b}_{2,i} = (\bar{b}_{2,i,\ell})_{\ell\in[M]}$ where:

$$\bar{b}_{2,i,\ell} = \left(\bar{b}_{1,i,\ell}\right)^{y''_{i,\pi_i^{-1}(\ell)}} = g^{z_{i,\ell} y''_{i,\pi_i^{-1}(\ell)} (\sigma_{i,\ell}+x_i)} \qquad \forall \ell \in [M] \tag{13}$$

and sends it to Alice and to $\mathcal{A}_0$, π_i is sent to Alice and $\mathcal{A}_1$. Then $\mathcal{A}_2$ proves that the $\bar{b}_{2,i,\ell}$ are correct with the DVNIZKP of Protocol 2, using:

$$\omega = y''_{i,\pi_i^{-1}(\ell)},\ \ u = g,\ \ z = g^{y''_{i,\pi_i^{-1}(\ell)}},\ \ \bar{u} = \bar{b}_{1,i,\ell},\ \ \bar{z} = \bar{b}_{2,i,\ell} \quad \forall \ell \in [M]. \tag{14}$$

6. $\mathcal{A}_0$ computes the step 3 of the ballot $\bar{b}_{3,i} = (\bar{b}_{3,i,\ell})_{\ell\in[M]}$ where:

$$\bar{b}_{3,i,\ell} = \left(\bar{b}_{2,i,\ell}\right)^{\frac{1}{\bar{z}_{i,\ell}}} = g^{z'_{i,\ell} y''_{i,\pi_i^{-1}(\ell)} (\sigma_{i,\ell}+x_i)} \qquad \forall \ell \in [M] \tag{15}$$

and sends it to Alice and to $\mathcal{A}_1$. Then $\mathcal{A}_0$ proves that the $\bar{b}_{3,i,\ell}$ are correct with the DVNIZKP of Protocol 2, using:

$$\omega = \frac{1}{\bar{z}_{i,\ell}}, \quad u = g^{\bar{z}_{i,\ell}}, \quad z = g, \quad \bar{u} = \bar{b}_{2,i,\ell}, \quad \bar{z} = \bar{b}_{3,i,\ell} \qquad \forall \ell \in [M]. \tag{16}$$

7. $\mathcal{A}_1$ computes the final ballot $b_i = (b_{i,\ell})_{\ell\in[M]}$ where:

$$b_{i,\ell} = \left(\bar{b}_{3,i,\pi_i(\ell)}\right)^{\frac{y'_{i,\ell}}{z'_{i,\pi_i(\ell)}}} = g^{y_{i,\ell}(\sigma_{i,\pi_i(\ell)}+x_i)} \qquad \forall \ell \in [M] \tag{17}$$

and sends it to Alice and publishes on the BB the pair (K_i, b_i). Then $\mathcal{A}_1$ proves that the $b_{i,\ell}$ are correct with the DVNIZKP of Protocol 2 and using:

$$\omega = \frac{y'_{i,\ell}}{z'_{i,\pi_i(\ell)}},\ u = g^{z'_{i,\pi_i(\ell)}},\ z = g^{y'_{i,\ell}},\ \bar{u} = \bar{b}_{3,i,\pi_i(\ell)},\ \bar{z} = b_{i,\ell} \quad \forall \ell \in [M]. \tag{18}$$

Note that Alice, thanks to the proofs and the knowledge of the intermediate values, knows which ones are a valid token (the ones with $\sigma_{i,\ell} = k$), but thanks to the random choices of V_i and π_i the authorities cannot distinguish the tokens unless they collude. Moreover the properties of the DVNIZKP allow Alice to forge the transcript changing which tokens are valid, making them useless for proving the validity of a token.

Voting Phase Voters express their preferences by assigning the valid tokens to their chosen candidates, and the decoy tokens to the others. This assignment is then signed by the voter associated to v_i with their private signing key s_i, and published on the BB, so voters can check that their votes have been correctly registered.

Once the voting phase ends, the ballots are filtered, removing:

- duplicates (i.e. ballots which are permutations of the same values), keeping only the most recent tuple;

– incomplete ballots;
– forged ballots:
  - ballots which have not been published by the authorities on the BB during the registration phase;
  - ballots which are published on the BB (during the voting phase) without a valid signature. A ballot b_i has a valid signature if it verifies with the public key K_i which has been associated during the registration phase to b_i.

The remaining ballots are used in the tallying. Note that anyone can perform this filtering step since it only involves public data.

Incomplete and forged ballots could also be rejected and not published by the BB during the voting phase to minimize the number of processed ballots.

Tallying Once the voting phase is over, the tallying can start.

In order to count the votes, the authorities have to process the tokens received by each candidate, substituting the *voter's masks* $y_{i,\ell}$ with the appropriate *candidate mask* α_ℓ. Suppose that $T \leq N$ participants voted. Without loss of generality, we can assume that only the participants with index $i \in [T]$ voted, while the remaining $N - T$ abstained from voting.

For every $i \in [T]$, let $\phi_i : [M] \longrightarrow [M]$ be the bijective map that associates to each candidate index ℓ the index of the token $b_{i,\phi_i(\ell)}$ that the voter associated to v_i sent to the candidate C_ℓ. Then, for every $i \in [T], \ell \in [M]$, the authorities process the token $b_{i,\phi_i(\ell)}$ by performing the following steps:

1. $\mathcal{A}_1$ computes and publishes the preliminary vote $\bar{t}_{\ell,i}$ as:

$$\bar{t}_{\ell,i} = \left(b_{i,\phi_i(\ell)}\right)^{\frac{\alpha'_\ell}{y'_{i,\phi_i(\ell)}}} = g^{\alpha'_\ell y''_{i,\phi_i(\ell)}(\sigma_{i,\pi_i(\phi_i(\ell))}+x_i)}, \tag{19}$$

   alongside a NIZKP that proves this computation correct. $\mathcal{A}_1$ proves that $\bar{t}_{\ell,i}$ is correct with the NIZKP version of Protocol 1 and using:

$$\omega = \frac{\alpha'_\ell}{y'_{i,\phi_i(\ell)}}, \quad u = g^{y'_{i,\phi_i(\ell)}}, \quad z = g^{\alpha'_\ell}, \quad \bar{u} = b_{i,\phi_i(\ell)}, \quad \bar{z} = \bar{t}_{\ell,i}. \tag{20}$$

2. $\mathcal{A}_2$ then computes and publishes the final vote $t_{\ell,i}$ as:

$$t_{\ell,i} = (\bar{t}_{\ell,i})^{\frac{\alpha''_\ell}{y''_{i,\phi_i(\ell)}}} = g^{\alpha_\ell(\sigma_{i,\pi_i(\phi_i(\ell))}+x_i)}, \tag{21}$$

   alongside a NIZKP that proves this computation correct. $\mathcal{A}_2$ proves that $t_{\ell,i}$ is correct with the NIZKP version of Protocol 1 and using:

$$\omega = \frac{\alpha''_\ell}{y''_{i,\phi_i(\ell)}}, \qquad u = g^{y''_{i,\phi_i(\ell)}}, \qquad z = g^{\alpha''_\ell}, \qquad \bar{t} = b_{\ell,i}, \qquad \bar{z} = t_{\ell,i}. \tag{22}$$

Once that all final votes have been computed, the actual tallying is performed.

Let R_ℓ be the number of valid tokens given to the ℓ-th candidate (i.e. the number of preferences received by said candidate), and let F_ℓ be the number of decoy tokens given to the ℓ-th candidate. Clearly $T = R_\ell + F_\ell \quad \forall \ell \in [M]$. The count R_ℓ can be computed with the following steps:

1. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ can compute g^{α_ℓ} (as $(g^{\alpha''_\ell})^{\alpha'_\ell}$ and $(g^{\alpha'_\ell})^{\alpha''_\ell}$ respectively). $\mathcal{A}_1$ can prove the correctness of this value by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = \alpha'_\ell, \qquad u = g, \qquad z = g^{\alpha'_\ell}, \qquad \bar{u} = g^{\alpha''_\ell}, \qquad \bar{z} = g^{\alpha_\ell}, \tag{23}$$

   $\mathcal{A}_2$ can prove the correctness of this value by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = \alpha''_\ell, \qquad u = g, \qquad z = g^{\alpha''_\ell}, \qquad \bar{u} = g^{\alpha'_\ell}, \qquad \bar{z} = g^{\alpha_\ell}. \tag{24}$$

   In practice, each authority may publish half of the values.
2. $\mathcal{A}_0$ computes and publishes $g^{\alpha_\ell k} = (g^{\alpha_\ell})^k$ and $g^{\alpha_\ell \lambda} = (g^{\alpha_\ell})^\lambda$. Then $\mathcal{A}_0$ proves that $g^{\alpha_\ell k}$ is correct by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = k, \qquad u = g, \qquad z = g^k, \qquad \bar{u} = g^{\alpha_\ell}, \qquad \bar{z} = g^{\alpha_\ell k}, \tag{25}$$

   and that $g^{\alpha_\ell \lambda}$ is correct by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = \lambda, \qquad u = g, \qquad z = g^\lambda, \qquad \bar{u} = g^{\alpha_\ell}, \qquad \bar{z} = g^{\alpha_\ell \lambda}. \tag{26}$$

3. $\mathcal{A}_1$ computes $\sum_{i=1}^T x'_i$, and publishes $g^{\alpha_\ell \sum_{i=1}^T x'_i}$. Then $\mathcal{A}_1$ proves that $g^{\alpha_\ell \sum_{i=1}^T x'_i}$ is correct by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = \sum_{i=1}^T x'_i, \quad u = g, \quad z = g^{\sum_{i=1}^T x'_i}, \quad \bar{u} = g^{\alpha_\ell}, \quad \bar{z} = g^{\alpha_\ell \sum_{i=1}^T x'_i}, \tag{27}$$

   noting that any observer can compute $g^{\sum_{i=1}^T x'_i} = \prod_{i=1}^T g^{x'_i}$.
4. Similarly, $\mathcal{A}_2$ computes $\sum_{i=1}^T x''_i$ and publishes $g^{\alpha_\ell \sum_{i=1}^T x''_i}$. Then $\mathcal{A}_2$ proves that $g^{\alpha_\ell \sum_{i=1}^T x''_i}$ is correct by publishing a NIZKP (from Protocol 1) computed using:

$$\omega = \sum_{i=1}^T x''_i, \qquad u = g, \quad z = g^{\sum_{i=1}^T x''_i}, \quad \bar{u} = g^{\alpha_\ell}, \quad \bar{z} = g^{\alpha_\ell \sum_{i=1}^T x''_i}, \tag{28}$$

   noting that any observer can compute $g^{\sum_{i=1}^T x''_i} = \prod_{i=1}^T g^{x''_i}$.

5. Given that any observer can compute the value:

$$g^{\alpha_\ell(\sum_{i=1}^T x_i + R_\ell k + F_\ell \lambda)} = \prod_{i=1}^T t_{\ell,i}, \tag{29}$$

and that:

$$g^{\alpha_\ell \sum_{i=1}^T x_i} = g^{\alpha_\ell \sum_{i=1}^T (x_i' + x_i'')} = g^{\alpha_\ell \sum_{i=1}^T x_i'} \cdot g^{\alpha_\ell \sum_{i=1}^T x_i''}, \tag{30}$$

then anyone can compute:

$$\mathfrak{T} = \left(g^{\alpha_\ell \sum_{i=1}^T x_i}\right)^{-1} \cdot g^{\alpha_\ell(\sum_{i=1}^T x_i + R_\ell k + F_\ell \lambda)} = \left(g^{\alpha_\ell k}\right)^{R_\ell} \cdot \left(g^{\alpha_\ell \lambda}\right)^{F_\ell}. \tag{31}$$

6. R_ℓ and F_ℓ can now be computed by brute force, giving the number of preferences received by the ℓ-th candidate.

Example 1. Let $T = 3$, then the possible values of $\mathfrak{T}$ are:

$$\mathfrak{T} = \begin{cases} \left(g^{\alpha_\ell k}\right)^0 \cdot \left(g^{\alpha_\ell \lambda}\right)^3 & \text{if} \quad R_\ell = 0, \quad F_\ell = 3, \\ \left(g^{\alpha_\ell k}\right)^1 \cdot \left(g^{\alpha_\ell \lambda}\right)^2 & \text{if} \quad R_\ell = 1, \quad F_\ell = 2, \\ \left(g^{\alpha_\ell k}\right)^2 \cdot \left(g^{\alpha_\ell \lambda}\right)^1 & \text{if} \quad R_\ell = 2, \quad F_\ell = 1, \\ \left(g^{\alpha_\ell k}\right)^3 \cdot \left(g^{\alpha_\ell \lambda}\right)^0 & \text{if} \quad R_\ell = 3, \quad F_\ell = 0. \end{cases}$$

Since $g^{\alpha_\ell k}$ and $g^{\alpha_\ell \lambda}$ are publicly available, all these possible values can be computed by anyone, identifying the values of R_ℓ and F_ℓ.

Given a positive integer $T \in \mathbb{N}$, it is possible to represent it in $T + 1$ ways as a sum of two non-negative integers. Given that the number of valid and decoy votes must sum up to the number of actual voters T, it follows that the number of possible values for $\mathfrak{T}$ is $T + 1$, so the effort of computing R_ℓ and F_ℓ is linear in the number of actual votes.

4 Usability

In order to cast a vote, the voter has to remember which are the P valid *v-tokens* among the M in their ballot. This can be an usability issue when P and M grow.

To help the voter remembering the position of the valid tokens, we can exploit error correcting codes. We can see the information on which tokens are valid as a binary vector of $\mathbb{F}_2^M$ with constant weight P. We can exploit constant-weight codes [12] to encode these vectors as a vector of the space $\mathbb{F}_q^\varkappa$ and then use a $[n, \varkappa]_q$ shortened Reed-Solomon code [31] to add error-correction capabilities. With this approach the voter has only to remember n elements of $\mathbb{F}_q$, with the added bonus that up to $\frac{n-\varkappa}{2}$ errors can be automatically corrected.

Example 2. We can encode the information about which tokens are valid with a 6 digits PIN that corrects up to two errors (and therefore also an inversion of two digits, which is a fairly common error). To do so we set $q = 9$, $n = 6$, $\varkappa = 2$. With this encoding we can cover any value of P if $M \leq 8$, and values of $P \leq 3$ or $P \geq M - 3$ if $M \leq 13$ $\left(\text{since in these cases } \binom{M}{P} < 9^2\right)$.

Example 3. With a short 3-letter sequence (case-insensitive) that automatically corrects one error, we can encode the information about which tokens are valid for $M \leq 6$, $M = 7 \wedge P \leq 2$, $M \leq 25 \wedge (P = 1 \vee P = M - 1)$, by setting $q = 25$, $n = 3$, $\varkappa = 1$. Adding a letter to the sequence ($n = 4$, $\varkappa = 2$), we can cover more cases:

$$\begin{aligned}
&M \leq 11, \\
&M = 12 \wedge (P \leq 4 \vee P \geq 8), \\
&M \leq 16 \wedge (P \leq 3 \vee P \geq M - 3), \\
&\quad\vdots
\end{aligned}$$

Finally, we highlight that, with the DVNIZKPs and the permutation received during the registrar phase, the voter can check whether they remember correctly the positions of the valid tokens.

5 Security Analysis

The goal is to prove that an adversary cannot distinguish between valid and decoy *v-tokens* and guess how voters cast their preferences. Since election results are obviously public, we have to avoid some trivial cases in which the adversary can deduce the votes by simply observing the results.

Therefore we assume that the adversary controls one authority and all but two voters, and that these two voters express distinct preferences. In particular, we let the adversary select two distinct sets of preferences, then we randomly assign to each of the two uncorrupted voters one set of these sets of preferences. The adversary wins the security game if it guesses correctly which voter expressed which set of preferences, i.e. guesses the random assignment.

5.1 Security Model

The security of the protocol will be proven in terms of vote indistinguishability (VI), as detailed in Definition 11.

The security of the protocol will be proven in the presence of a malicious authority, so the simulator in the proof will take on the roles of the two honest authorities and of the two voters that the adversary does not control.

To simplify our analysis we assume that the adversary-controlled authority does not intentionally fail decommitments or (DV)NIZKPs, so the protocol does not abort. This is a reasonable assumption considering the application context, however it is not necessary to attain security. In fact, if the adversary wins the security game with non-negligible advantage, then it must run the protocol smoothly with non-negligible probability (since it outputs its guess only once the protocol has correctly terminated).

Definition 10 (Security Game). *The security game for the election protocol proceeds as follows:*

- ***Init.*** *The adversary $\mathcal{A}$ chooses the authority and the $N-2$ voters that it will control. This means that the adversary knows which are the valid and decoy* v-tokens *of these voters. The remaining two voters are called* free voters*. The challenger $\mathcal{C}$ takes the role of the other authorities and the free voters.*
- ***Phase 0.*** *$\mathcal{A}$ and $\mathcal{C}$ run the* Setup *and* Registrar *phases of the protocol, interacting as needed.*
- ***Phase 1.*** *The adversary votes with some or all of the voters it controls.*
- ***Challenge.*** *The challenge phase is articulated as follows:*
  1. *$\mathcal{A}$ selects two distinct sets of preferences $\tilde{P}_0 \neq \tilde{P}_1$, with $\tilde{P}_i \subset [M]$, $\#\tilde{P}_i = P$ for $i = 0, 1$, and sends them to $\mathcal{C}$;*
  2. *$\mathcal{C}$ flips a random coin $\mu \in \{0, 1\}$ to determine which preference set the first free voter will use, i.e. $P_1 = \tilde{P}_\mu$, setting also $P_2 = \tilde{P}_{\mu \oplus 1}$;*
  3. *$\mathcal{C}$ constructs two random ballot assignment maps $\tilde{\phi}_1, \tilde{\phi}_2 : [M] \longrightarrow [M]$ such that $\tilde{\phi}_i(\ell)$ refers to a valid token if and only if $\ell \in P_i$, for $i = 1, 2$;*
  4. *finally, $\mathcal{C}$ votes by sending to the candidate C_ℓ the $\tilde{\phi}_1(\ell)$-th token of the first free voter and the $\tilde{\phi}_2(\ell)$-th token of the second free voter, $\forall \ell \in [M]$.*

  .
- ***Phase 2.*** *The adversary votes with some or all of the voters it controls.*
- ***Phase 3.*** *$\mathcal{A}$ and $\mathcal{C}$ run the* Tallying *phase of the protocol, and the election result is published.*
- ***Guess.*** *The adversary outputs a guess μ' of the coin flip that randomly assigned the voting preferences of the two free voters.*

$\mathcal{A}$ wins if $\mu' = \mu$.

Definition 11 (Vote Indistinguishability). *An E-Voting Protocol with security parameter θ is VI-secure if, for every probabilistic polynomial-time adversary $\mathcal{A}$ that outputs a guess μ' of the coin flip μ (as described in the security game of Definition 10), there exists a negligible function η such that:*

$$\mathbb{P}[\mu' = \mu] \leq \frac{1}{2} + \eta(\theta). \tag{32}$$

In the following theorem we prove our voting protocol VI-secure under the DDH assumption in the security game defined above.

Theorem 1. *In the random oracle model, if the DDH assumption holds, then the protocol described in Section 3.1 is VI-secure, as per Definition 11.*

Proof. Suppose there exists a polynomial time adversary $\mathcal{A}$, that can attack the scheme with advantage ε. We claim that a simulator $\mathcal{S}$ can be built to play the decisional DH game with advantage $\frac{\varepsilon}{2}$. The simulator controls the random oracle that defines the hash function H, and starts by taking in a DDH challenge:

$$(g, A = g^a, B = g^b, \varXi), \tag{33}$$

with $\varXi = g^{ab}$ or $\varXi = R = g^{\xi}$.

First we consider the case in which the adversary controls $\mathcal{A}_0$, where the simulation proceeds as follows.

- **Init**. The adversary chooses the $N-2$ voters to control. Without loss of generality we may assume that the two free voters are associated to v_1 and v_2.
- **Setup**. $\mathcal{S}$ chooses uniformly at random in $\mathbb{Z}_p^*$ the values $\tilde{x}_i$, $\tilde{\alpha}_\ell$, $\tilde{y}_{i,\ell}$, and $\tilde{z}_{i,\ell}$ for all $i \in [2]$, $\ell \in [M]$, and implicitly sets for all $i \in [2]$, $\ell \in [M]$:

$$x_i'' = \tilde{x}_i + (-1)^i b, \qquad \alpha_\ell' = a \cdot \tilde{\alpha}_\ell, \qquad y_{i,\ell}' = a \cdot \tilde{y}_{i,\ell}, \qquad z_{i,\ell}' = a \cdot \tilde{y}_{i,\ell}. \tag{34}$$

  $\mathcal{S}$ chooses the other values for authorities $\mathcal{A}_1$ and $\mathcal{A}_2$ following the protocol. In the improbable case that $a = 0$, the DDH problem is trivially solvable ($g^a = g^{ab} = 1$). If $a \neq 0$, since a and b come from an uniform distribution, then also these implicit values are uniformly distributed, so the choices of the simulator are indistinguishable from a real protocol execution.
  Note that $\mathcal{S}$ can compute all the values $g^{x_i''}$, $g^{\alpha_\ell'}$, $g^{y_{i,\ell}'}$, $g^{z_{i,\ell}'}$, either normally (when the parameter has been explicitly chosen) or as follows:

$$g^{x_i''} = g^{\tilde{x}_i} \cdot B^{(-1)^i}, \qquad g^{\alpha_\ell'} = A^{\tilde{\alpha}_\ell}, \qquad g^{y_{i,\ell}'} = A^{\tilde{y}_{i,\ell}}, \qquad g^{z_{i,\ell}'} = A^{\tilde{z}_{i,\ell}}. \tag{35}$$

  for all $i \in [2]$, $\ell \in [M]$. Therefore, $\mathcal{S}$ can simulate the setup phase, exploiting the random oracle to simulate the NIZKPs for x_i'', α_ℓ', $y_{i,\ell}'$ $z_{i,\ell}'$ for $i \in [2]$, $\ell \in [M]$.
- **Registrar Phase**. For the voters associated to v_i with $3 \leq i \leq N$, $\mathcal{S}$ can simulate this phase following the protocol normally (since all relevant parameters have been explicitly chosen), while for $i \in [2]$ $\mathcal{S}$ does the following:
  1. $\mathcal{A}$ computes the initial step of the ballot $\bar{b}_{0,i}$ on behalf of $\mathcal{A}_0$ and proves its correctness with the appropriate DVNIZKPs. By rewinding $\mathcal{A}$ and exploiting the control of the random oracle, $\mathcal{S}$ is able to extract from the DVNIZKPs the values of k, λ, and $\bar{z}_{i,\ell}$ for all $\ell \in [M]$ (see [35]). Moreover, since $\mathcal{A}_0$ communicates the set of indexes of valid tokens V_i to the voter associated to v_i (that is controlled by the simulator), $\mathcal{S}$ can reconstruct the values of the $\sigma_{i,\ell}$ for all $\ell \in [M]$.
  2. $\mathcal{S}$ computes step 1 of the ballot $\bar{b}_{1,i} = (\bar{b}_{1,i,\ell})_{\ell \in [M]}$ as:

$$\bar{b}_{1,i,\ell} = A^{\bar{z}_{i,\ell}\tilde{z}_{i,\ell}(\sigma_{i,\ell}+x_i'+\tilde{x}_i)} \cdot \varXi^{\bar{z}_{i,\ell}\tilde{z}_{i,\ell}(-1)^i} \overset{*}{=} g^{z_{i,\ell}(\sigma_{i,\ell}+x_i)} \qquad \forall \ell \in [M] \tag{36}$$

where $\stackrel{*}{=}$ of Equation (36) holds iff $\Xi = g^{ab}$ in the DDH challenge. Since it controls the voter associated to v_i, $\mathcal{S}$ can forge the DVNIZKPs exploiting the value e_i. In order to hide from $\mathcal{A}$ which tokens are valid, these DVNIZKPs are forged using random values.

3. $\mathcal{S}$ can perform step 2 on behalf of $\mathcal{A}_2$ normally, then $\mathcal{A}$ computes step 3 on behalf of $\mathcal{A}_0$ and proves its correctness.
4. Finally $\mathcal{S}$ computes the final ballot $b_i = (b_{i,\ell})_{\ell\in[M]}$ as:

$$b_{i,\ell} = A^{\tilde{y}_{i,\ell} y''_{i,\ell}(\sigma_{i,\pi_i(\ell)}+x'_i+\tilde{x}_i)} \cdot \Xi^{\tilde{y}_{i,\ell} y''_{i,\ell}(-1)^i} \stackrel{*}{=} g^{y_{i,\ell}(\sigma_{i,\pi_i(\ell)}+x_i)} \tag{37}$$

where again $\stackrel{*}{=}$ of Equation (37) holds if and only if $\Xi = g^{ab}$ in the DDH challenge.

- **Voting**: Phases 1, 2, and the Challenge are performed as in Definition 10.
- **Tallying**. Without loss of generality, suppose that only the v_i with $i \in [T]$ have voted. For $\ell \in [M]$, $\mathcal{S}$ carries on with the simulation as follows:
  1. $\mathcal{S}$ computes the preliminary and final votes on behalf of $\mathcal{A}_1$ and $\mathcal{A}_2$ following the protocol without problems. In fact, for $i \in [2]$, we have that

$$\frac{\alpha'_\ell}{y'_{i,\tilde{\phi}_i(\ell)}} = \frac{a\tilde{\alpha}_\ell}{a\tilde{y}_{i,\tilde{\phi}_i(\ell)}} = \frac{\tilde{\alpha}_\ell}{\tilde{y}_{i,\tilde{\phi}_i(\ell)}} \qquad \forall \ell \in [M], \tag{38}$$

  and these values are known to $\mathcal{S}$.
  2. $\mathcal{S}$ computes and publishes the values $g^{\alpha_\ell} = A^{\tilde{\alpha}_\ell \alpha''_\ell}$ $\forall \ell \in [M]$, and simulates the proofs of correctness.
  3. Finally note that $\mathcal{S}$ can compute:

$$\sum_{i=1}^{T} x''_i = \tilde{x}_1 - b + \tilde{x}_2 + b + \sum_{i=3}^{T} x''_i = \tilde{x}_1 + \tilde{x}_2 + \sum_{i=3}^{T} x''_i, \tag{39}$$

  so for the rest of the tallying phase $\mathcal{S}$ can follow the protocol.
- **Guess** Eventually $\mathcal{A}$ will output a guess μ' of the coin flip performed by $\mathcal{S}$ during the Challenge. $\mathcal{S}$ then outputs 0 to guess that $\Xi = g^{ab}$ if $\mu' = \mu$, otherwise it outputs 1 to indicate that Ξ is a random group element $R \in \mathbb{G}$.

The case in which the adversary controls $\mathcal{A}_1$ and the case in which the adversary controls $\mathcal{A}_2$, proceed similarly. If $\mathcal{A}_1$ is corrupted, the main difference is that $\mathcal{S}$ implicitly sets:

$$\alpha''_\ell = a \cdot \tilde{\alpha}_\ell, \qquad y''_{i,\ell} = a \cdot \tilde{y}_{i,\ell}, \qquad \bar{z}_{i,\ell} = a \cdot \tilde{y}_{i,\ell}, \tag{40}$$

while α'_ℓ, $y'_{i,\ell}$, $z'_{i,\ell}$ are chosen normally.

If $\mathcal{A}_2$ is corrupted, the main difference is that $\mathcal{S}$ implicitly sets:

$$x'_i = \tilde{x}_i + (-1)^i b, \tag{41}$$

while x''_i is chosen normally.

Essentially, in all three cases when Ξ is not random the simulator $\mathcal{S}$ gives a perfect simulation. This means that the advantage is preserved, so it holds that:

$$\mathbb{P}[\mathcal{S}(g, A, B, \Xi = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{42}$$

On the contrary, when Ξ is a random element $R \in \mathbb{G}$, every token and vote belonging to the free voters becomes independent from the values that would have been computed by following the protocol (since they are simulated using the random value R), so $\mathcal{A}$ can gain no information about the votes from them, while the tally is always correct. Since the security game is structured in such a way that the tally and the tokens of the other voters (i.e. the values where Ξ is not used in the computation by $\mathcal{S}$) do not give any information about the coin flip μ, we have that:

$$\mathbb{P}[\mathcal{S}(g, A, B, \Xi = R) = 0] = \frac{1}{2}. \tag{43}$$

Therefore, $\mathcal{S}$ can play the DDH game with non-negligible advantage $\frac{\varepsilon}{2}$. □

5.2 General properties of the protocol

The general properties of a vote system introduced in Section 2.5, can all be proved for the protocol described in Section 3.1.

Proposition 1 (Correctness). *If the underlying BB is insert-only (as described in Section 2.4), then the protocol is correct, as per Definition 5.*

Proof. This property derives directly from the properties of the bulletin board: since it is insert-only cast votes cannot be altered or erased. As specified at the end of the voting phase (see Section 3.1), forged ballots are not accepted. This means that only the voter to whom the *v-tokens* have been issued is able to cast them since the adversary does not have the signing key s_i of honest voters. Finally, in case of multiple ballots cast by the same voter only the most recent one is considered, preventing double voting. □

Proposition 2 (Fairness). *In the random oracle model, if the DDH assumption holds, then the protocol is fair, as per Definition 6.*

Proof. Thanks to Theorem 1, in the random oracle model, if the DDH assumption holds then the protocol has vote-indistinguishability. Therefore, the votes cast do not reveal how many preferences each candidate has received until they are processed by the authorities, and this does not happen until the voting phase has ended. Even the processed votes do not reveal such information until the tallying values $g^{\alpha_\ell \sum x'_i}$ and $g^{\alpha_\ell \sum x''_i}$ are published by the authorities in the steps 3 and 4 of tallying, at which point the end results are computable by anyone and therefore public. □

Proposition 3 (Privacy). *In the random oracle model, if the DDH assumption holds, the bulletin board is publicly readable and insert-only (as described in Section 2.4), then the protocol is private, as per Definition 7.*

Proof. Thanks to Theorem 1, in the random oracle model, if the DDH assumption holds then the protocol has vote-indistinguishability, so a vote (even processed for the tallying) does not reveal the preference expressed by the voter. A vote only reveals that the corresponding ballot has actually been used (note that in normal elections it is often public the information whether a voter has voted). The protocol uses pseudonymous identifiers that can not be directly linked to the real identity of voters, so privacy is preserved. □

Proposition 4 (Verifiability). *In the random oracle model, if the DDH assumption holds, and the bulletin board is publicly readable, then the protocol satisfies both universal and individual (end-to-end) verifiability, as per Definition 8.*

Proof. Since the BB is publicly readable, anyone can check that every vote has been cast correctly. Moreover, anyone can see the processed votes received by the candidates and count the preferences using the values published by the authorities during the tallying. In the random oracle model, if the DDH assuption holds the NIZKPs are sound, so they allow everyone to do a consistency check of the computations performed at every step, and universal verifiability holds.

Similarly, the DVNIZKPs given during the Registrar phase allow the voters to verify which of their tokens are valid. Analyzing the BB and the tallying values, every voter can also check that their preferences have been correctly expressed and counted in the final result. □

Proposition 5 (Vote-Coercion Resistance). *In the random oracle model, if the DDH assumption holds, then the protocol is vote-coercion resistant, as per Definition 9.*

Proof. In order to comply with the coercer's request, a voter associated to $v_i \in V_c$ has to assign the valid tokens to $(C_{i,1}, \ldots, C_{i,P})$. Since the Registrar Phase is performed in a protected environment, only the voter associated to v_i knows which tokens are valid, and cannot give a meaningful proof of this fact to $\mathcal{A}$ as discussed at the end of the registrar phase (Section 3.1).

Thanks to Theorem 1, in the random oracle model, if the DDH assumption holds, then the protocol has vote-indistinguishability and the only way to determine if a vote expresses a specific choice is to distinguish valid and decoy tokens. Since $\mathcal{A}$ cannot do so, all the information that can be gained from the votes is given by the final tally. This means exactly that the probability of $\mathcal{A}$ detecting that a voter in V_c has not followed its instruction is the same in Ψ_1 and Ψ_2. □

6 Conclusions

In this paper we have generalized the *two-candidates-one-preference* e-voting protocol of [35] into an *M-candidates-P-preferences* protocol. We have tweaked the system of ZKPs that ensure transparency and full auditability of the process by using non-interactive proofs to enhance efficiency, exploiting designated-verifier proofs to preserve plausible deniability against coercers. Moreover, we have abandoned the blockchain infrastructure in favor of a more traditional bulletin board.

Compared with the two-candidates protocol, our generalization introduces an additional authority, that is required in order to properly mask the multiple valid and decoy tokens in each ballot, so that the system remains secure even if one authority is corrupt.

Note that the authorities can perform the setup phase asynchronously, and possible DOS attacks may be mitigated with a long-lasting Registrar phase. We can also adopt the strategy of dividing the authorities in independent triplets that manage restricted pools of voters (much like how large-scale elections are divided in voting districts). This approach limits the damage in case more than one authority is corrupted, speeds up the final step of tallying (whose computational cost is linear in the number of votes managed by a triplet of authorities), and enhances the overall efficiency by distributing the workload.

Efficiency and scalability. The computational and resource cost of our protocol scales linearly in $N \cdot M$, where M is the number of candidates and N is the number of voters managed by a triplet of authorities. In particular:

- In the setup phase the size of the published values is:

  $$2 \cdot (2MN + N + M + 1) \cdot (2|\mathbb{G}| + |\mathbb{Z}_p|) + N \cdot |v|,$$

  where $|v|$ is the size of a pseudonymous identifier. $\mathcal{A}_0$ stores $2 + NM$ secret elements of $\mathbb{Z}_p$, $\mathcal{A}_1$ stores $2NM + N + M$ secret elements of $\mathbb{Z}_p$, $\mathcal{A}_2$ stores $NM + N + M$ secret elements of $\mathbb{Z}_p$. Every authority also stores the N identifiers.
- In the registrar phase each of the N voters receive data of size:

  $$23M \cdot |\mathbb{G}| + 24M \cdot |\mathbb{Z}_p| + |v|,$$

  and have to store also the designated and signing key-pairs which have additional size $|\mathbb{G}| + |\mathbb{Z}_p| + |K| + |s|$ ($|K|$ and $|s|$ are respectively the size of the public and secret signing keys).
  The size of the data published on the BB in this phase is:

  $$N \cdot (M \cdot |\mathbb{G}| + |K| + |v|).$$

- In the voting phase the size of the data published on the BB is:

  $$(T + \texttt{revote}) \cdot (|v| + |\texttt{sig}| + M \cdot |C|),$$

  where T is the number of voters that cast a valid ballot, `revote` is the number of duplicate votes, $|\texttt{sig}|$ is the size of the signature, $|C|$ is the size of a candidate's identifier.
- In the tallying phase the size of the data published on the BB is:

  $$M \cdot \big[(6T + 15) \cdot |\mathbb{G}| + (2T + 5) \cdot |\mathbb{Z}_p|\big].$$

The effort required by an observer to compute the results of the election is:

$$M \cdot \left[(2T+5)\cdot \texttt{hash} + (8T+9)\cdot \texttt{mul} + (10T+21)\cdot \texttt{exp} + (5T+10)\cdot \texttt{check}\right],$$

where hash denotes the cost of computing the hash digest on 6 elements of $\mathbb{G}$, mul denotes the cost of the group operation (multiplication) in $\mathbb{G}$, exp denotes the cost of the scalar operation (exponentiation) in $\mathbb{G}$, check denotes the cost of comparing two elements of $\mathbb{G}$.
Once the results have been published, to check them we save a computational effort of $\left[M\cdot(T-1)\right]\cdot(\texttt{mul}+2\texttt{exp}+\texttt{check})$ since we do not have to re-compute R_ℓ and F_ℓ.

Security. The protocol fulfills all the security properties required for an e-voting protocol to be considered secure, proven in the random oracle model under the classical Decisional Diffie-Hellman Assumption.

Regarding coercion resistance, the differences between definitions are subtle. In its strongest form, coercion resistance includes protection against forced abstention attacks and randomized voting. Randomized vote attacks are less effective in swaying an election result with respect to other coercion attacks, forced abstention may be more effective, but it would require the coercer more effort, considering that more voters have to be controlled in order to achieve an impacting result. In fact, in our protocol the attacker should identify every coerced voter by requesting a signature, in order to link the voter's identity with a public key and its ballot, as published in the BB.

Although our definition of coercion resistance seems weaker, we remark that the most prominent e-voting protocols with stronger defence against coercion assume that there is a moment during the voting phase when the voter is not under control of the attacker. The Amun protocol, instead, protects the voter even if during the voting period there is constant surveillance from the coercer. Therefore, this may be preferable when the voting period is limited, since, in this scenario, it is more likely for the attacker to maintain continuous control.

To have any kind of anti-coercion resistance is essential that there is a moment where the voter receives some private information that can then be concealed from the coercer with plausible deniability. In the description of the protocol we have assumed that the communication between the voter and the authorities during the registrar phase happens in a safe and controlled environment, where the coercer has no power. This requirement is equivalent to exchanging information through untappable channels. This is a common assumption in coercion-resistant protocols [19,7].

In [26], the authors propose an alternative to untappable channels, introducing the weaker assumption that the adversary cannot maintain active surveillance over the voter. In other words, they assume that there will be wide surveillance gaps in which the voter is free to act, and exploit these gaps to enact anti-coercion strategies.

In particular when a voter registers, the voting credential is not issued right away, but instead delivered (at least partially) after a random delay. After a second, subsequent random time, the registration authorities send a Designated-Verifier Non-Interactive Zero-Knowledge Proof (DVNIZKP) to prove the correctness of the credential. During the waiting periods, a coerced voter can exploit a surveillance gap to request a *forged credential*, which can be used to evade coercion. If the voter received the real credential when not under active surveillance, they can feign to have never received it and pretend that the forged credential is the real one. The waiting period before receiving the DVNIZKP allows a coerced voter to exploit the designated verifier secret key to construct a DVNIZKP that validates the forged credential.

The same approach can also be employed with our protocol. In this case the credential is the set of indexes of the valid *v-tokens* inside the ballot.

Final remarks. Many election systems allow voters to cast a blank ballot or to leave some of the P possible preferences unexpressed. This feature can be easily added to the protocol presented here by simply adding P *dummy* candidates that represent blank choices.

References

1. Adida, B., Neff, C.A.: Ballot casting assurance. EVT **6**, 7 (2006)
2. Benaloh, J., Rivest, R., Ryan, P.Y., Stark, P., Teague, V., Vora, P.: End-to-end verifiability (04 2015)
3. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: Sok: A comprehensive analysis of game-based ballot privacy definitions. In: 2015 IEEE Symposium on Security and Privacy. pp. 499–516. IEEE (2015)
4. Bohli, J.M., Müller-Quade, J., Röhrich, S.: Bingo voting: Secure and coercion-free voting using a trusted random number generator. In: Proceedings of International Conference on E-Voting and Identity. pp. 111–124. Springer (2007)
5. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of computer and system sciences **37**(2), 156–189 (1988)
6. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: Beleniosrf: A non-interactive receipt-free electronic voting scheme. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1614–1625 (2016)
7. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy. pp. 354–368 (2008)
8. Cortier, V., Fuchsbauer, G., Galindo, D.: Beleniosrf: A strongly receipt-free electronic voting scheme. IACR Cryptol. ePrint Arch. p. 629 (2015)
9. Cortier, V., Gaudry, P., Glondu, S.: Belenios: a simple private and verifiable electronic voting system. In: Foundations of Security, Protocols, and Equational Reasoning, pp. 214–238. Springer (2019)
10. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Annual International Cryptology Conference. pp. 174–187. Springer (1994)
11. Escala, A., Guasch, S., Herranz, J., Morillo, P.: Universal cast-as-intended verifiability. In: International Conference on Financial Cryptography and Data Security. pp. 233–250. Springer (2016)

12. Fischer, J.B., Stern, J.: An efficient pseudo-random generator provably as secure as syndrome decoding. In: Advances in Cryptology—EUROCRYPT'96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings 15. pp. 245–255. Springer (1996)
13. Gardner, R.W., Garera, S., Rubin, A.D.: Coercion resistant end-to-end voting. In: International Conference on Financial Cryptography and Data Security. pp. 344–361. Springer (2009)
14. Grewal, G., Ryan, M., Bursuc, S., Ryan, P.: Caveat Coercitor: Coercion-Evidence in Electronic Voting. In: Proceedings - IEEE Symposium on Security and Privacy. pp. 367–381. IEEE (2013). https://doi.org/10.1109/SP.2013.32
15. Haines, T., Smyth, B.: Surveying definitions of coercion resistance. Cryptology ePrint Archive (2019)
16. Heather, J., Lundin, D.: The append-only web bulletin board. In: International Workshop on Formal Aspects in Security and Trust. pp. 242–256. Springer (2008)
17. Iovino, V., Visconti, I.: Non-interactive zero knowledge proofs in the random oracle model. In: Codes, Cryptology and Information Security: Third International Conference, C2SI 2019, Rabat, Morocco, April 22–24, 2019, Proceedings-In Honor of Said El Hajji. pp. 118–141. Springer (2019)
18. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Advances in Cryptology—EUROCRYPT'96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings. pp. 143–154. Springer (2001)
19. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Towards Trustworthy Elections, pp. 37–63. Springer (2010)
20. Kiayias, A., Kuldmaa, A., Lipmaa, H., Siim, J., Zacharias, T.: On the security properties of e-voting bulletin boards. In: Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings. pp. 505–523. Springer (2018)
21. Krips, K., Willemson, J.: On practical aspects of coercion-resistant remote voting systems. In: International Joint Conference on Electronic Voting. pp. 216–232. Springer (2019)
22. Kusters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: 2010 23rd IEEE Computer Security Foundations Symposium (2010)
23. Küsters, R., Truderung, T., Vogt, A.: Verifiability, privacy, and coercion-resistance: New insights from a case study. In: 2011 IEEE Symposium on Security and Privacy. pp. 538–553. IEEE (2011)
24. Lindell, Y.: Tutorials on the Foundations of Cryptography, chap. How to Simulate It – A Tutorial on the Simulation Proof Technique, pp. 277–346. Springer (2017). https://doi.org/10.1007/978-3-319-57048-8_6
25. Longo, R.: Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols. Ph.D. thesis, University Of Trento, Department of Mathematics (2018)
26. Longo, R., Morelli, U., Spadafora, C., Tomasi, A.: Adaptation of an i-voting scheme to italian elections for citizens abroad. In: E-Vote-ID 2022. University of Tartu Press (2022)
27. Ltd., S.P.: Swiss post official protocol specifications, https://gitlab.com/swisspost-evoting/documentation/-/blob/master/Protocol/Swiss_Post_Voting_Protocol_Computational_proof.pdf

28. of Ministers, C.: Recommendation cm/rec(2017)5 of the committee of ministers to member states on standards for e-voting, https://search.coe.int/cm/Pages/result_details.aspx?ObjectID=0900001680726f6f
29. NIST: Voluntary voting system guidelines overview - nist, https://www.nist.gov/system/files/documents/2017/05/24/vvsgvol12.pdf
30. Popoveniuc, S., Kelsey, J., Regenscheid, A., Vora, P.: Performance requirements for end-to-end verifiable elections. In: Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections. pp. 1–16 (2010)
31. Roth, R.M.: Introduction to coding theory. IET Communications **47**(18-19), 4 (2006)
32. Ryan, P.Y., Rønne, P.B., Iovino, V.: Selene: Voting with transparent verifiability and coercion-mitigation. In: International Conference on Financial Cryptography and Data Security. pp. 176–192. Springer (2016)
33. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4**, 161–174 (1991). https://doi.org/10.1007/BF00196725
34. Shoup, V., Alwen, J.: Σ-Protocols Continued and Introduction to Zero Knowledge (2007), https://cs.nyu.edu/courses/spring07/G22.3220-001/lec3.pdf
35. Spadafora, C., Longo, R., Sala, M.: A coercion-resistant blockchain-based E-voting protocol with receipts (2021). In: Advances in Mathematics of Communications. American Institute of Mathematical Sciences, doi:10.3934/amc.2021005 (2021)