Multiple Candidates Coercion-Resistant Blockchain-Based E-Voting Protocol With Receipts

Riccardo Longo[1] and Chiara Spadafora[2]

[1] riccardolongomath@gmail.com
[2] c.spadaf@libero.it
Department of Mathematics, University Of Trento, 38123 Povo, Trento, Italy

Abstract. This paper extends the two-candidate's protocol of [11] to the multi-candidate case, making it applicable to elections where each voter expresses P preferences among M possible choices. The generalized protocol still achieves coercion and vote-selling resistance while being transparent, fully verifiable and receipt-based. The protocol relies on a generic blockchain with standard properties, and we prove the security of the construction under the standard Decisional Diffie Hellman assumption.

1 Introduction

Electronic voting can help speed up the whole election process and make it safer, however this technology can also create complex challenges to election operations. The advent of blockchain, with its nice properties such as transparency and non-repudiation [14], and its application to e-voting, promises to enhance the transparency, verifiability and security of the whole election process, but it also opens the door to a brand new complex set of problems. The blockchain layer may be a custom implementation (e.g. through the framework Hyperledger Fabric [2]) or exploit an existing and widely used blockchain (e.g. Bitcoin [8]) and use specific protocols to maintain a consistent subchain on top of it (e.g. [7]).

Since an election is a sensitive matter, a secure voting scheme should be robust against to both coercion and vote-selling. A protocol is *coercion-resistant* if voters can cast their ballots as they want, even if someone tries to actively force them to vote for a specific candidate. A protocol is *vote-selling resistant* if it does not give a proof of vote that can be understood by everyone, so any transaction that may involve a specific vote as a counterpart does not have a proof, thus the voter actually maintains free choice.

The protocol presented in [11] satisfied these properties while being receipt-based[3] (so that voters can check the correct tallying of their votes), by exploiting an underlying blockchain. However the protocol considers only elections with

[3] We exploit cryptography to give receipts that allow voters to verify that their votes were counted as cast, without revealing which candidates were voted for.

just two candidates, i.e. two possible choices. In this paper we propose a natural generalization that allows for multiple candidates and more than one choice, which can also be exploited to express blank or partial ballots.

Organization We present some preliminaries in Section 2, in particular in Section 2.2 we state the protocol that we use to demonstrate the correctness of the system, and in Section 2.4 we describe what we mean by the term *blockchain* and which are the consistency rules that miners are needed to enforce. We describe our protocol in Section 3 and we provide a proof of security in Section 4. Finally, in Section 5 we draw some conclusions.

Related Work As of June 2021, according to the database of the International Institute for Democracy and Electoral Assistance[4], twelve countries allow for internet voting. Estonia remains the pioneer state in the sector with the use of *Helios* [1]. In Helios, voters do not need to be authenticated until they cast votes. Under such a situation, anyone can participate and test the system.

On the other hand, many countries allow remote voting only under special circumstances. France, New Zealand, Pakistan and Panama allow remote voting for citizens living abroad, Armenia reserves this privilege only to military and diplomats.

Different is the use of physical e-voting, which uses DRE voting machines, optical scanners or punched cards. These systems are far more widespread, for example they are widely used in the US, India, Mexico, Venezuela, and other countries.

Far more interesting are blockchain-based e-voting protocols, and some states are already trying to use this technology. For example, this is the case of West Virginia in 2018 and Utah County in 2020: both used the voting application Voatz [13]. *Voatz* [12] is based on an application which is able to perform biometric identifcation of the voter. The system runs on the Voatz blockchain which is built using the Hyperledger framework. In 2020, the Japanese startup LayerX announced[5] its collaboration with the Tsukuba city Council to build a blockchain-based e-voting application. This applications has not yet been used in a real election but has been used to vote for social development proposals. In September 2019, the city of Moscow allowed its voters to vote for the Parliamentary election through the use of a blockhain-based platform. Before its deployment a French researcher found two vulnerabilities in the system that made it completely insecure [4].

In [9] the authors intend to provide an overview of a current state of the art and current trends in the field of blockchain-based electronic voting (as of 2021).

2 Preliminaries

In this section we recall some basic definitions that we will use later on.

[4] The database is available at https://www.idea.int/data-tools/data/icts-elections.

[5] https://layerx.co.jp/news/pr201105/

Definition 1 (Index Notation). *To abbreviate notation for the various indexes, we will write $[n]$ to indicate the set of integers between 1 and n:*

$$[n] = \{i \in \mathbb{N} : 1 \leq i \leq n\}, \tag{1}$$

consequently we define the following notations for a set and a tuple of indexed elements respectively:

$$\{e_i\}_{i\in[n]} = \{e_i : i \in [n]\} \tag{2}$$

$$(t_j)_{j\in[m]} = (t_1, \dots, t_m). \tag{3}$$

Definition 2 (Negligible Function). *$\eta : \mathbb{N} \to \mathbb{R}$ is a negligible function in $k \in \mathbb{N}$ if, for every $c \in \mathbb{N}$ and for every $\gamma \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that*

$$|\eta(k)| < \left|\frac{1}{ck^\gamma}\right|, \quad \forall k > k_0.$$

2.1 Decisional Diffie-Hellman Assumption

We adopt the definition of the decisional Diffie–Hellman (DDH) problem and the relative hardness assumption given in [6].

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

$$\mathbb{B}\left(g, A = g^a, B = g^b, T\right) \to \{0, 1\} \tag{4}$$

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:

$$Adv_{\mathbb{B}} = \left|\mathbb{P}\left[\mathbb{B}\left(g, A, B, g^{ab}\right) = 1\right] - \mathbb{P}\left[\mathbb{B}\left(g, A, B, g^{\xi}\right) = 1\right]\right|, \tag{5}$$

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

Definition 3 (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 protocol which allows one party (the prover) to convince another party (the verifier) 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 4 (Completeness). *If $x \in \mathcal{L}$ (i.e. the prover is honest) then the verifier should accept the proof with probability 1.*

Definition 5 (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 6 (Zero-Knowledge). *For every verifier V there exists an efficient simulator that can generate transcripts that are indistinguishable from real interaction between a real prover and V.*

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

Equality of discrete logarithms We recall the variation of the Schnorr interactive protocol [10] presented in [11], which will be used in the proof of security described in Section 4.

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:*

$$u^\omega = z \quad \text{and} \quad \bar{u}^\omega = \bar{z}, \tag{6}$$

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 $t = u^r$ and $\bar{t} = \bar{u}^r$, then sends $(t, \bar{t})$ to the verifier.*
2. *The verifier computes a random $c \in \{0, 1\}$ and sends it to the prover.*
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.*
5. *The previous steps are repeated $\tau = \text{poly}(\log_2(p))$ times, i.e. the number of repetitions is polynomial in the length of p (the security parameter).*

Proposition 1 (Properties of Protocol 1). *Under the DDH assumption, the Protocol 1 satisfies the completeness, soundness and zero-knowledge properties, as per Definitions 4 to 6.*

Proof. A complete proof can be found in [11]. □

We recall also the following corollary and lemma, that will be useful in the security analysis. Again, the proof may be found in [11], while further discussion on Zero-Knowledge proofs and simulations can be found in [5].

Corollary 1 (Simulation of a ZKP). *Let $\mathcal{S}$ be a simulator that has to prove the validity of an input $(u, z, \bar{u}, \bar{z})$ to a verifier V that can be rewound. If the DDH assumption holds, $\mathcal{S}$ can simulate the proof without knowing ω, and this simulation is indistinguishable from a real Zero-Knowledge proof.*

Lemma 1 (Extracting the Secret). *If $\mathcal{A}$ has to prove to us the equality of discrete logarithms with the protocol above, and we have the ability to rewind its execution, then we can extract from $\mathcal{A}$ the secret exponent ω.*

2.3 Commitment Scheme

A commitment scheme [3] is composed by two algorithms:

- $\texttt{Commit}(m, r)$**:** takes the message m to commit with some random value r as input and outputs the commitment c and a decommitment value d.
- $\texttt{Verify}(c, m, d)$**:** 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 $\texttt{Verify}(c, m, d) = \texttt{Verify}(c, m', d') = \texttt{true}$.
- **Hiding:** Let $[c_1, d_1] = \texttt{Commit}(m_1, r_1)$ and $[c_2, d_2] = \texttt{Commit}(m_2, r_2)$ 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.

As in [11], our construction uses commitments to enhance the security, but in our analysis are only marginally involved in the proof.

2.4 Blockchain

Also this "primitive" is used exactly as in the two-candidates protocol, so we refer once again to [11] for a more comprehensive review and here we only recall the main properties that we require:

- **public**: the contents of are publicly readable, and no attacker is not able to indefinitely negate access or pass off a counterfeit copy as the original blockchain;
- **append-only**: the contents are immutable once published, but new data can be added, no attacker is able to reorder, delete or modify past transactions
- **transaction authorization**: a user's tokens cannot be spent by anybody else;
- **vote validation**: only valid votes are accepted and registered on the blockchain, i.e. users must spend all their tokens together, and send them to different candidates (see Section 3);
- **no double spending**: each token can be used in only one valid transaction.

2.5 General Requirements for Remote Voting Systems

Since an election is a sensitive matter, remote voting systems should satisfy certain requirements before being deployed. In [11] these properties are formally defined, and then it is proven that the two-candidates protocol satisfies them. The generalized protocol presented here satisfies the same properties, and in particular the properties of *Correctness, Fairness, Transparency, Privacy*, and *Verifiability* may be proven in the exact same way as in the original protocol, so we do not include definitions and proofs here.

To take into account the changes due to the presence of multiple candidates and choices, however, we have to adapt the proof of *Vote-Selling and Coercion Resistance*. So we recall the definition here, and we will prove that our proposed protocol satisfies them in Section 4.2.

Definition 7 (Vote-coercion resistance). *Voters should be able to cast their ballots as they want, even if someone tries to coerce them.*

Voters may sell their voting credentials (in the case of remote electronic voting) and there is no mathematical countermeasure to prevent it. However, a mitigation is to require additional information to properly operate with said credentials, and prevent the certification of correctness of this additional info.

Definition 8 (Vote-Selling Resistance). *A voting system is* vote-selling resistant *if any vote receipt that it provides and the credentials to cast a vote can be respectively properly interpreted and properly used only knowing some additional information ζ. The value of ζ is randomly generated and specific to the voter, who is the only one who knows it, and can easily construct a fake value ζ' indistinguishable from ζ (in particular there is no certificate for the real value of ζ).*

Note that this definition covers also Definition 7, if, given two candidate's choices C and C', for every coercer that wants to force the choice C', a voter (that wishes to express the choice C) can fabricate a value ζ' (as before) such that the vote it casts expresses the choice C' if the additional information is ζ', while it expresses C if the additional information is ζ.

3 Multi-Candidate Voting System

This section presents our proposal for a remote e-voting protocol based on blockchain technology 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 fake, 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 fake ones to the others. The voter gets a vote receipt on which the transaction of all the tokens will be displayed. In the final tally the fake *v-tokens* are discarded[6] and the whole process is publicly auditable. The aim of this voting system is to be fully verifiable, and to prevent coercion and vote selling, while being almost completely transparent and giving to the voter ballot casting assurance.

The protocol is divided into five phases:

[6] With *discard* we do not mean that the tokens are removed from the blockchain, which is infeasible due to our assumptions, but that everyone can count the valid tokens, among all the ones received by the candidate.

- **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.
- **Registrar.** Each voter creates a wallet (over which no one else has control) and registers it with the three authorities, which then proceed to create M indistinguishable *v-tokens*, P valid and $M - P$ fake, that will be controlled by this wallet. During the creation the voter receives also the information on which token is valid and which is fake, and the authorities prove with ZKPs that the tokens are correct. We assume that the interaction between the voter and any authority is private and untappable (even an authority does not see what the others are telling the voter). The information on which *v-token* is valid and which is fake is given without a receipt so the voter cannot officially prove the validity of a *v-token*, in particular this means that the ZKP must be interactive, so that any transcript of the proofs is worthless for an outsider.
- **Voting Phase.** All M *v-tokens* of a voter must be spent together to have a valid transaction, and they have to go to distinct candidates. After the *v-tokens* have been spent, a receipt is given to the voter. 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 Section 3.1) and the number of valid and fake votes received by each candidate is published. In the same time, the authorities publish a set of values that allows to check that there have been no manipulations of the ballots. Every voter can check, by examining the history of transactions received by the candidate's node, that their *v-tokens* have been cast correctly. Finally anyone can request a series of ZKPs to assure that the *v-tokens* have been correctly processed during the tallying phase.

3.1 Protocol Description

The key components involved in the protocol are:

1. A finite set of voters $V = \{v_i\}_{i \in [N]}$, with $N \in \mathbb{N}$ the number of eligible voters;
2. A finite set of candidates $C = \{c_l\}_{l \in [M]}$ with $M \in \mathbb{N}$ the number of candidates;
3. Three trusted authorities[7] $\mathcal{A}_0$, $\mathcal{A}_1$, 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.

Let us now present the details of the protocol phase by phase.

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

Setup The authority $\mathcal{A}_0$ selects a secure group $\mathbb{G}$ of prime order p in which the DDH assumption holds (see Definition 3), along with a generator $g \in \mathbb{G}$. Then it publishes $\mathbb{G}$, g, p.

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 fake, but this information is kept secret;
2. chooses uniformly at random $N \cdot M$ distinct values $\bar{z}_{i,l} \in \mathbb{Z}_p^*$, with $i \in [N]$, $l \in [M]$;
3. finally, $\mathcal{A}_0$ commits (see Section 2.3) to the values g^k, g^λ, and for every $i \in [N]$ it commits to $\big(v_i, (g^{\bar{z}_{i,l}})_{l\in[M]}\big)$.

It is important that all the values $\bar{z}_{i,l}$, k, λ remain private.

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

1. chooses uniformly at random M distinct values $\alpha'_l \in \mathbb{Z}_p^*$, with $l \in [M]$, 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 $z'_{i,l}, y'_{i,l} \in \mathbb{Z}_p^*$, with $i \in [N], l \in [M]$;
4. finally, $\mathcal{A}_1$ commits (see Section 2.3) to the values $g^{\alpha'_l}$, $\forall l \in [M]$, and for every $i \in [N]$ it commits to the tuple $\Big(v_i, g^{x'_i}, (g^{z'_{i,l}})_{l\in[M]}, (g^{y'_{i,l}})_{l\in[M]}\Big)$.

It is important that all the values α'_l, x'_i, $z'_{i,l}$, $y'_{i,l}$ remain private.

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

1. chooses uniformly at random M distinct values $\alpha''_l \in \mathbb{Z}_p^*$, with $l \in [M]$, 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 $y''_{i,l} \in \mathbb{Z}_p^*$, with $i \in [N]$, $l \in [M]$;
4. Finally $\mathcal{A}_2$ commits (see Section 2.3) to the values $g^{\alpha''_l}$, $\forall l \in [M]$, and for every $i \in [N]$ it commits to the tuple $\Big(v_i, g^{x''_i}, (g^{y''_{i,l}})_{l\in[M]}\Big)$.

It is important that all the values α''_l, x''_i, $y''_{i,l}$ remain private.

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, and all the tuples $\big(v_i, (g^{\bar{z}_{i,l}})_{l\in[M]}\big)\, \forall i \in [N]$;
– $\mathcal{A}_1$ publishes the decommitments for the values $g^{\alpha'_l} \forall l \in [M]$, and the tuples $\Big(v_i, g^{x'_i}, (g^{z'_{i,l}})_{l\in[M]}, (g^{y'_{i,l}})_{l\in[M]}\Big)\, \forall i \in [N]$;

– $\mathcal{A}_2$ publishes the decommitments for the values $g^{\alpha_l''} \forall l \in [M]$, and the tuples $\left(v_i, g^{x_i''}, (g^{y_{i,l}''})_{l\in[M]}\right) \forall i \in [N]$.

To simplify notation we introduce the following definitions for aggregate values:

$$x_i = x_i' + x_i'' \qquad \forall i \in [N] \tag{7}$$
$$\alpha_l = \alpha_l' \cdot \alpha_l'' \qquad \forall l \in [M] \tag{8}$$
$$z_{i,l} = \bar{z}_{i,l} \cdot z_{i,l}' \qquad \forall i \in [N], \forall l \in [M] \tag{9}$$
$$y_{i,l} = y_{i,l}' \cdot y_{i,l}'' \qquad \forall i \in [N], \forall l \in [M] \tag{10}$$

Registrar Phase In the description of this phase we omit the details of the operations that involve the blockchain, focusing on the interactions between the registering voter and the authorities.

For every voter $v_i \in V$ the following steps are performed:

1. Let Alice be the person associated to the voter v_i, note that the authorities do not need to know this association, and v_i can be a pseudonymous id. She creates her own new wallet, and goes in a safe and controlled environment where she is identified and authenticated as the eligible voter v_i. In this environment she can interact with all three authorities without fear that any adversary can eavesdrop or interfere.
2. Alice proves to the authorities that she controls her wallet (e.g. by signing a challenge message with the wallet's private key), and the authorities associate her wallet address to v_i in their respective voters lists.
3. $\mathcal{A}_1$ proves to Alice with a ZKP the knowledge of the exponent x_i' corresponding to the value $g^{x_i'}$, that had publicly decommitted at the end of the setup phase;
4. $\mathcal{A}_2$ proves to Alice with a ZKP the knowledge of the exponent x_i'' corresponding to the value $g^{x_i''}$, that had publicly decommitted at the end of the setup phase;
5. chooses, for every $i \in [N]$, a random subset $V_i \subset [M]$ such that its cardinality is exactly P, then sets:
$$\sigma_{i,l} = \begin{cases} k \iff l \in V_i \\ \lambda \iff l \notin V_i \end{cases} \tag{11}$$
i.e. the random choice of the V_i determines which tokens will be valid and which will be fake;
6. $\mathcal{A}_0$ takes the (publicly available) values $g^{x_i'}$ and $g^{x_i''}$ and creates the step 0 of the ballot $\bar{b}_{0,i} = (\bar{b}_{0,i,l})_{l\in[M]}$ where:
$$\bar{b}_{0,i,l} = \left(g^{\sigma_{i,l}} \cdot g^{x_i'} \cdot g^{x_i''}\right)^{\bar{z}_{i,l}} \tag{12}$$
$$= g^{\bar{z}_{i,l}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{13}$$

and sends to $\mathcal{A}_1$ the initial ballot $\bar{b}_{0,i}$, and sends to Alice $\bar{b}_{0,i}$ and V_i.

7. $\mathcal{A}_0$ proves the correctness of computations with the Schnorr ZKP presented in Section 2.2:
   (a) First $\mathcal{A}_0$ proves that the $g^{\bar{z}_{i,l}x'_i}$ are correct using:
   $$\omega = \bar{z}_{i,l}, \quad u = g, \quad z = g^{\bar{z}_{i,l}}, \quad \bar{u} = g^{x'_i}, \quad \bar{z} = g^{\bar{z}_{i,l}x'_i} \quad \forall l \in [M]. \tag{14}$$
   (b) Then $\mathcal{A}_0$ proves that the $g^{\bar{z}_{i,l}x''_i}$ are correct using:
   $$\omega = \bar{z}_{i,l}, \quad u = g, \quad z = g^{\bar{z}_{i,l}}, \quad \bar{u} = g^{x''_i}, \quad \bar{z} = g^{\bar{z}_{i,l}x''_i} \quad \forall l \in [M]. \tag{15}$$
   (c) Finally $\mathcal{A}_0$ proves that the $g^{\bar{z}_{i,l}\sigma_{i,l}}$ are correct using:
   $$\omega = k, \quad u = g, \quad z = g^{k}, \quad \bar{u} = g^{\bar{z}_{i,l}}, \quad \bar{z} = g^{\bar{z}_{i,l}k} \qquad \forall l \in V_i, \tag{16}$$
   and:
   $$\omega = \lambda, \quad u = g, \quad z = g^{\lambda}, \quad \bar{u} = g^{\bar{z}_{i,l}}, \quad \bar{z} = g^{\bar{z}_{i,l}\lambda} \quad \forall l \in [M] \setminus V_i. \tag{17}$$
8. $\mathcal{A}_1$ computes the step 1 of the ballot $\bar{b}_{1,i} = (\bar{b}_{1,i,l})_{l\in[M]}$ where:
   $$\bar{b}_{1,i,l} = \left(\bar{b}_{0,i,l}\right)^{z'_{i,l}} \tag{18}$$
   $$= g^{z_{i,l}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{19}$$
   and sends it to Alice and to $\mathcal{A}_2$.
9. $\mathcal{A}_1$ proves that the $\bar{b}_{1,i,l}$ are correct using:
   $$\omega = z'_{i,l}, \quad u = g, \quad z = g^{z'_{i,l}}, \quad \bar{u} = \bar{b}_{0,i,l}, \quad \bar{z} = \bar{b}_{1,i,l} \qquad \forall l \in [M]. \tag{20}$$
10. $\mathcal{A}_2$ chooses uniformly at random a permutation $\pi_i \in \mathrm{Sym}([M])$ and computes the step 2 of the ballot $\bar{b}_{2,i} = (\bar{b}_{2,i,l})_{l\in[M]}$ where:
    $$\bar{b}_{2,i,l} = \left(\bar{b}_{1,i,l}\right)^{y''_{i,\pi_i^{-1}(l)}} \tag{21}$$
    $$= g^{z_{i,l}y''_{i,\pi_i^{-1}(l)}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{22}$$
    and sends it to Alice and to $\mathcal{A}_0$, π_i is sent to Alice and $\mathcal{A}_1$.
11. $\mathcal{A}_2$ proves that the $\bar{b}_{2,i,l}$ are correct using:
    $$\omega = y''_{i,\pi_i^{-1}(l)}, \quad u = g, \quad z = g^{y''_{i,\pi_i^{-1}(l)}}, \quad \bar{u} = \bar{b}_{1,i,l}, \quad \bar{z} = \bar{b}_{2,i,l} \qquad \forall l \in [M]. \tag{23}$$
12. $\mathcal{A}_0$ computes the step 3 of the ballot $\bar{b}_{3,i} = (\bar{b}_{3,i,l})_{l\in[M]}$ where:
    $$\bar{b}_{3,i,l} = \left(\bar{b}_{2,i,l}\right)^{\frac{1}{\bar{z}_{i,l}}} \tag{24}$$
    $$= g^{z'_{i,l}y''_{i,\pi_i^{-1}(l)}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{25}$$
    and sends it to Alice and to $\mathcal{A}_1$.

13. $\mathcal{A}_0$ proves that the $\bar{b}_{3,i,l}$ are correct using:

$$\omega = \frac{1}{\bar{z}_{i,l}}, \quad u = g^{\bar{z}_{i,l}}, \quad z = g, \quad \bar{u} = \bar{b}_{2,i,l}, \quad \bar{z} = \bar{b}_{3,i,l} \qquad \forall l \in [M]. \tag{26}$$

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

$$b_{i,l} = \left(\bar{b}_{3,i,\pi_i(l)}\right)^{\frac{y'_{i,l}}{z'_{i,\pi_i(l)}}} \tag{27}$$

$$= g^{y_{i,l}(\sigma_{i,\pi_i(l)}+x_i)} \qquad \forall l \in [M] \tag{28}$$

and sends it to Alice and her wallet (where the value becomes final and public).

15. $\mathcal{A}_1$ proves that the $b_{i,l}$ are correct using:

$$\omega = \frac{y'_{i,l}}{z'_{i,\pi_i(l)}}, \quad u = g^{z'_{i,\pi_i(l)}}, \quad z = g^{y'_{i,l}}, \quad \bar{u} = \bar{b}_{3,i,\pi_i(l)}, \quad \bar{z} = b_{i,l} \qquad \forall l \in [M]. \tag{29}$$

Note that Alice, thanks to the proofs and the knowledge of the intermediate values, knows which ones are a valid token (the ones with $\sigma_{i,l} = k$), but thanks to the random choices of V_i and π_i the authorities cannot distinguish the tokens unless they collude.

Voting Phase Voters express their preferences by sending the valid tokens to their chosen candidates, and the fake tokens to the other candidates. The M *v-tokens* are sent with a single transaction on the blockchain to the respective candidates. As stated in Section 2.4, blockchain rules allow only votes that send one token to each candidate, prevent voters to vote twice, and guarantee that only registered voters can cast their ballots. Each voter receives the receipt of the vote (which basically is the insertion of the transaction in the blockchain), and the assumed properties of the blockchain guarantee that no vote is changed or deleted.

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* $y_{i,l}$ 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. Note that the voting addresses can be seen by everyone in the blockchain, and since the authorities have registered the association of these addresses to the ids v_i, they can correctly process each token, but they do not know the real identities of the corresponding person.

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 $b_{i,\phi_i(l)}$ that the voter v_i sent to the candidate C_l. Then, for every $i \in [T], l \in [M]$, the authorities process the token $b_{i,\phi_i(l)}$ by performing the following steps:

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

$$\bar{t}_{l,i} = \left(b_{i,\phi_i(l)}\right)^{\frac{\alpha'_l}{y'_{i,\phi_i(l)}}} \tag{30}$$
$$= g^{\alpha'_l y''_{i,\phi_i(l)}(\sigma_{i,\pi_i(\phi_i(l))}+x_i)}, \tag{31}$$

   and registers it on the blockchain.
2. Any observer could ask for a proof that this computation is correct. $\mathcal{A}_1$ proves that $\bar{t}_{l,i}$ is correct using:

$$\omega = \frac{\alpha'_l}{y'_{i,\phi_i(l)}}, \qquad u = g^{y'_{i,\phi_i(l)}}, \qquad z = g^{\alpha'_l}, \qquad \bar{u} = b_{i,\phi_i(l)}, \qquad \bar{z} = \bar{t}_{l,i}. \tag{32}$$

3. $\mathcal{A}_2$ then computes the final vote $t_{l,i}$ as:

$$t_{l,i} = (\bar{t}_{l,i})^{\frac{\alpha''_l}{y''_{i,\phi_i(l)}}} \tag{33}$$
$$= g^{\alpha_l(\sigma_{i,\pi_i(\phi_i(l))}+x_i)}, \tag{34}$$

   and registers it on the blockchain.
4. Again, any observer could ask for a proof that this computation is correct. $\mathcal{A}_2$ proves that $t_{l,i}$ is correct using:

$$\omega = \frac{\alpha''_l}{y''_{i,\phi_i(l)}}, \qquad u = g^{y''_{i,\phi_i(l)}}, \qquad z = g^{\alpha''_l}, \qquad \bar{t} = b_{l,i}, \qquad \bar{z} = t_{l,i}. \tag{35}$$

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 fake tokens given to the l-th candidate. Clearly $T = R_l + F_l \quad \forall l \in [M]$. The count R_l can be computed with the following steps:

1. $\mathcal{A}_1$ computes and publishes $g^{\alpha_l} = \left(g^{\alpha''_l}\right)^{\alpha'_l}$. Note that the same could be done by $\mathcal{A}_2$ as: $\left(g^{\alpha'_l}\right)^{\alpha''_l}$.
   The correctness can be proved by $\mathcal{A}_1$ using:

$$\omega = \alpha'_l, \qquad u = g, \qquad z = g^{\alpha'_l}, \qquad \bar{u} = g^{\alpha''_l}, \qquad \bar{z} = g^{\alpha_l}, \tag{36}$$

   and by $\mathcal{A}_2$ using:

$$\omega = \alpha''_l, \qquad u = g, \qquad z = g^{\alpha''_l}, \qquad \bar{u} = g^{\alpha'_l}, \qquad \bar{z} = g^{\alpha_l}. \tag{37}$$

   A practical approach could be that $\mathcal{A}_1$ publishes half of the values, and $\mathcal{A}_2$ the other half.

2. $\mathcal{A}_0$ computes and publishes $g^{\alpha_l k} = (g^{\alpha_l})^k$ and $g^{\alpha_l \lambda} = (g^{\alpha_l})^\lambda$. then proves that $g^{\alpha_l k}$ is correct using:
$$\omega = k, \qquad u = g, \qquad z = g^k, \qquad \bar{u} = g^{\alpha_l}, \qquad \bar{z} = g^{\alpha_l k}, \tag{38}$$
and that $g^{\alpha_l \lambda}$ is correct using:
$$\omega = \lambda, \qquad u = g, \qquad z = g^\lambda, \qquad \bar{u} = g^{\alpha_l}, \qquad \bar{z} = g^{\alpha_l \lambda}. \tag{39}$$
3. $\mathcal{A}_1$ computes $\sum_{i=1}^T x'_i$, and publishes $g^{\alpha_1 \sum_{i=1}^T x'_i}$.
4. Note that any observer can compute $g^{\sum_{i=1}^T x'_i} = \prod_{i=1}^T g^{x'_i}$, and then ask for a proof that the authority's computations are correct.
$\mathcal{A}_1$ proves that $g^{\alpha_l \sum_{i=1}^T x'_i}$ is correct using:
$$\omega = \sum_{i=1}^T x'_i, \quad u = g, \quad z = g^{\sum_{i=1}^T x'_i}, \quad \bar{u} = g^{\alpha_l}, \quad \bar{z} = g^{\alpha_l \sum_{i=1}^T x'_i}. \tag{40}$$
5. Similarly, $\mathcal{A}_2$ computes $\sum_{i=1}^T x''_i$ and publishes $g^{\alpha_1 \sum_{i=1}^T x''_i}$.
6. Again, any observer can compute $g^{\sum_{i=1}^T x''_i} = \prod_{i=1}^T g^{x''_i}$, and then ask for a proof that the authority's computation is correct.
$\mathcal{A}_2$ proves that $g^{\alpha_l \sum_{i=1}^T x''_i}$ is correct using:
$$\omega = \sum_{i=1}^T x''_i, \qquad u = g, \quad z = g^{\sum_{i=1}^T x''_i}, \quad \bar{u} = g^{\alpha_l}, \quad \bar{z} = g^{\alpha_l \sum_{i=1}^T x''_i}. \tag{41}$$
7. Note that any observer can compute the value:
$$g^{\alpha_l (\sum_{i=1}^T x_i + R_l k + F_l \lambda)} = \prod_{i=1}^T t_{l,i}. \tag{42}$$
Given that:
$$g^{\alpha_l \sum_{i=1}^T x_i} = g^{\alpha_l \sum_{i=1}^T (x'_i + x''_i)} = g^{\alpha_l \sum_{i=1}^T x'_i} \cdot g^{\alpha_l \sum_{i=1}^T x'_i}, \tag{43}$$
anyone can compute:
$$\left(g^{\alpha_l k}\right)^{R_l} \cdot \left(g^{\alpha_l \lambda}\right)^{F_l} = \left(g^{\alpha_l \sum_{i=1}^T x_i}\right)^{-1} \cdot g^{\alpha_l (\sum_{i=1}^T x_i + R_l k + F_l \lambda)}. \tag{44}$$
8. Finally, note that now R_l and F_l can be easily computed by brute force. In fact, 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, and the number of valid and fake votes must sum up to the number of actual voters T, so the effort is linear in the number of actual votes.

4 Security Analysis

The goal is to prove that an adversary cannot distinguish between valid and fake *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, but they are randomly assigned to the two voters. The adversary wins the security game if it guesses correctly which voter expressed which preferences.

4.1 Security Model

The security of the protocol will be proven in terms of vote indistinguishability (VI), as detailed in Definition 10.

The security of the protocol will be proved in the presence of a malicious authority, so the simulator in the proof will take on the roles of the two honest authorities and 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 ZKPs, so the protocol does not abort. This is a reasonable assumption considering the application context, however it is not necessary to attain the 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 once the protocol has correctly terminated).

Definition 9 (Security Game). *The security game for the election protocol proceeds as follows:*

- ***Init.*** *The adversary $\mathcal{A}$ chooses the authority and the $N-2$ users that it will control. This means that the adversary knows which are the valid and fake* v-tokens *of these users. The remaining two users 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.*** *$\mathcal{A}$ selects two distinct sets of preferences $\tilde{P}_0 \neq \tilde{P}_1$, with $\tilde{P}_i \subset [M]$, $\#\tilde{P}_i = P$ for $i = 0, 1$. $\mathcal{C}$ flips a random coin $\mu \in \{0, 1\}$ to determine which preference set the first free voter will use, i.e. $P_1 = \tilde{P}_\mu$ (the second one uses the set $P_2 = \tilde{P}_{\mu\oplus 1}$). Then, $\mathcal{C}$ constructs two random ballot assignment maps $\tilde{\phi}_1, \tilde{\phi}_2 : [M] \longrightarrow [M]$ such that $\tilde{\phi}_i(l)$ is a valid token if and only if $l \in P_i$, for $i = 1, 2$. Finally, $\mathcal{C}$ votes by sending to the candidate C_l the $\tilde{\phi}_1(l)$-th token of the first free voter and the $\tilde{\phi}_2(l)$-th token of the second free voter, $\forall l \in [M]$.*
- ***Phase 2.*** *The adversary votes with some or all of the voters it controls which did not vote in Phase 1.*

– ***Phase 3.*** *$\mathcal{A}$ and $\mathcal{C}$ run the* Tallying *phase of the protocol, and the election result is published. Note that the adversary can request the ZKPs of the correctness of the computations performed by $\mathcal{C}$, and vice versa.*
– ***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 10 (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 9), there exists a negligible function η such that:*

$$\mathbb{P}[\mu' = \mu] \leq \frac{1}{2} + \eta(\theta). \tag{45}$$

In the following theorem we prove our voting protocol VI-secure under the DDH assumption (Definition 3) in the security game defined above.

Theorem 1. *If the DDH assumption of Definition 3 holds, then the protocol described in Section 3.1 is VI-secure, as per Definition 10.*

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 starts taking in a DDH challenge:

$$(g, A = g^a, B = g^b, T), \tag{46}$$

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

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

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

$$x''_i = \tilde{x}_i + (-1)^i b, \qquad \forall i \in [2], \tag{47}$$

$$\alpha'_l = a \cdot \tilde{\alpha}_l \qquad \forall l \in [M], \tag{48}$$

$$y'_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M], \tag{49}$$

$$z'_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M]. \tag{50}$$

$\mathcal{S}$ chooses the other values for the authorities $\mathcal{A}_1$ and $\mathcal{A}_2$ following the protocol. Notice that in the improbable case where $a = 0$ the DDH problem is easily solvable ($g^a = g^{ab} = 1$), otherwise since a and b come from an uniform distribution, then also these implicit values are uniform distributed, so the choices of the simulator are indistinguishable from a real protocol execution. Note also that $\mathcal{S}$ can compute all the values $g^{x''_i}$, $g^{\alpha'_l}$, $g^{y'_{i,l}}$, $g^{z'_{i,l}}$, either normally (when the parameter has been explicitly chosen) or as follows:

$$g^{x''_i} = g^{\tilde{x}_i} \cdot B^{(-1)^i} \qquad \forall i \in [2], \tag{51}$$

$$g^{\alpha'_l} = A^{\tilde{\alpha}_l} \qquad \forall l \in [M], \tag{52}$$

$$g^{y'_{i,l}} = A^{\tilde{y}_{i,l}} \qquad \forall i \in [2], l \in [M], \tag{53}$$

$$g^{z'_{i,l}} = A^{\tilde{z}_{i,l}} \qquad \forall i \in [2], l \in [M]. \tag{54}$$

Therefore, $\mathcal{S}$ can perfectly simulate the setup phase.

– **Registrar Phase**. For the voters 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 $i \in [2]$ the simulation is carried on as follows:
  1. $\mathcal{S}$ proves the knowledge of x'_i normally, and simulates the ZKP for x''_i, as per Corollary 1.
  2. $\mathcal{A}$ computes the initial step of the ballot $\bar{b}_{0,i}$ on behalf of $\mathcal{A}_0$ and proves its correctness. From these proofs $\mathcal{S}$ extracts the values of k, λ, and $\bar{z}_{i,l}$ $\forall l \in [M]$, as per Lemma 1. Moreover, since $\mathcal{A}_0$ communicates the set of indexes of valid tokens V_i to the voter v_i (that is controlled by the simulator), $\mathcal{S}$ can reconstruct the values of the $\sigma_{i,l} \forall l \in [M]$.
  3. $\mathcal{S}$ computes step 1 of the ballot $\bar{b}_{1,i} = (\bar{b}_{1,i,l})_{l\in[M]}$ as:

$$\bar{b}_{1,i,l} = A^{\bar{z}_{i,l}\tilde{z}_{i,l}(\sigma_{i,l}+x'_i+\tilde{x}_i)} \cdot T^{\bar{z}_{i,l}\tilde{z}_{i,l}(-1)^i} \tag{55}$$

$$\overset{*}{=} g^{\bar{z}_{i,l}a\tilde{z}_{i,l}\left(\sigma_{i,l}+x'_i+\tilde{x}_i+(-1)^i b\right)} \tag{56}$$

$$= g^{z_{i,l}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{57}$$

  where Equation (56) holds if and only if $T = g^{ab}$ in the DDH challenge. Notice that, since it controls the voter v_i, $\mathcal{S}$ does not have to simulate the ZKPs.
  4. $\mathcal{S}$ 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.
  5. Finally $\mathcal{S}$ computes the final ballot $b_i = (b_{i,l})_{l\in[M]}$ as:

$$b_{i,l} = A^{\tilde{y}_{i,l}y''_{i,l}(\sigma_{i,\pi_i(l)}+x'_i+\tilde{x}_i)} \cdot T^{\tilde{y}_{i,l}y''_{i,l}(-1)^i} \tag{58}$$

$$\overset{*}{=} g^{a\tilde{y}_{i,l}y''_{i,l}\left(\sigma_{i,\pi_i(l)}+x'_i+\tilde{x}_i+(-1)^i b\right)} \tag{59}$$

$$= g^{y_{i,l}(\sigma_{i,\pi_i(l)}+x_i)} \qquad \forall l \in [M] \tag{60}$$

  where again Equation (59) holds if and only if $T = g^{ab}$ in the DDH challenge, and the ZKPs can be omitted.

– **Voting**: Phase 1, the Challenge, and Phase 2 are performed as described in Definition 9.

– **Tallying**. Without loss of generality, suppose that only the v_i with $i \in [T]$ have voted. For $l \in [M]$, $\mathcal{S}$ 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
$$\frac{\alpha'_l}{y'_{i,\tilde{\phi}_i(l)}} = \frac{a\tilde{\alpha}_l}{a\tilde{y}_{i,\tilde{\phi}_i(l)}} = \frac{\tilde{\alpha}_l}{\tilde{y}_{i,\tilde{\phi}_i(l)}} \qquad \forall l \in [M], \tag{61}$$
and these values are known to $\mathcal{S}$.
2. $\mathcal{S}$ computes and publishes the values g^{α_l} as:
$$g^{\alpha_l} = A^{\tilde{\alpha}_l \alpha''_l} \qquad \forall l \in [M], \tag{62}$$
and simulates the proofs of correctness.
3. Finally note that $\mathcal{S}$ can compute:
$$\sum_{i=1}^{T} x''_i = \tilde{x}_1 - b + \tilde{x}_2 + b + \sum_{i=3}^{T} x''_i = \tilde{x}_1 + \tilde{x}_2 + \sum_{i=3}^{T} x''_i, \tag{63}$$
so for the rest of the tallying phase $\mathcal{S}$ can follow the protocol without any problem.

– **Guess** Eventually the adversary will output a guess μ' of the coin flip performed by $\mathcal{S}$ during the Challenge. The simulator then outputs 0 to guess that $T = g^{ab}$ if $\mu' = \mu$, otherwise it outputs 1 to indicate that T is a random group element $R \in \mathbb{G}$.

Now we consider the case in which the adversary controls $\mathcal{A}_1$. Focusing on the differences from the previous case, the simulation proceeds as follows.

– **Setup**. $\mathcal{S}$ chooses uniformly at random in $\mathbb{Z}_p^*$ the values $\tilde{x}_i$ for $i \in [2]$, $\tilde{\alpha}_l$ for $l \in [M]$, and $\tilde{y}_{i,l}$, $\tilde{z}_{i,l}$ for $i \in [2], l \in [M]$, and implicitly sets:
$$x''_i = \tilde{x}_i + (-1)^i b, \qquad \forall i \in [2], \tag{64}$$
$$\alpha''_l = a \cdot \tilde{\alpha}_l \qquad \forall l \in [M], \tag{65}$$
$$y''_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M], \tag{66}$$
$$\bar{z}_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M]. \tag{67}$$
$\mathcal{S}$ chooses the other values for the authorities $\mathcal{A}_0$ and $\mathcal{A}_2$ following the protocol. As before, the choices of the simulator are indistinguishable from a real protocol execution. Note also that $\mathcal{S}$ can compute all the values $g^{x''_i}$, $g^{\alpha''_l}$, $g^{y''_{i,l}}$, $g^{\bar{z}_{i,l}}$, either normally (when the parameter has been explicitly chosen) or as follows:
$$g^{x''_i} = g^{\tilde{x}_i} \cdot B^{(-1)^i} \qquad \forall i \in [2], \tag{68}$$
$$g^{\alpha''_l} = A^{\tilde{\alpha}_l} \qquad \forall l \in [M], \tag{69}$$
$$g^{y''_{i,l}} = A^{\tilde{y}_{i,l}} \qquad \forall i \in [2], l \in [M], \tag{70}$$
$$g^{\bar{z}_{i,l}} = A^{\tilde{z}_{i,l}} \qquad \forall i \in [2], l \in [M]. \tag{71}$$
Therefore, $\mathcal{S}$ can perfectly simulate the setup phase.

– **Registrar Phase**. For $i \in [2]$ the simulation is carried on as follows:
1. $\mathcal{A}$ proves the knowledge of x'_i on behalf of $\mathcal{A}_1$, and from this proof $\mathcal{S}$ extracts the proven value.
2. $\mathcal{S}$ simulates the ZKP for x''_i on behalf of $\mathcal{A}_2$.
3. $\mathcal{S}$ computes on behalf of $\mathcal{A}_0$ the initial step of the ballot $\bar{b}_{0,i} = (\bar{b}_{0,i,l})_{l\in[M]}$ as:

$$\begin{aligned} \bar{b}_{0,i,l} &= A^{\tilde{z}_{i,l}(\sigma_{i,l}+x'_i+\tilde{x}_i)} \cdot T^{\tilde{z}_{i,l}(-1)^i} && (72)\\ &\overset{*}{=} g^{a\tilde{z}_{i,l}\left(\sigma_{i,l}+x'_i+\tilde{x}_i+(-1)^i b\right)} && (73)\\ &= g^{\bar{z}_{i,l}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] && (74)\end{aligned}$$

where Equation (73) holds if and only if $T = g^{ab}$ in the DDH challenge. Notice that, since it controls the voter v_i, $\mathcal{S}$ does not have to simulate the ZKPs.
4. $\mathcal{A}$ performs the step 1 and proves its correctness. From the proof $\mathcal{S}$ extracts the value of $z'_{i,l}$.
5. Since the output of step 2 is only seen by v_i and $\mathcal{A}_0$ (both controlled by the simulator), $\mathcal{S}$ does not have to simulate it, and computes directly the step 3 $\bar{b}_{3,i} = (\bar{b}_{3,i,l})_{l\in[M]}$ (on behalf of $\mathcal{A}_0$) as:

$$\begin{aligned} \bar{b}_{3,i,l} &= A^{z'_{i,l}\tilde{y}_{i,\pi_i^{-1}(l)}(\sigma_{i,l}+x'_i+\tilde{x}_i)} \cdot T^{z'_{i,l}\tilde{y}_{i,\pi_i^{-1}(l)}(-1)^i} && (75)\\ &\overset{*}{=} g^{z'_{i,l}a\tilde{y}_{i,\pi_i^{-1}(l)}\left(\sigma_{i,l}+x'_i+\tilde{x}_i+(-1)^i b\right)} && (76)\\ &= g^{z'_{i,l}y''_{i,\pi_i^{-1}(l)}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] && (77)\end{aligned}$$

where again Equation (76) holds if and only if $T = g^{ab}$ in the DDH challenge, and the ZKPs can be omitted.
6. $\mathcal{A}$ computes the final ballot and proves its correctness. From the proof $\mathcal{S}$ extracts the values $\frac{y'_{i,l}}{z'_{i,\pi_i(l)}}$, from which it derives the values $y'_{i,l}$ since the $z'_{i,l}$ and π_i are already known.

– **Tallying**. Without loss of generality, suppose that only the v_i with $i \in [T]$ have voted. For $l \in [M]$, $\mathcal{S}$ carries on the simulation as follows:
1. $\mathcal{A}$ computes the preliminary votes on behalf of $\mathcal{A}_1$ and proves their correctness. From the proofs $\mathcal{S}$ extracts the values $\frac{\alpha'_l}{y'_{i,\tilde{\phi}_i(l)}}$, from which it derives the values α'_l since the $y'_{i,l}$ and $\tilde{\phi}_i$ are already known.
2. $\mathcal{S}$ computes and proves the correctness of the final votes on behalf of $\mathcal{A}_2$ following the protocol without problems. In fact, for $i \in [2]$, we have that

$$\frac{\alpha''_l}{y''_{i,\tilde{\phi}_i(l)}} = \frac{a\tilde{\alpha}_l}{a\tilde{y}_{i,\tilde{\phi}_i(l)}} = \frac{\tilde{\alpha}_l}{\tilde{y}_{i,\tilde{\phi}_i(l)}} \qquad \forall l \in [M], \tag{78}$$

and these values are known to $\mathcal{S}$.

3. $\mathcal{S}$ may compute and publish the values g^{α_l} as:
$$g^{\alpha_l} = A^{\alpha'_l \tilde{\alpha}_l} \qquad \forall l \in [M], \tag{79}$$
and simulates the proofs of correctness.
4. Finally note that $\mathcal{S}$ can compute:
$$\sum_{i=1}^{T} x''_i = \tilde{x}_1 + \tilde{x}_2 + \sum_{i=3}^{T} x''_i, \tag{80}$$
so for the rest of the tallying phase $\mathcal{S}$ can follow the protocol without any problem.

Finally we consider the case in which the adversary controls $\mathcal{A}_2$. Focusing on the differences from the previous cases, the simulation proceeds as follows.

– **Setup**. $\mathcal{S}$ chooses uniformly at random in $\mathbb{Z}_p^*$ the values $\tilde{x}_i$ for $i \in [2]$, $\tilde{\alpha}_l$ for $l \in [M]$, and $\tilde{y}_{i,l}$ for $i \in [2], l \in [M]$, and implicitly sets:
$$x'_i = \tilde{x}_i + (-1)^i b, \qquad \forall i \in [2], \tag{81}$$
$$\alpha'_l = a \cdot \tilde{\alpha}_l \qquad \forall l \in [M], \tag{82}$$
$$y'_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M], \tag{83}$$
$$z'_{i,l} = a \cdot \tilde{y}_{i,l}, \qquad \forall i \in [2], l \in [M]. \tag{84}$$
$\mathcal{S}$ chooses the other values for the authorities $\mathcal{A}_0$ and $\mathcal{A}_1$ following the protocol. Once again the choices of the simulator are indistinguishable from a real protocol execution. Note also that $\mathcal{S}$ can compute all the values $g^{x'_i}$, $g^{\alpha'_l}$, $g^{y'_{i,l}}$, $g^{z'_{i,l}}$, either normally (when the parameter has been explicitly chosen) or as follows:
$$g^{x'_i} = g^{\tilde{x}_i} \cdot B^{(-1)^i} \qquad \forall i \in [2], \tag{85}$$
$$g^{\alpha'_l} = A^{\tilde{\alpha}_l} \qquad \forall l \in [M], \tag{86}$$
$$g^{y'_{i,l}} = A^{\tilde{y}_{i,l}} \qquad \forall i \in [2], l \in [M], \tag{87}$$
$$g^{z'_{i,l}} = A^{\tilde{z}_{i,l}} \qquad \forall i \in [2], l \in [M]. \tag{88}$$
Therefore, $\mathcal{S}$ can perfectly simulate the setup phase.
– **Registrar Phase**. For $i \in [2]$ the simulation is carried on as follows:
1. $\mathcal{S}$ and simulates the ZKP for x'_i on behalf of $\mathcal{A}_1$.
2. $\mathcal{A}$ proves the knowledge of x''_i on behalf of $\mathcal{A}_2$, and from this proof $\mathcal{S}$ extracts the proven value.
3. Since the output of step 0 is only seen by v_i and $\mathcal{A}_1$ (both controlled by the simulator), $\mathcal{S}$ does not have to simulate it, and computes directly the step 1 $\bar{b}_{1,i} = (\bar{b}_{1,i,l})_{l\in[M]}$ as:
$$\bar{b}_{1,i,l} = A^{\bar{z}_{i,l}\tilde{z}_{i,l}(\sigma_{i,l}+\tilde{x}_i+x''_i)} \cdot T^{\bar{z}_{i,l}\tilde{z}_{i,l}(-1)^i} \tag{89}$$
$$\overset{*}{=} g^{\bar{z}_{i,l}a\tilde{z}_{i,l}\left(\sigma_{i,l}+\tilde{x}_i+(-1)^i b+x''_i\right)} \tag{90}$$
$$= g^{z_{i,l}(\sigma_{i,l}+x_i)} \qquad \forall l \in [M] \tag{91}$$

where Equation (90) holds if and only if $T = g^{ab}$ in the DDH challenge. Notice that, since it controls the voter v_i, $\mathcal{S}$ does not have to simulate the ZKPs.

4. $\mathcal{A}$ performs the step 2 and proves its correctness. From the proof $\mathcal{S}$ extracts the value of $y''_{i,l}$. Notice also that $\mathcal{A}$ gives the permutations π_i to $\mathcal{S}$.
5. Since the output of step 3 is only seen by v_i and $\mathcal{A}_1$ (both controlled by the simulator), $\mathcal{S}$ does not have to simulate it, and computes directly the final ballot $b_i = (b_{i,l})_{l\in[M]}$ (on behalf of $\mathcal{A}_1$) as:

$$b_{i,l} = A^{\tilde{y}_{i,l}y''_{i,l}(\sigma_{i,\pi_i(l)}+\tilde{x}_i+x''_i)} \cdot T^{\tilde{y}_{i,l}y''_{i,l}(-1)^i} \tag{92}$$

$$\overset{*}{=} g^{a\tilde{y}_{i,l}y''_{i,l}\left(\sigma_{i,\pi_i(l)}+\tilde{x}_i+(-1)^i b+x''_i\right)} \tag{93}$$

$$= g^{y_{i,l}(\sigma_{i,\pi_i(l)}+x_i)} \qquad \forall l \in [M] \tag{94}$$

where again Equation (93) holds if and only if $T = g^{ab}$ in the DDH challenge, and the ZKPs can be omitted.

– **Tallying**. Without loss of generality, suppose that only the v_i with $i \in [T]$ have voted. For $l \in [M]$, $\mathcal{S}$ carries on the simulation as follows:
  1. $\mathcal{S}$ computes and proves the correctness of the preliminary votes on behalf of $\mathcal{A}_1$ following the protocol without problems. In fact, for $i \in [2]$, we have that

$$\frac{\alpha'_l}{y'_{i,\tilde{\phi}_i(l)}} = \frac{a\tilde{\alpha}_l}{a\tilde{y}_{i,\tilde{\phi}_i(l)}} = \frac{\tilde{\alpha}_l}{\tilde{y}_{i,\tilde{\phi}_i(l)}} \qquad \forall l \in [M], \tag{95}$$

  and these values are known to $\mathcal{S}$.
  2. $\mathcal{A}$ computes the final votes on behalf of $\mathcal{A}_2$ and proves their correctness. From the proofs $\mathcal{S}$ extracts the values $\frac{\alpha''_l}{y''_{i,\tilde{\phi}_i(l)}}$, from which it derives the values α''_l since the $y''_{i,l}$ and $\tilde{\phi}_i$ are already known.
  3. $\mathcal{S}$ may compute and publish the values g^{α_l} as:

$$g^{\alpha_l} = A^{\tilde{\alpha}_l\alpha''_l} \qquad \forall l \in [M], \tag{96}$$

  and simulates the proofs of correctness.
  4. Finally note that $\mathcal{S}$ can compute:

$$\sum_{i=1}^{T} x'_i = \tilde{x}_1 + \tilde{x}_2 + \sum_{i=3}^{T} x'_i, \tag{97}$$

  so for the rest of the tallying phase $\mathcal{S}$ can follow the protocol without any problem.

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

$$\mathbb{P}[\mathcal{S}(g, A, B, T = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{98}$$

On the contrary, when T 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 tallying is always correct. Since the security game is structured in such a way that the tallying and the token and votes of the other voters (i.e. the values where T is not used in the computation by $\mathcal{S}$) do not give any information about the coin flip μ, we have that:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = R) = 0] = \frac{1}{2}. \tag{99}$$

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

4.2 General properties of the protocol

As already stated in Section 2.5, the general properties of a vote system introduced in [11] can all be proved for the protocol described in Section 3.1 in the exact same way as for the two-candidates protocol, with the sole exception of vote-selling and coercion resistance, which are given by the following proposition.

Proposition 2 (Vote-Selling and Coercion Resistance). *If the DDH assumption holds, then the protocol is vote-selling and coercion resistant, as per Definitions 7 and 8.*

Proof. Thanks to Theorem 1, if the DDH assumption holds, the protocol has vote-indistinguishability and the only way to distinguish the proper votes is to distinguish valid and fake tokens. The voter v_i can do so with the additional information $\zeta_i = \{\pi_i(l) : l \in V_i\}$ (i.e. the the indexes of the valid tokens in the final ballot, randomly determined during the registrar phase). V_i is chosen by $\mathcal{A}_0$, π_i is chosen by $\mathcal{A}_2$ and communicated to $\mathcal{A}_1$, but supposing at most one malicious/corrupted authority and that the ballot generation is performed in a safe environment, then ζ_i is known only to the voter. Note that $\zeta_i \in M_P = \{X \subset [M] : \#X = P\}$ and every subset of $[M]$ of cardinality P is admissible and equiprobable as a possible ζ_i (since $V_i \in M_P$ and $\pi_i \in \mathrm{Sym}([M])$) are chosen uniformly at random), thus the voter can easily fabricate a value ζ_i' that possibly changes the vote to fake compliance with the coercer's choices. Thus, any value obtainable by a third party cannot certify that the voter expressed a particular preference, disallowing both vote-selling and vote-coercion. □

5 Conclusions

In this paper we have generalized the two-candidates-one-preference e-voting protocol of [11] into an M-candidates-P-preferences protocol. We have preserved the general construction which uses an underlying blockchain infrastructure and a system of ZKPs to ensure transparency and full auditability of the whole process. The protocol achieves also the same extensive security properties (proven under the classical Decisional Diffie-Hellman Assumption), including *coercion* and *vote-selling* resistance, while retaining receipts.

The same considerations about the auxiliary infrastructure still apply to the generalized protocol, such as using an ad-hoc blockchain to contain the transaction costs of casting ballots, and the hurdles of providing a safe and authenticated environment for the registrar phase which remain, however, an improvement over the traditional voting booths.

Our generalization introduces an additional authority, that is required in order to properly mask the multiple valid and fake tokens in each ballot. However, the generalization is sufficiently close to the original protocol, so the same considerations about the authorities still apply. In particular, they can perform the setup phase may be performed 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 (like a voting district). 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), and enhances the overall efficiency distributing the workload.

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. So, if Alice wants to cast a blank ballot, during the voting phase she will send the valid tokens to the dummy candidates and the fake tokens to the real candidates. Similarly, if she wants to express only $P' < P$ preferences, she will send P' valid tokens to the chosen candidates, and the remaining $P - P'$ valid tokens to any subset of the dummy candidates.

Note, however, that it is impossible to allow blank ballots but not partial votes. In fact, the design of the protocol specifically unlinks the tokens expressed by a single voter during the tallying. As a matter of fact, the vote-indistinguishability property that makes the protocol secure, also makes it is impossible to check if the tokens sent to the dummy candidates are either all valid or all fake, unless the authorities collude.

Acknowledgments The core of this work is derived from the second author's MSC thesis that would like to thank her supervisor, prof. Massimiliano Sala. The first author is a member of the INdAM Research group GNSAGA.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX security symposium. vol. 17, pp. 335–348 (2008)
2. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a

distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. pp. 1–15. ACM (2018)

3. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of computer and system sciences **37**(2), 156–189 (1988)
4. Gaudry, P., Golovnev, A.: Breaking the encryption scheme of the moscow internet voting system. In: International Conference on Financial Cryptography and Data Security. pp. 32–49. Springer (2020)
5. 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
6. Longo, R.: Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols. Ph.D. thesis, University Of Trento, Department of Mathematics (2018)
7. Longo, R., Podda, A.S., Saia, R.: Analysis of a consensus protocol for extending consistent subchains on the bitcoin blockchain. Computation **8**(3), 67 (2020)
8. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (2008)
9. Pawlak, M., Poniszewska-Marańda, A.: Trends in blockchain-based electronic voting systems. Information Processing & Management **58**(4), 102595 (2021)
10. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4**, 161–174 (1991). https://doi.org/10.1007/BF00196725
11. 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). https://doi.org/10.3934/amc.2021005
12. Specter, M.A., Koppel, J., Weitzner, D.: The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in us federal elections. Preprint available at: https://internetpolicy. mit. edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public. pdf (2020)
13. Specter, M.A., Koppel, J., Weitzner, D.: The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in us federal elections. In: 29th {USENIX} Security Symposium ({USENIX} Security 20). pp. 1535–1553 (2020)
14. Xiao, S., Wang, X.A., Wang, W., Wang, H.: Survey on blockchain-based electronic voting. In: Barolli, L., Nishino, H., Miwa, H. (eds.) Advances in Intelligent Networking and Collaborative Systems. pp. 559–567. Springer (2020)