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

Riccardo Longo[1][a] and Chiara Spadafora [2][b]
[1]*Fondazione Bruno Kessler, Center for Cybersecurity, Trento, Italy*
[2]*Department of Mathematics, University Of Trento, 38123 Povo, Trento, Italy*
*rlongo@fbk.eu,chiara.spadafora@unitn.it*

Keywords: E-Voting, Over-the-Shoulder Attack, Coercion-Resistance, End-to-End Verifiability, Formal Proof of Security.

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. We prove the security of the construction under the standard Decisional Diffie Hellman assumption in the random oracle model.

## 1 INTRODUCTION

Remote voting, enhanced by advanced cryptographic techniques, offers more security than traditional paper-based voting but raises privacy concerns when voting outside secure booths.

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 (JCJ10), which are indistinguishable from real ones but that do not produce valid votes. However, if the adversary keeps the voter under control until the end of the voting period, it becomes impossible to re-vote with the valid credential.

Here we present the Amun[1] protocol, which hides the real choice expressed by a voter even if an adversary is physically monitoring the elector during 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.

The authors in (SLS21) suggest a blockchain-based remote e-voting protocol for two candidates, where voters have two voting tokens (*v-tokens*): one is valid and the other is a decoy, but only the voter knows which is which. The Amun protocol extends this concept to support multiple candidates and choices.

In a setting where the voter can choose $P$ out of $M$ candidates, extra care has to be taken in the design of the *v-tokens*. In fact, a naïve straightforward adaptation of (SLS21) might reveal whether two *v-tokens* have the same validity, since there are multiple valid and decoy *v-tokens*. To deal with this we had to introduce extra masking steps and another authority to fully hide the *v-tokens* validity. So, 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 (SLS21), votes are cast by assigning to the candidates some "voting tokens" generated during registration. 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, 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 (GGR09; HS19; JCJ10; KW19; KTV10). *Civitas* (CCM08), which derives from *JCJ* (JCJ10), deals with coercion by allowing voters to vote multiple times via a mechanism of *real*

[a] https://orcid.org/0000-0002-8739-3091
[b] https://orcid.org/0000-0003-3352-9210
[1] Amun was a major ancient Egyptian deity. The name Amun 𓇋𓏠𓈖𓀭 meant something like "the hidden one" or "invisible".

and *fake* credentials. *Selene* (RRI16) 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. *Belenios* (CGG19) 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 *BeleniosRF* (CCFG16).

**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

Most of the algebraic preliminaries we need to build the protcol, such as the *Decisional Diffie-Hellmann Assumption* (DDH) and commitment schemes, can be found in (SLS21). Here we report only the additional tools required for the generalization.

For the sake of compactness we use the following notation for the indexes: $[n] = \{i \in \mathbb{N} : 1 \leq i \leq n\}$, $(t_j)_{j\in[m]} = (t_1, \dots, t_m)$.

### 2.1 Equality of discrete logarithms

Since it is used extensively in the protocol, we report here the Zero-Knowledge Proof (ZKP) for the equality of two discrete logarithms (SLS21), which is a variation of the Schnorr interactive protocol (Sch91; SA07).

**Protocol 1.** *Let $\mathbb{G}$ be a cyclic group of prime order $p$, let $u, \bar{u}$ be generators of $\mathbb{G}$, and let $z, \bar{z} \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. $\mathcal{P}$ knows $\omega$ and wants to convince $\mathcal{V}$ that $u^\omega = z$ and $\bar{u}^\omega = \bar{z}$, without disclosing $\omega$. The values of $u$, $z$, $\bar{u}$ and $\bar{z}$ are publicly known.*

1. *$\mathcal{P}$ generates a random $r$ and computes the commitments $t = u^r$ and $\bar{t} = \bar{u}^r$, then sends $(t, \bar{t})$ to $\mathcal{V}$.*
2. *$\mathcal{V}$ generates a challenge $c \in \mathbb{Z}_p$ and sends it to $\mathcal{P}$*[2].
3. *$\mathcal{P}$ computes $s = r + c \cdot \omega$ and sends $s$ to $\mathcal{V}$.*
4. *$\mathcal{V}$ 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 protocol aborts.*

[2]Depending on how this challenge is generated, different types of ZKP can be instantiated, see Section 2.2

### 2.2 Non-Interactive Zero-Knowledge Proofs.

Non-Interactive Zero-Knowledge Proofs (NIZKP) are a special type of ZKPs that allow the prover to publish a proof that can be independently verified by all the relevant parties later on. The Fiat-Shamir technique (IV19) can be used to transform an interactive sigma protocol into a NIZKP by exploiting a hash function modeled as a random oracle (RO). 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.

### 2.3 Designated-Verifier Zero-Knowledge Proofs.

Designated-Verifier Non-Interactive ZKP systems (DVNIZKPs (JSI01)) are protocols which retain most of the security properties of a NIZKP, but are not publicly verifiable: only the owner of some secret (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.

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 (CDS94) 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 $\mathcal{V}$ with $D = u^e \in \mathbb{G}$ the corresponding public key. As in Protocol 1, $\mathcal{P}$ knows $\omega$ and wants to convince $\mathcal{V}$ that $u^\omega = z$ and $\bar{u}^\omega = \bar{z}$, without disclosing $\omega$. 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 $\omega$ (such an $\omega$ 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 $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D)$, $c_1 = c - c_0$;*
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 $c = H(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D)$, $c_0 = c - c_1$;
4. $\mathcal{V}$ computes $s_0 = r_0 + c_0 \cdot e$ and obtains the forged DVNIZKP: $(u, \bar{u}, z, \bar{z}, t, \bar{t}, t_0, D, s_0, s_1, c_0, c_1)$.

## 2.4 Bulletin Board

The concept of *(Web) Bulletin Board* (BB, (KKL$^+$18)) is well established in literature, as its use in e-voting.

A BB is a log service (CCM08) 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 (HL08).

## 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 1** (Correctness). *Correctness (JCJ10) 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 2** (Fairness). *Fairness (oM17) requires that no information about how many votes each candidate has received can be learned until the voting results are published.*

**Definition 3** (Privacy). *Privacy ((NIS17; KTV11; BCG$^+$15)) is defined as the inability of the adversary to distinguish, given two candidates $C_1, C_2$, whether $V_i$ voted for $C_1$ or $C_2$.*

**Definition 4** (Verifiability). *Verifiability (AN06; JCJ10; CCFG16) 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 (AN06) 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 (EGHM16): every voter can check that their vote was correctly cast.*
  - *Recorded-as-cast verifiability (Ltd21): every voter can check that their vote was recorded as it was cast.*
  - *Tallied-as-recorded verifiability (PKRV10): anyone can check that cast votes were correctly tallied.*

*In (BRR$^+$15), the combination of cast-as-intended, recorded-as-cast, and tallied-as-recorded, is called* End-to-end.

Coercion resistance ((JCJ10; HS19)) 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.

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 use a slightly weaker adaptation of the definition of Coercion Resistance given in (JCJ10):

**Definition 5** (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 $\Psi_1$ be the scenario in which $\mathcal{A}$ has access only to the final tally. Let $\Psi_2$ 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 $\Psi_1$ and $\Psi_2$.*

## 3 MULTI-CANDIDATE E-VOTING

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. with the same *v-tokens* ignoring 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.

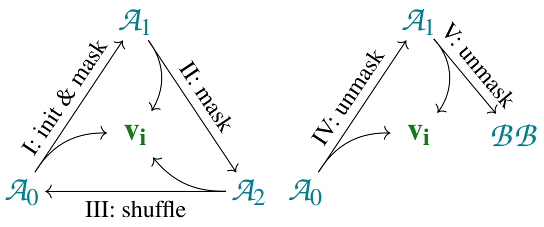

Figure 1: The main steps of the ballot generation procedure, which correspond to steps 3-7 of the Registrar Phase as described in Section 3.1.2.

- **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 computations are published, alongside a set of NIZKPs that allow anyone to check that the results are correct and there has not been any tampering. Every voter can also check, by examining the bulletin board, that their *v-tokens* have been cast and counted correctly.

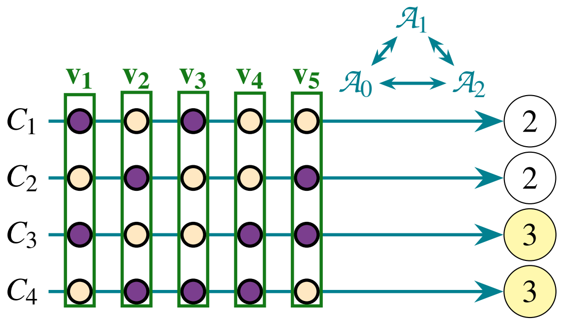

Figure 2: Example of voting and tallying. Each voter has two valid tokens ● and two decoy tokens ●. After the tallying it is revealed that candidates $C_3$ and $C_4$ are elected having received more preferences (3) with respect to the other two candidates (who received only 2).

### 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[3] $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$.
4. one ballot $b_i$ (comprising $M$ *v-tokens*) for every $i \in [N]$, i.e. one for each eligible voter.

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

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.

### 3.1.1 Setup

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

1. a secure group $\mathbb{G}$ of prime order $p$ in which the DDH assumption holds, with a generator $g \in \mathbb{G}$;
2. a commitment scheme `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 $\lambda$ in $\mathbb{Z}_p^*$. $\mathcal{A}_0$ knows that the *v-tokens* computed using $k$ are valid, while the ones computed using $\lambda$ 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 $g^k$, $g^\lambda$, and, for every $i \in [N]$, it commits to $\big(v_i, (g^{\bar{z}_{i,\ell}})_{\ell \in [M]}\big)$.

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

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 $\Big(v_i, g^{x'_i}, (g^{z'_{i,\ell}})_{\ell \in [M]}, (g^{y'_{i,\ell}})_{\ell \in [M]}\Big)$.

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

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 $\Big(v_i, g^{x''_i}, (g^{y''_{i,\ell}})_{\ell \in [M]}\Big)$.

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

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

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

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 and the Fiat-Shamir transformation just like in Section 2.2.

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

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

### 3.1.2 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 of eavesdropping or interference.
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 $V_i \subset [M]$ with cardinality is exactly $P$, then sets:

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

i.e. the random choice of the $V_i$ determines which tokens will be valid and which 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, $\forall \ell \in [M]$:

$$\bar{b}_{0,i,\ell} = \left(g^{\sigma_{i,\ell}} \cdot g^{x'_i} \cdot g^{x''_i}\right)^{\bar{z}_{i,\ell}} = g^{\bar{z}_{i,\ell}(\sigma_{i,\ell}+x_i)};$$

(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 its computations correct 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:

$$\begin{aligned} &\omega = k, && u = g, && z = g^k, \\ &\bar{u} = g^{\bar{z}_{i,\ell}}, && \bar{z} = g^{\bar{z}_{i,\ell}k}, && \forall \ell \in V_i, \\ &\omega = \lambda, && u = g, && z = g^{\lambda}, \\ &\bar{u} = g^{\bar{z}_{i,\ell}}, && \bar{z} = g^{\bar{z}_{i,\ell}\lambda} && \forall \ell \in [M] \setminus V_i, \end{aligned}$$

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

$$\begin{aligned} &\omega = \bar{z}_{i,\ell}, && u = g, && z = g^{\bar{z}_{i,\ell}}, \\ &\bar{u} = g^{\sigma_{i,\ell}} \cdot g^{x'_i} \cdot g^{x''_i}, && \bar{z} = \bar{b}_{0,i,\ell}. \end{aligned}$$

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

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

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:

$$\begin{aligned} &\omega = z'_{i,\ell}, && u = g, && z = g^{z'_{i,\ell}}, \\ &\bar{u} = \bar{b}_{0,i,\ell}, && \bar{z} = \bar{b}_{1,i,\ell} && \forall \ell \in [M]. \end{aligned}$$

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, $\forall \ell \in [M]$:

$$\bar{b}_{2,i,\ell} = \left(\bar{b}_{1,i,\ell}\right)^{y''_{i,\pi_i^{-1}(\ell)}} = g^{z_{i,\ell}y''_{i,\pi_i^{-1}(\ell)}(\sigma_{i,\ell}+x_i)}$$

and sends it to Alice and to $\mathcal{A}_0$, $\pi_i$ 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:

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

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

$$\bar{b}_{3,i,\ell} = \left(\bar{b}_{2,i,\ell}\right)^{\frac{1}{\bar{z}_{i,\ell}}} = g^{z'_{i,\ell}y''_{i,\pi_i^{-1}(\ell)}(\sigma_{i,\ell}+x_i)}$$

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:

$$\begin{aligned} &\omega = \frac{1}{\bar{z}_{i,\ell}}, && u = g^{\bar{z}_{i,\ell}}, && z = g, \\ &\bar{u} = \bar{b}_{2,i,\ell}, && \bar{z} = \bar{b}_{3,i,\ell} && \forall \ell \in [M]. \end{aligned}$$

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

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

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:

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

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 $\pi_i$ the authorities cannot distinguish the tokens unless they collude. Effectively, the DVNIZKPs prove to Alice that the ballot has been created by $\mathcal{A}_0$ with the correct number of valid and decoy tokens, and that it has been correctly shuffled by $\mathcal{A}_2$. 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. In fact, since Alice is in a protected environment, she can manipulate the received data without being able to prove or disprove any manipulation.So, given that she knows $e_i$, she can forge a proof that states the presumed validity of any $P$ of the $M$ tokens, making any proof worthless to a coercer.

### 3.1.3 Voting Phase

Voters assign their valid tokens to preferred candidates and decoy tokens to the others. Each voter signs this assignment with their private key and publishes it on the BB for verification. After voting, duplicate, incomplete, and forged ballots are filtered out (relying on public data only).

### 3.1.4 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* $\alpha_\ell$. 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 $\ell$ the index of the token $b_{i,\phi_i(\ell)}$ that the voter associated to $v_i$ sent to the candidate $C_\ell$. 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:

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

   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:

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

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

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

   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:

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

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

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

1. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ can compute $g^{\alpha_\ell}$ (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:

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

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

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

   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:

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

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

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

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) using:

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

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

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:

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

   again, anyone 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:

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

   and that:

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

   then anyone can compute:

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

6. $R_\ell$ and $F_\ell$ can now be computed by brute force, giving the number of preferences received by the $\ell$-th candidate.

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 $R_\ell$ and $F_\ell$ 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 (FS96) 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 (Rot06) 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 1.** *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 $M \leq 13$ (since in these cases $\binom{M}{P} < 9^2$).*

**Example 2.** *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:*

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

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 7. We will assume 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 6** (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 (i.e. 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 $C_\ell$, $\forall \ell \in [M]$, the $\tilde{\phi}_1(\ell)$-th and $\tilde{\phi}_2(\ell)$-th tokens of the first and second free voter respectively.*

.

- ***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 $\mu'$ of the coin flip that randomly assigned the voting preferences of the two free voters.*

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

**Definition 7** (Vote Indistinguishability). *An E-Voting Protocol with security parameter $\theta$ is VI-secure if, for every probabilistic polynomial-time adversary $\mathcal{A}$ that outputs a guess $\mu'$ of the coin flip $\mu$ (as described in the security game of Definition 6), there exists a negligible function $\eta$ such that $\mathbb{P}[\mu' = \mu] \leq \frac{1}{2} + \eta(\theta)$.*

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 (ROM), if the DDH assumption holds, then the protocol of Section 3.1 is VI-secure, as per Definition 7.*

*Proof.* Suppose there exists a polynomial time adversary $\mathcal{A}$, that can attack the scheme with advantage $\varepsilon$. 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:

$$(g, A = g^a, B = g^b, \Xi),$$

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

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

- **Init**. The adversary chooses the $N-2$ voters to control. Without loss of generality we 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 $i \in [2]$, $\ell \in [M]$:

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

$\mathcal{S}$ chooses the other values for authorities $\mathscr{A}_1$ and $\mathscr{A}_2$ 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:

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

for all $i \in [2]$, $\ell \in [M]$. Therefore, $\mathcal{S}$ can simulate the setup phase, exploiting the RO to simulate the NIZKPs for $x_i''$, $\alpha_\ell'$, $y_{i,\ell}'$ $z_{i,\ell}'$ 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 $\mathscr{A}_0$ 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 $k, \lambda$, and $\bar{z}_{i,\ell}$ for all $\ell \in [M]$ (see (SLS21)). Moreover, since $\mathscr{A}_0$ 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:

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

where $\overset{*}{=}$ of Equation (1) 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 $\mathscr{A}_2$ normally, then $\mathcal{A}$ computes step 3 on behalf of $\mathscr{A}_0$ and proves its correctness.
4. Finally $\mathcal{S}$ computes the final ballot $b_i = (b_{i,\ell})_{\ell \in [M]}$ as:

$$\begin{aligned} b_{i,\ell} &= A^{\tilde{y}_{i,\ell}y_{i,\ell}''(\sigma_{i,\pi_i(\ell)}+x_i'+\tilde{x}_i)} \cdot \Xi^{\tilde{y}_{i,\ell}y_{i,\ell}''(-1)^i} \\ &\overset{*}{=} g^{y_{i,\ell}(\sigma_{i,\pi_i(\ell)}+x_i)} \end{aligned} \tag{2}$$

where again $\stackrel{*}{=}$ of Equation (2) holds if and only if $\Xi = g^{ab}$ in the DDH challenge.

- **Voting**: Phases 1, 2, and the Challenge are performed as in Definition 6.
- **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:
  $$\frac{\alpha'_\ell}{y'_{i,\tilde{\phi}_i(\ell)}} = \frac{a\tilde{\alpha}_\ell}{a\tilde{y}_{i,\tilde{\phi}_i(\ell)}} = \frac{\tilde{\alpha}_\ell}{\tilde{y}_{i,\tilde{\phi}_i(\ell)}} \qquad \forall \ell \in [M],$$
  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:
  $$\sum_{i=1}^{T} x''_i = \tilde{x}_1 - b + \tilde{x}_2 + b + \sum_{i=3}^{T} x''_i = \tilde{x}_1 + \tilde{x}_2 + \sum_{i=3}^{T} x''_i,$$
  so for the rest of the tallying phase $\mathcal{S}$ can follow the protocol.
- **Guess** Eventually $\mathcal{A}$ will output a guess $\mu'$ 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 $\Xi$ 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:

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

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

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

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

while $x''_i$ is chosen normally.

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

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

On the contrary, when $\Xi$ 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 $\Xi$ is not used in the computation by $\mathcal{S}$) do not give any information about the coin flip $\mu$, we have that:

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

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 1.*

*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 3.1.3), 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** (Vote-Coercion Resistance)**.** *In the ROM, if the DDH assumption holds, then the protocol is vote-coercion resistant, as per Definition 5.*

*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 $(C_{i,1}, \dots, C_{i,P})$. 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 3.1.2).

Thanks to Theorem 1, in the ROM, 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 $\Psi_1$ and $\Psi_2$. □

The proof of the following propositions are straightforward adaptations of the proofs of Proposition 5, 7, 8 of (SLS21).

**Proposition 3** (Fairness)**.** *In the ROM, if the DDH assumption holds, then the protocol is fair, as per Definition 2.*

**Proposition 4** (Privacy)**.** *In the ROM, if the DDH assumption holds, the bulletin board is publicly readable and insert-only (see Section 2.4), then the protocol is private, as per Definition 3.*

**Proposition 5** (Verifiability)**.** *In the ROM, 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 4.*

## 6 CONCLUSIONS

In this paper we have generalized the *two-candidates-one-preference* e-voting protocol of (SLS21) 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 that 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:

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

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

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

  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 $R_\ell$ and $F_\ell$.

**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, while forced abstention may be more effective, but it would require more effort, since 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 (JCJ10; CCM08).

In the scenario where the voting period is limited, some voters may struggle to access a secure physical voting booth, potentially leading to disenfranchisement. However, the longer duration of the registration phase provides more opportunities for voters to reach a secure registration booth.

The authors of (LMST22; BLM$^{+}$23) propose a method to protect voters from coercion by exploiting surveillance gaps. They assume that an adversary cannot maintain constant surveillance over a voter, allowing the voter to act freely during these gaps. In particular, when a voter registers, the voting credential is not issued immediately, but after a random delay. A DVNIZKP is sent after another random wait to prove the credential's correctness. These random waiting periods allow a coerced voter to exploit surveillance gaps to forge a credential and its DVNIZKP. This approach can also be applied to our protocol, where the credential can be seen as the set of indexes of the valid *v-tokens*.

**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.

**ACKNOWLEDGMENTS** This work has been partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. This work has been partially supported by a joint laboratory between Fondazione Bruno Kessler and the Italian Government Printing Office and Mint, Italy. The authors are members of the INdAM Research Group GNSAGA.

## References

Ben Adida and C Andrew Neff. Ballot casting assurance. *EVT*, 6, 2006.

David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *Symposium on Security and Privacy*. IEEE, 2015.

Matteo Bitussi, Riccardo Longo, Francesco Antonio Marino, Umberto Morelli, Amir Sharif, Chiara Spadafora, and Alessandro Tomasi. Coercion-resistant i-voting with short pin and oauth 2.0. In *E-Vote-ID 2023*, LNI. Herausgeber et al. Bonn, 2023. to appear.

Josh Benaloh, Ronald Rivest, Peter Y.A. Ryan, Philip Stark, Vanessa Teague, and Poorvi Vora. End-to-end verifiability, 04 2015.

Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, 2008.

Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*. Springer, 1994.

Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: a simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning*. Springer, 2019.

Alex Escala, Sandra Guasch, Javier Herranz, and Paz Morillo. Universal cast-as-intended verifiability. In *International Conference on Financial Cryptography and Data Security*. Springer, 2016.

Jean-Bernard Fischer and Jacques Stern. 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*. Springer, 1996.

Ryan W Gardner, Sujata Garera, and Aviel D Rubin. Coercion resistant end-to-end voting. In *International Conference on Financial Cryptography and Data Security*. Springer, 2009.

James Heather and David Lundin. The append-only web bulletin board. In *International Workshop on Formal Aspects in Security and Trust*. Springer, 2008.

Thomas Haines and Ben Smyth. Surveying definitions of coercion resistance. *Cryptology ePrint Archive*, 2019.

Vincenzo Iovino and Ivan Visconti. 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*. Springer, 2019.

Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections*. Springer, 2010.

Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. 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*. Springer, 2001.

Aggelos Kiayias, Annabell Kuldmaa, Helger Lipmaa, Janno Siim, and Thomas Zacharias. 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*. Springer, 2018.

Ralf Kusters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *2010 23rd IEEE Computer Security Foundations Symposium*, 2010.

Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *Symposium on Security and Privacy*. IEEE, 2011.

Kristjan Krips and Jan Willemson. On practical aspects of coercion-resistant remote voting systems. In *International Joint Conference on Electronic Voting*. Springer, 2019.

Riccardo Longo, Umberto Morelli, Chiara Spadafora, and Alessandro Tomasi. Adaptation of an i-voting scheme to italian elections for citizens abroad. *University of Tartu Press*, 2022.

Swiss Post Ltd. Swiss post official protocol specifications, 2021.

NIST. Voluntary voting system guidelines overview - nist, 2017.

Committee of Ministers. Recommendation cm/rec(2017)5 of the committee of ministers to member states on standards for e-voting, 2017.

Stefan Popoveniuc, John Kelsey, Andrew Regenscheid, and Poorvi Vora. Performance requirements for end-to-end verifiable elections. In *Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections*, 2010.

Ron M Roth. Introduction to coding theory. *IET Communications*, 47(18-19), 2006.

Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *International Conference on Financial Cryptography and Data Security*. Springer, 2016.

Victor Shoup and Joel Alwen. Σ-Protocols Continued and Introduction to Zero Knowledge, 2007.

Claus Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

Chiara Spadafora, Riccardo Longo, and Massimiliano Sala. A coercion-resistant blockchain-based E-voting protocol with receipts (2021). In *Advances in Mathematics of Communications*. AIMS, doi:10.3934/amc.2021005, 2021.