Coercion-Resistant Blockchain-Based E-Voting Protocol

Chiara Spadafora[1], Riccardo Longo[2], and Massimiliano Sala[3]

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

Abstract. Coercion resistance is one of the most important features of a secure voting procedure. Because of the properties such as transparency, decentralization, and non-repudiation, blockchain is a fundamental technology of great interest in its own right, and it also has large potential when integrated into many other areas. Here we propose a decentralized e-voting protocol that is coercion-resistant and vote-selling resistant, while being also completely transparent and not receipt-free. We prove the security of the protocol under the standard DDH assumption.

1 Introduction

An election is a significant event in many democratic countries, however, traditional voting systems may be inefficient given the large number of areas and population involved in modern elections. Since the advent of Internet there has been interest in developing remote voting or e-voting [8], but with the development of blockchain technologies there has been a significant boost in this field, exploiting some nice properties of these constructions, such as transparency and non-repudiation [16]. Usually e-voting protocols are requested provide two properties: *ballot casting assurance*, where each voter gains personal assurance that their vote was correctly cast, and *universal verifiability*, where any observer can verify that all cast votes were properly tallied. A voting scheme is also requested to be robust and resistant 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.

In this paper we propose an e-voting protocol, which aims at providing resistance versus coercion and vote-selling, while giving ballot casting assurance (thanks to a receipt) to every voter. To the best of our knowledge the presence of both the receipt of the vote and the described resistance to malicious entities, all deployed on a blockchain infrastructure, is an innovative proposal.

Related Work. The research in the field of e-voting is constantly growing, with a lot of protocols proposed. *Civitas* [6] deals with coercion allowing voters to generate, with their designated private key, fake credentials and then erasing all the votes submitted through them. *Caveat Coercitor* [9] is a unique voting system that, instead of preventing coercion, allows it, while recording unforgeable evidence of voter-coercions. Indeed Caveat Coercitor outputs the evidence of the amount of suspicious voter-coercions that occurred in the elections. Observers can decide whether or not the outcome is valid based on the number of suspicious ballots. *Bingo Voting* [4] is a e-voting protocol that relies on a trusted random-number generator. Every voter receives a receipt for all the candidates, even for those it did not vote for. Fake votes are generated for every candidate and eliminated in tallying.

A more comprehensive comparison of voting protocols can be found in [10].

2 Preliminaries

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

2.1 Decisional Diffie-Hellman Assumption

We use the definition of the decisional Diffie–Hellman (DDH) problem and the relative hardness assumption given in [12].

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

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

to efficiently 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 clearly written as:

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

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

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

2.2 Zero Knowledge Proofs

Zero-knowledge proofs were first conceived in 1989 by Shafi Goldwasser, Silvio Micali, and Charles Rackoff [3] introducing the first zero-knowledge proof for a concrete problem: deciding *quadratic non-residues* modulo an integer. Together with the work of Laszlo Babai and Shlomo Moran [2], those two papers invented the interactive proof system, for which all five authors won the first *Gödel Prize* in 1993.

A zero-knowledge proof (ZKP) is a cryptographic proof which allows one party (the prover) to prove to another party (the verifier) the veracity of some statement, without revealing anything else to the verifier.

Equality of discrete logarithms One of the simplest proof of knowledge is due to Schnorr [14]: the proof of knowledge of a *discrete logarithm*. Here we present a variation of the Schnorr interactive protocol, similar to [15], that 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 finally let $z, \bar{z} \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:*

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

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

1. *The prover generates a random r and computes $t = u^r$ 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 t times, where t is polynomial in the length of p (that is the security parameter).*

Security considerations We now prove that Protocol 1 satisfies the following properties, under the DDH assumption.

- **Completeness.** To show that this protocol is correct, (namely that an honest prover can indeed convince a verifier to accept a true statement simply by following the prescribed protocol) it suffices to verify that the equations of steps 3 and 4 hold when s is computed correctly.
- **Soundness.** To show soundness (namely that even an arbitrarily malicious prover cannot convince the verifier to accept a false statement with more than negligible probability) first note that the prover can guess all t values of the challenges c only with probability 2^{-t} which is negligible. Therefore if the prover manages to complete a proof with more than negligible probability then, in at least one protocol repetition, the prover does not fail even when guessing wrong, i.e. it can answer both possible challenges correctly. This means that the prover can compute both $r + \omega$ and r, and therefore compute ω, but we were assuming that the prover did not know ω, hence the contradiction.
- **Zero-knowledge.** This is the property that the verifier cannot gain even a single bit of extra information other that the prover knows ω. To show that, we use a simulator $\mathcal{S}$ that takes in input $(u, z, \bar{u}, \bar{z})$ and can interact with a (possibly malicious) verifier V producing a view that is indistinguishable from a real one, as follows:
  1. $\mathcal{S}$ initialises the verifier V with $u, z, \bar{u}, \bar{z}$ and $i = 0$;
  2. $\mathcal{S}$ selects $c' \in \{0, 1\}$ at random;
  3. $\mathcal{S}$ selects $s \in \mathbb{Z}_p$ at random and sets $t = u^s \cdot z^{-c'}$, $\bar{t} = \bar{u}^s \cdot \bar{z}^{-c'}$;
  4. $\mathcal{S}$ gives $(t, \bar{t})$ and gets the challenge c;

5. If $c \neq c'$ $\mathcal{S}$ rewinds V and goes back to step 2 with the same i, otherwise it proceeds;
6. $\mathcal{S}$ gives s to V, since the check succeeds, if $i = t$ the proof successfully completes, otherwise $\mathcal{S}$ sets $i = i + 1$ and proceeds with the simulation repeating from step 2.

Note that this simulator runs in expected polynomial time since for every repetition the probability of guessing the correct c' is $\frac{1}{2}$, so the expected number of repetition needed to complete is $2t$ which is polynomial. Now let us suppose that there exists a V that can distinguish this simulation from a real protocol interaction. Note that this is equivalent to distinguishing whether the input tuple $(u, z, \bar{u}, \bar{z})$ satisfies Equation (3) for some $\omega \in \mathbb{Z}_p$, in fact when such ω exists then the view produced by $\mathcal{S}$ has the exact same distribution of a real protocol interaction. To complete our proof we use this distinguishing verifier V to solve the decisional Diffie-Hellman problem: given a challenge (g, A, B, T) we pass it as input of $\mathcal{S}$ with $u = g, z = A, \bar{u} = B, \bar{z} = T$, that corresponds to implicitly setting $\omega = a$ if $T = g^{ab}$. In this case $\mathcal{S}$ perfectly simulates the real protocol, otherwise if $T = g^r$ then the verifier could detect that it is a simulation and we can break DDH assumption.

2.3 Commitment Scheme

A commitment scheme [5] 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.

In our construction we use commitments to enhance the privacy of the voters during the election even towards authorities, and to prevent some malicious choice of parameters. However in our analysis we assume the honesty of the authorities, so commitments are only marginally involved in the proof. For this reason we do not specify the meaning of *infeasibility* in the aforementioned security properties, noting that a commitment scheme can achieve *perfect* (information theoretic) security in only one of the two properties, while the other is at most *computationally* secure.

2.4 Blockchain

The e-voting protocol we propose exploits the accountability and immutability properties of an underlying blockchain. With *blockchain* we mean a decentralised data structure with the following properties:

- **public:** the contents of the blockchain is publicly readable and examinable by anyone, in particular we assume that an attacker is not able to efficiently and indefinitely negate access to the blockchain or pass off a counterfeit (tampered) copy as the original one;
- **append-only:** the contents of the blockchain are immutable once published, but new data can be added afterwards, more specifically an attacker is not able to modify the blockchain so that the original status is not a prefix of the alteration.

Users send their proposals of new data to be included in the blockchain to the *miners*, a special subset of users that actively maintain the blockchain reaching a consensus (e.g. [13]) on what to append and publishing the updated status.

In our protocol the main users are the voters, while the miners are roughly equivalent to election officers, the data registered on the blockchain is how the ballots are cast (with suitable obfuscation to preserve privacy), in the form of *transactions* that transfer voting tokens to special accounts that represent the candidates.

Not every user of the blockchain is allowed to participate in the election, however we do not take in consideration how users are identified. We follow their actions only once they become "voters", that is, the protocols knows which blockchain addresses correspond to possible voters. To restrict the participation only to identified blockchain users, there are at least two possible ways:

- to use a *permissioned* blockchain[4] (so the internal PKI will guarantee the identification)
- to use a *permissionless* blockchain[5] with either a smart contract and/or an external oracle.

Finally we need that miners enforce some consistency rules:

- transactions are properly authorised by the user who owns the token, i.e. a user's tokens cannot be spent by anybody else;
- only valid votes are accepted and registered on the blockchain, i.e. users must spend both of their tokens together, and send them to different candidates (see Section 3);
- each token can be spent only once.

[4] We are implementing a prototype with HyperLedger Fabric [1].
[5] We are also implementing a prototype with Quadrans [7].

3 Protocol Description

This section presents our proposal for a remote e-voting protocol in an election with two candidates, based on blockchain technology.

The basic idea is that every voter owns two voting tokens (*v-tokens*): one is real, the other is fake. When voting, every voter expresses their preference assigning the real *v-token* to the chosen candidate and the fake one to the other. On the vote receipt both transactions will be displayed, but only for the voter this receipt will be understandable since no one else knows which *v-token* is real and which is fake. In the final tally the fake *v-tokens* are erased and the whole process is publicly auditable.

The aim of this voting system is to be fully verifiable, prevent coercion and vote selling while being almost completely transparent and giving to the voter both ballot casting assurance and a receipt of the vote which is meaningful only for the direct recipient. The protocol is divided into five phases:

– **Setup.** The authorities generate the values needed by the creation of both the *v-tokens* and the masks associated to the candidates. They also create the *v-tokens* as well as a list containing the eligible voters.
– **Registrar.** A wallet is associated to every eligible voter, so that no one else has access to it. In this wallet there are two indistinguishable *v-tokens*, one real and one fake. Upon registration the voter receives also the information on which token is real and which is fake. This is obviously a very delicate phase, therefore it takes place in a safe and controlled environment (e.g. police station) forcing users to identify themselves. The information on which *v-token* is real and which is fake is given without a receipt so the voter cannot officially prove the validity of a *v-token*.
– **Voting Phase.** Both *v-tokens* must be spent together to have a valid transaction and they have to go to different candidates. After the *v-tokens* have been spent, a receipt is given to the voter. Here we assume that both candidates receive at least one legitimate vote (with a valid *v-token*), otherwise it is trivial to discern valid tokens from the election results.
– **Tallying.** The number of real and fake votes received by each candidate is published.
– **Correctness Check.** The authorities publish a set of values that permits 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-token* has been cast correctly.

Finally each voter can request a series of ZKP to assure that their *v-tokens* have not been manipulated during the voting phase. These proofs are executed in a safe and authenticated environment so that no coercer can exploit them to assess how someone has voted.

3.1 General properties

– **Vote selling and coercion resistance.** The voting protocol presented here provides coercion-resistance and vote selling resistance even if the receipt of the vote is given. In fact when voters casts their ballot, they receive a receipt with written on it to whom the *v-tokens* have been sent to. Since they are indistinguishable, the coercer/vote buyer cannot be sure that the voter voted with the valid *v-token* for the chosen candidate.

- **Correctness.** Every voter can prove that they voted just one time per candidate through their vote receipt, or browsing the blockchain ledger.
- **Prevent double-voting.** The protocol delegates to the underlying blockchain the control against double voting, exploiting consensus rules against double spending of tokens.
- **Privacy.** The privacy of the voter is preserved through pseudonymous wallets and the multiple *v-tokens* that hide the preferences.
- **Fairness.** Fairness of the system is assured by the nature of the blockchain. It allows every voter to see the number of transactions received by both candidates and count the votes received, just after the voting phase ends. This can also be used by voters to check that their *v-tokens* went to the chosen candidates.
- **Transparency.** Since the mathematical background of the whole process is public, everyone can audit it. Moreover the checking phase allows everyone to do a consistency check of the computations performed.

3.2 Protocol Description

The key components involved in the protocol are:

1. A finite set of voters $V = \{v_1, \dots, v_N\}$ with $N \in \mathbb{N}$ the number of eligible voters.
2. Two distinct candidates named *Alpha* and *Beta*.
3. Two different trusted authorities[6] $\mathcal{A}_1$ and $\mathcal{A}_2$
4. One ballot b_i comprising two *v-tokens* for $i \in \{1 \dots N\}$, i.e. one for each eligible voter.

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

Setup

1. $\mathcal{A}_1$ sets up the public ledger, then $\mathcal{A}_1$ chooses a group $\mathbb{G}$ of prime order p, along with a generator $g \in \mathbb{G}$. Then it publishes $\mathbb{G}$, g, p.
2. $\mathcal{A}_1$ chooses uniformly at random, for every voter v_i, a value $x_i \in \mathbb{Z}_p^*$ with $x_i \neq x_j$ for all $i \neq j \in \{1 \dots N\}$. Then $\mathcal{A}_1$ chooses two random values k and λ in $\mathbb{Z}_p^*$. It is important that all the x_is, k and λ remain private. $\mathcal{A}_1$ knows that the *v-tokens* computed using k are valid, while the ones computed using λ are fake, but this information is kept secret.
3. For both candidates $\mathcal{A}_1$ chooses at random a value in $\mathbb{Z}_p^*$: α' and β' respectively, with $\alpha' \neq \beta'$. Those will be the first half of the masks for the votes.

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

4. $\mathcal{A}_1$ chooses a random value $r \in \mathbb{Z}_p^*$ and commits (see Section 2.3) to the values g^r, g^k, g^λ, $g^{\alpha'}$, $g^{\beta'}$, $g^{\alpha' k}$, $g^{\beta' k}$, $g^{\alpha'\lambda}$, $g^{\beta'\lambda}$.
5. For both candidates, $\mathcal{A}_2$ chooses at random a value in $\mathbb{Z}_p^*$: α'' and β'' respectively, with $\alpha'' \neq \beta''$. Those will be the second half of the masks for the votes. Then it commits to the values $g^{\alpha''}$, $g^{\beta''}$.
6. $\mathcal{A}_1$ chooses uniformly at random for every voter v_i a value $y_i' \in \mathbb{Z}_p^*$, with $y_i' \neq y_j'$ for all $i \neq j \in \{1 \dots N\}$. Then for every $i \in \{1 \dots N\}$ it commits to the pairs (v_i, g^{rx_i}), $(v_i, g^{y_i'})$, $(v_i, g^{x_i y_i'})$.
7. For every voter v_i, $\mathcal{A}_1$ chooses uniformly at random $\pi_i \in \{1, 2\}$ and creates the preliminary ballot

$$\bar{b}_i = (\bar{b}_{i,1}, \bar{b}_{i,2}) = \left(g^{y_i'(x_i+\sigma_{i,1})}, g^{y_i'(x_i+\sigma_{i,2})}\right) \tag{4}$$

where:

$$\sigma_{i,l} := \begin{cases} k \iff \pi_i = l & \text{i.e. } \bar{b}_{i,l} \text{ is real} \\ \lambda \text{ otherwise,} & \text{i.e. } \bar{b}_{i,l} \text{ is fake} \end{cases} \tag{5}$$

In this notation, i represents the voter while $l = 1, 2$ is the *v-token* position in the pair.
8. $\mathcal{A}_1$ creates a list of all the eligible voters and associates to everyone a wallet in which the *v-tokens* will be stored. Eventually it sends to $\mathcal{A}_2$: the list of the ballots $\bar{b}_i$, the list of wallet addresses, and their association.
9. Once $\mathcal{A}_2$ has the address-ballots list, it associates to every voter a random value $y_i'' \in \mathbb{Z}_p^*$, with $y_i'' \neq y_j''$ for all $i \neq j \in \{1 \dots N\}$. In this way $\mathcal{A}_2$ creates the final ballot for every voter v_i:

$$b_i = \left(\bar{b}_{i,1}^{y_i''}, \bar{b}_{i,2}^{y_i''}\right) = \left(g^{y_i(x_i+\sigma_{i,1})}, g^{y_i(x_i+\sigma_{i,2})}\right), \quad \text{with} \quad y_i := y_i' \cdot y_i''. \tag{6}$$

Finally for every $i \in \{1 \dots N\}$ $\mathcal{A}_2$ commits to the pair $(v_i, g^{y_i''})$.

Voting Phase

1. $\mathcal{A}_2$ sends to every voter its associated ballot, a pair b_i.
2. For every $1 \leq i \leq N$ the voting phase proceeds in the same way, as we elaborate below. Since the *v-tokens* can be reordered, the voter orders them so that the first *v-token* is meant for *Alpha* and the second for *Beta*. It sends its reordered pair b_i off-chain to $\mathcal{A}_1$[7]. The first authority records that v_i has voted, checking also that the same voter has not voted yet. $\mathcal{A}_1$ then computes the preliminary vote:

$$\bar{b}_{i_\alpha} = b_{i,1}^{\frac{\alpha'}{y_i'}} = \left(g^{y_i(x_i+\sigma_{i,1})}\right)^{\frac{\alpha'}{y_i'}} = \left(g^{\alpha' \cdot y_i''(x_i+\sigma_{i,1})}\right), \tag{7}$$

$$\bar{b}_{i_\beta} = b_{i,2}^{\frac{\beta'}{y_i'}} = \left(g^{y_i(x_i+\sigma_{i,2})}\right)^{\frac{\beta'}{y_i'}} = \left(g^{\beta' \cdot y_i''(x_i+\sigma_{i,2})}\right), \tag{8}$$

[7] The authority $\mathcal{A}_1$ gains no information from this since the two *v-tokens* are indistinguishable thanks to the mask y_i''.

and sends off-chain the pair $\bar{b}_{i_{\alpha\beta}} = (\bar{b}_{i_\alpha}, \bar{b}_{i_\beta})$ to $\mathcal{A}_2$[8]. $\mathcal{A}_2$ then computes the final vote (let $\alpha := \alpha' \cdot \alpha''$ and $\beta := \beta' \cdot \beta''$):

$$b_{i_\alpha} = \bar{b}_{i_\alpha}^{\frac{\alpha''}{y_i''}} = \left(g^{\alpha' \cdot y_i''(x_i+\sigma_{i,1})}\right)^{\frac{\alpha''}{y_i''}} = \left(g^{\alpha(x_i+\sigma_{i,1})}\right), \tag{9}$$

$$b_{i_\beta} = \bar{b}_{i_\beta}^{\frac{\beta''}{y_i''}} = \left(g^{\beta' \cdot y_i''(x_i+\sigma_{i,2})}\right)^{\frac{\beta''}{y_i''}} = \left(g^{\beta(x_i+\sigma_{i,2})}\right), \tag{10}$$

and sends off-chain the pair $b_{i_{\alpha\beta}} = (b_{i_\alpha}, b_{i_\beta})$ back to the voter. Also $\mathcal{A}_2$ records that v_i has voted and assures that no one tries to vote more than once.

3. The two masked *v-tokens* are sent with a transaction on the blockchain to the respective candidates. The voter receives the receipt of the vote which basically is the insertion of the transaction in the blockchain.

Tallying Once the voting phase is over, the tallying can start. This phase is symmetrical for *Alpha* and *Beta*, and it is sufficient to count the votes only for one candidate, so we will describe it just for *Alpha*.

Suppose that $T \leq N$ participants voted. Without loss of generality, we can assume that only the participants with index $1 \leq i \leq T$ voted, while the remaining $N - T$ abstained from voting[9].

1. $\mathcal{A}_1$ publishes the decommitments for $g^{\alpha'}$, $g^{\beta'}$, $g^{\alpha' k}$, $g^{\beta' k}$, $g^{\alpha' \lambda}$, $g^{\beta' \lambda}$. $\mathcal{A}_2$ publishes $g^{\alpha k}$, $g^{\beta k}$, $g^{\alpha \lambda}$, $g^{\beta \lambda}$, g^{α}, g^{β} (which $\mathcal{A}_2$ can compute after seeing those decommitments, e.g. $g^{\alpha k} = (g^{\alpha' k})^{\alpha''}$), and the decommitments for $g^{\alpha''}$, $g^{\beta''}$ (now $\mathcal{A}_1$ can check these computations).
2. Knowing which participants voted, $\mathcal{A}_1$ computes $\texttt{sum} = \sum_{s=1}^{T} x_s$, then it publishes $g^{\alpha \cdot \texttt{sum}}$, $g^{\beta \cdot \texttt{sum}}$, $g^{r \cdot \texttt{sum}}$. For public verifiability $\mathcal{A}_1$ publishes the decommitment of $(v_i, g^{r x_i})$ for $i \in \{1, \ldots, T\}$, and can prove in ZK that `sum` is the same in every exponent (see Section 3.2).
3. $\mathcal{A}_1$ and $\mathcal{A}_2$ decommit the other values (previously committed), excluding the pairs $(v_i, g^{y_i'})$, $(v_i, g^{x_i y_i'})$, $(v_i, g^{y_i''})$ $\forall i$.
4. Multiplying all *v-tokens* in *Alpha*'s wallet and dividing by $g^{\alpha \cdot \texttt{sum}}$ anyone can compute:

$$(g^{\alpha \cdot \texttt{sum}})^{-1} \prod_{i=1}^{T} g^{\alpha(x_i+\sigma_{i,l})} = \left(g^{\alpha k}\right)^{\texttt{valid}_\alpha} \left(g^{\alpha \lambda}\right)^{\texttt{fake}_\alpha} \tag{11}$$

where $l = 1$ or 2 depending on the *v-token* used and

(a) $\texttt{valid}_\alpha$ is the number of valid votes received by *Alpha*,
(b) $\texttt{fake}_\alpha$ is the number of fake votes received by *Alpha*.

[8] The authority $\mathcal{A}_2$ also gains no information from this since the two *v-tokens* are indistinguishable.

[9] Note that the voting addresses can be seen by everyone in the blockchain.

Note that $\texttt{fake}_\alpha$ and $\texttt{valid}_\alpha$ (that correspond respectively to $\texttt{fake}_\beta$ and $\texttt{valid}_\beta$) can be easily computed by brute force, in fact given an 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.

Comparing the number of valid votes for each candidate, the winner of the election is found.

ZKP for integrity checks Firstly, a ZKP is needed to assure that votes have been masked correctly. In other words, that the authorities computed

$$g^{y_i(x_i+k)} \to g^{\alpha(x_i+k)} \tag{12}$$

without messing with the exponents. Note that the same argument holds for λ and β instead of k and α respectively, in any combination. The voter v_i knows g^α and the value of the *v-tokens*, before and after the voting mask has been applied. In a safe and authenticated environment $\mathcal{A}_1$ and $\mathcal{A}_2$ decommit to v_i the values of $(v_i, g^{y_i'})$ and $(v_i, g^{y_i''})$ respectively. Then $\mathcal{A}_1$ computes g^{y_i} and proves that the result is correct with the Schnorr ZKP presented in Section 2.2 and using:

$$\omega = y_i', \qquad u = g, \qquad z = g^{y_i'}, \qquad \bar{u} = g^{y_i''}, \qquad \bar{z} = g^{y_i}. \tag{13}$$

Then $\mathcal{A}_1$ can prove the correctness of the mask setting:

$$\omega = (x_i + k), \quad u = g^{y_i}, \quad z = g^{y_i(x_i+k)}, \quad \bar{u} = g^\alpha, \quad \bar{z} = g^{\alpha(x_i+k)}. \tag{14}$$

Adapting the proof of $g^{y_i'}, g^{y_i''} \to g^{y_i}$, $\mathcal{A}_1$ can also prove the correct computation of $g^{\alpha'}, g^{\alpha''} \to g^\alpha$, $g^{\alpha' k}, g^{\alpha''} \to g^{\alpha k}$ and similarly for β and λ.
Any voter[10] could ask also for a proof of the exponentiation $g^{k\alpha}$ (and similarly for λ and β). The voter knows g^α, g^k, g, so $\mathcal{A}_1$ can prove it by setting:

$$\omega = k, \qquad u = g, \qquad z = g^k, \qquad \bar{u} = g^\alpha, \qquad \bar{z} = g^{k\cdot\alpha}. \tag{15}$$

Any voter could ask for a proof that $\texttt{sum} = \sum_{i=1}^{T} x_i$. The voter knows g^α, $g^{\alpha\cdot\texttt{sum}}$, g^r, and $g^{r\sum_{i=1}^{T} x_i}$ (computed observing the votes on chain and using the pairs (v_i, g^{rx_i}) previously decommitted), so $\mathcal{A}_1$ can prove it by setting:

$$\omega = \texttt{sum}, \qquad u = g^r, \qquad z = g^{r\sum_{i=1}^{T} x_i}, \qquad \bar{u} = g^\alpha, \qquad \bar{z} = g^{\alpha\cdot\texttt{sum}}. \tag{16}$$

3.3 On the honesty of $\mathcal{A}_1$

Finally, a voter v_i can ask (always in a safe and authenticated environment) for a proof that in the registration phase the authority $\mathcal{A}_1$ correctly computed and

[10] Actually anyone observing the protocol execution.

identified the *v-tokens*, i.e. that the *v-token* identified as `valid` by $\mathcal{A}_1$ was the one containing k.

Recall that a *v-token* is:

$$b_{i,l} = g^{y_i(x_i+\sigma_{i,l})} = g^{y_i \cdot x_i} \cdot g^{y_i \cdot \sigma_{i,l}}, \quad \text{with} \quad \sigma_{i,l} \in \{k, \lambda\}. \tag{17}$$

$\mathcal{A}_1$ starts by decommitting $(v_i, g^{y_i' x_i})$ to the voter, then computes $g^{y_i x_i}$ from $g^{y_i''}$ setting:

$$\omega = y_i' x_i, \qquad u = g, \qquad z = g^{y_i' x_i}, \qquad \bar{u} = g^{y_i''}, \qquad \bar{z} = g^{y_i \cdot x_i}. \tag{18}$$

Then the voter knows g, g^{y_i}, g^k, g^λ, $g^{r \cdot x_i}$ and g^r. Now $\mathcal{A}_1$ can prove the validity of the factor $g^{y_i \cdot x_i}$ by setting:

$$\omega = x_i, \qquad u = g^r, \qquad z = g^{r \cdot x_i}, \qquad \bar{u} = g^{y_i}, \qquad \bar{z} = g^{y_i \cdot x_i}. \tag{19}$$

To conclude its proof, $\mathcal{A}_1$ can prove than the `valid` coin contains k while the `fake` contains λ by setting:

$$\omega = k, \qquad u = g, \qquad z = g^k, \qquad \bar{u} = g^{y_i}, \qquad \bar{z} = g^{y_i \cdot k}, \tag{20}$$

and:

$$\omega = \lambda, \qquad u = g, \qquad z = g^\lambda, \qquad \bar{u} = g^{y_i}, \qquad \bar{z} = g^{y_i \cdot \lambda}, \tag{21}$$

where the values of $\bar{z}$ can be deduced by the voter dividing the *v-tokens* by $g^{y_i \cdot x_i}$, which has been proved correct in the previous step.

4 Proof of security

For the sake of clarity and to simplify the notation in this section we omit the random ordering of the *v-tokens*, so we write either k or λ instead of $\sigma_{i,l}$, without loss of generality.

The goal is to prove that an adversary cannot distinguish between real and fake *v-tokens* and guess how voters cast their preference. Since election results are obviously public we have to avoid some trivial cases in which the adversary can deduce the votes simply observing the results. Therefore we assume that the adversary controls all but two voters an that these two voters select distinct candidates. The adversary wins the security game if it guesses correctly for which candidate each of the two voted.

4.1 Security Model

The security of the protocol will be proven in terms of vote indistinguishability (VI), as detailed in Definition 3.

The security of the protocol will be proved in terms of two authorities, $\mathcal{A}_1$ that is honest and $\mathcal{A}_2$ that is honest but *leaky*, i.e. the second authority leaks information to the adversary, specifically the values of y_i'' for all i, α'' and β''.

Definition 2 (Security Game). *The security game for a two-candidates protocol proceeds as follows:*

– ***Init.*** *The adversary $\mathcal{E}$ chooses $N-2$ users that it will control and therefore the adversary knows which are the real and fake* v-tokens *of these users. The remaining two are called* free voters.
– ***Setup.*** *The Challenger takes the role of both authorities and constructs all v-tokens.*
– ***Phase 0.*** *The adversary may request to see the v-tokens of any voter, including the free voters.*
– ***Phase 1.*** *The adversary may request to vote with some or all of the voters it controls.*
– ***Challenge.*** *The Challenger votes with the v-tokens of the free voters flipping a random coin and gives to the adversary the receipt of the vote.*
– ***Phase 2.*** *The adversary can vote with some or all of the voters it controls which did not vote in Phase 1.*
– ***Phase 3.*** *The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results on behalf of the voters it controls.*
– ***Guess.*** *The adversary chooses one free voter and outputs a guess on which candidate it voted for* [11].

Definition 3 (Vote Indistiguishability). *A Two-Candidates Protocol with security parameter ξ is VI-secure if for all probabilistic polynomial-time adversaries $\mathcal{E}$ there exists a negligible function ϕ such that:*

$$\mathbb{P}[\mathcal{E}\ wins] \leq \frac{1}{2} + \phi(\xi). \tag{22}$$

In the following theorem we prove our voting protocol VI-secure under the DDH assumption (Definition 1) in the security game defined above and under the assumption that the commitment scheme is perfectly hiding and computationally binding.

Theorem 1. *Suppose that the commitment scheme is perfectly hiding and computationally binding. If an adaptive distinguisher adversary can break the scheme, then a simulator can be constructed to play the DDH game with non-negligible advantage.*

Proof. Suppose there exists a polynomial time adversary $\mathcal{E}$, 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 simulation proceeds as follows.

– **Init** The adversary chooses the $N-2$ users to control.

[11] Therefore the adversary implicitly guesses also the vote of the other free voter. In other words, for each of the four *v-tokens* outside its control, the adversary guesses if the *v-token* is `valid` or not.

The simulator takes in a DDH challenge:

$$(g, A = g^a, B = g^b, T), \tag{23}$$

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

– **Setup** Without loss of generality we may assume that the two uncontrolled voters are v_1 and v_2. The simulator chooses uniformly at random the values $d, e, \bar{y}_1, \bar{y}_2, \bar{k}, \bar{\lambda}, \alpha', \alpha'', \beta', \beta'', r \in \mathbb{Z}_p^*$. Then it chooses uniformly at random y'_i, x_i for $3 \leq i \leq N$, y''_j for $1 \leq j \leq N$. Finally $\mathcal{S}$ implicitly sets:

$$y'_1 = \frac{\bar{y}_1}{a}, \qquad y'_2 = \frac{\bar{y}_2}{a}, \qquad k = \bar{k} \cdot a, \qquad \lambda = \bar{\lambda} \cdot a. \tag{24}$$

Since authority $\mathcal{A}_2$ is leaky, its parameters are given to the adversary.
The *v-tokens* of the *free* voters are constructed in this way:

$$b_1 = \left(g^{y''_1\bar{y}_1(d+\bar{k}-\bar{\lambda})} \cdot B^{y''_1\bar{y}_1}, B^{y''_1\bar{y}_1} \cdot g^{y''_1\bar{y}_1 d}\right), \tag{25}$$

$$b_2 = \left(B^{-y''_2\bar{y}_2} \cdot g^{y''_2\bar{y}_2 e}, g^{y''_2\bar{y}_2(e-\bar{k}+\bar{\lambda})} \cdot B^{-y''_2\bar{y}_2}\right), \tag{26}$$

implicitly setting:

$$x_1 + k = (d + \bar{k} - \bar{\lambda})a + ab, \tag{27}$$

$$x_1 + \lambda = ab + da, \tag{28}$$

$$x_2 + k = -ab + ea, \tag{29}$$

$$x_2 + \lambda = (e - \bar{k} + \bar{\lambda})a - ab. \tag{30}$$

so that the DDH challenge appears only in the votes.
For every other voter v_i with $3 \leq i \leq N$ the ballots are computed as:

$$b_i = \left(g^{y_i(x_i+\bar{k}a)}, g^{y_i(x_i+\bar{\lambda}a)}\right) = \left(g^{y_i x_i} \cdot A^{\bar{k}}, g^{y_i x_i} \cdot A^{\bar{\lambda}}\right), \tag{31}$$

where $y_i := y'_i \cdot y''_i$.
The simulator can compute every value it has to commit and then decommit to the adversary as:

$$g^r, g^{\alpha'}, g^{\beta'}, g^{\alpha''}, g^{\beta''}, g^{\alpha}, g^{\beta}, \tag{32}$$

$$g^{rx_i}, g^{y'_i}, g^{y''_i}, g^{x_i y'_i}, g^{y_i}, g^{x_i y_i}, g^{y_i k} = A^{y_i\bar{k}}, g^{y_i\lambda} = A^{y_i\bar{\lambda}} \;\; 3 \leq i \leq N, \tag{33}$$

$$g^{rx_1} = T^r A^{r(d-\bar{\lambda})}, \quad g^{rx_2} = T^{-r} A^{r(e-\bar{k})}, \tag{34}$$

$$g^{\alpha' k} = A^{\alpha'\bar{k}}, \quad g^{\beta' k} = A^{\beta'\bar{k}}, \quad g^{\alpha'\lambda} = A^{\alpha'\bar{\lambda}}, \quad g^{\beta'\lambda} = A^{\beta\bar{'}\lambda}, \tag{35}$$

$$g^{\alpha k} = A^{\alpha\bar{k}}, \quad g^{\beta k} = A^{\beta\bar{k}}, \quad g^{\alpha\lambda} = A^{\alpha\bar{\lambda}}, \quad g^{\beta\lambda} = A^{\beta\bar{\lambda}}, \tag{36}$$

$$g^{k} = A^{\bar{k}}, \quad g^{\lambda} = A^{\bar{\lambda}}, \tag{37}$$

$$g^{\alpha\left(\sum_{s=1}^{N} x_s\right)} = A^{(d+e-\bar{\lambda}-\bar{k})\alpha} \cdot g^{\alpha\left(\sum_{s=3}^{N} x_s\right)}, \tag{38}$$

$$g^{\beta\left(\sum_{s=1}^{N} x_s\right)} = A^{(d+e-\bar{\lambda}-\bar{k})\beta} \cdot g^{\beta\left(\sum_{s=3}^{N} x_s\right)}, \tag{39}$$

$$g^{r\left(\sum_{s=1}^{N} x_s\right)} = A^{(d+e-\bar{\lambda}-\bar{k})r} \cdot g^{r\left(\sum_{s=3}^{N} x_s\right)}, \tag{40}$$

and therefore it can actually compute these commitments. $\mathcal{S}$ however does not know the values that should be decommitted privately to v_1 or v_2, but it can commit random values instead and still simulate the protocol correctly (since the commitment scheme is perfectly hiding).

- **Phase 0** The values of the ballots are given to the adversary.
- **Phase 1** The adversary chooses some voters i, $3 \leq i \leq N$ and decides how they vote. Without loss of generality let us suppose that v_i votes for Alpha, then the voting tokens are computed as follows:
$$V_i = \left(g^{\alpha(x_i+k)}, g^{\beta(x_i+\lambda)}\right) = \left(g^{\alpha x_i} \cdot A^{\bar{k}\alpha}, g^{\beta x_i} \cdot A^{\bar{\lambda}\beta}\right). \tag{41}$$
- **Challenge** The *Challenger* flips a coin and decides which voter votes for *Alpha* with the real *v-token* and which for *Beta*. For simplicity, suppose that v_1 votes with the real *v-token* for *Alpha* and with the fake one for *Beta* while v_2 does exactly the opposite. The votes are constructed as:
$$V_1 = \left(T^{\alpha} \cdot A^{(d+\bar{k}-\bar{\lambda})\alpha}, T^{\beta} \cdot A^{\beta \cdot d}\right), \tag{42}$$
$$V_2 = \left(T^{-\beta} \cdot A^{\beta \cdot e}, T^{-\alpha} \cdot A^{\alpha(e-\bar{k}+\bar{\lambda})}\right). \tag{43}$$
Note that:
$$V_{1,\alpha} \cdot V_{2,\alpha} \cdot \prod_{i=3}^{N} V_{i,\alpha} = A^{(d+e)\alpha} \cdot \prod_{i=3}^{N} V_{i,\alpha}, \tag{44}$$
$$V_{1,\beta} \cdot V_{2,\beta} \cdot \prod_{i=3}^{N} V_{i,\beta} = A^{(d+e)\beta} \cdot \prod_{i=3}^{N} V_{i,\beta}, \tag{45}$$
so the product of all the votes received by both of the candidates does not contain the value of the challenge T.
- **Phase 2** During this phase the simulator acts exactly as in *Phase* 1.
- **Phase 3** The voting phase ends and the previously committed values are decommitted (excluding the values reserved for v_1 and v_2). In addition the adversary could request the Zero-Knowledge Proofs of the correct computations of the votes. Since the simulator has many implicit parameters (i.e. in many cases does not know the actual value of ω to use in the proof), then some simulations of the Schnorr protocol are requested, as presented in Section 2.2. Further discussion on Zero-Knowledge proofs and simulations can be found in [11].
- **Guess** After watching both the initial ballots b_i and the computed votes V_i, the adversary will eventually output a guess of the real and fake *v-tokens* of v_1 and v_2. The simulator then outputs 0 to guess that $T = g^{ab}$ if the guess of $\mathcal{E}$ was correct, otherwise it outputs 1 to indicate that T is a random group element in $\mathbb{G}$. In fact when T is not random the simulator $\mathcal{S}$ gives a perfect simulation:
$$V_1 = \left(T^{\alpha} \cdot A^{(d+\bar{k}-\bar{\lambda})\alpha}, T^{\beta} \cdot A^{\beta \cdot d}\right) = \left(g^{\alpha(x_1+k)}, g^{\beta(x_1+\lambda)}\right), \tag{46}$$
$$V_2 = \left(T^{-\beta} \cdot A^{\beta \cdot e}, T^{-\alpha} \cdot A^{\alpha(e-\bar{k}+\bar{\lambda})}\right) = \left(g^{\beta(x_2+k)}, g^{\alpha(x_2+\lambda)}\right). \tag{47}$$

This means that the advantage is preserved and so it holds that:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{48}$$

On the contrary when T is a random element $R \in \mathbb{G}$ the votes are completely random values from the adversary point of view, so:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = R) = 0] = \frac{1}{2}. \tag{49}$$

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

□

5 Conclusions and future works

In the protocol proposed here, an underlying blockchain infrastructure and a system of ZKP ensure transparency and auditability of the whole process, achieving extensive security, including coercion and vote-selling resistance, while retaining receipts. Indeed our approach disguises real and fake votes, which remain indistinguishable even after tallying.

This work considers two authorities for the sake of exposition clarity. In a real case scenario, the work of these two authorities can be divided into more independent authorities, each two of them managing a restricted pool of voters (like a voting district). These authorities do not need to share anything: all of them can compute the result of the elections in their own way and then share just the number of valid votes received in that district. This approach limits the knowledge that an authority can gain. In fact, after the election, $\mathcal{A}_1$ can compute how every voter cast their ballot since it knows $g^{\alpha x_i}$ for every i. However the second authority has been introduced so that nothing can be discovered about the votes until the election is over, therefore preventing possible interferences and tampering from the first authority.

A generalization of this protocol to multiple candidates is under active investigation.

References

1. 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)
2. Babai, L., Moran, S.: Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Class. Journal of Computer and System Sciences **36**(2), 254–276 (1988). https://doi.org/10.1016/0022-0000(88)90028-1, https://doi.org/10.1016/0022-0000(88)90028-1

3. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 103–112. STOC, ACM (1988)
4. Bohli, J.M., Müller-Quade, J., Röhrich, S.: Bingo voting: Secure and coercion-free voting using a trusted random number generator. In: Proceedings of International Conference on E-Voting and Identity. pp. 111–124. Springer (2007)
5. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of computer and system sciences **37**(2), 156–189 (1988)
6. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy. pp. 354–368. SP, IEEE Computer Society (2008). https://doi.org/10.1109/SP.2008.32, https://doi.org/10.1109/SP.2008.32
7. Costa, D., Fiori, F., Milan, P., Sala, M., Vitale, A., Vitale, M.: Quadrans whitepaper (2019), https://quadrans.io/content/files/quadrans-white-paper-rev01.pdf
8. Fouard, L., Duclos, M., Lafourcade, P.: Survey on electronic voting schemes (2007)
9. Grewal, G., Ryan, M., Bursuc, S., Ryan, P.: Caveat Coercitor: Coercion-Evidence in Electronic Voting. In: Proceedings - IEEE Symposium on Security and Privacy. pp. 367–381. IEEE (2013). https://doi.org/10.1109/SP.2013.32
10. Li, H., Kankanala, A., Zou, X.: A taxonomy and comparison of remote voting schemes. In: Proceedings - International Conference on Computer Communications and Networks, ICCCN. pp. 1–8. IEEE (08 2014). https://doi.org/10.1109/ICCCN.2014.6911807
11. 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
12. Longo, R.: Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols. Ph.D. thesis, University Of Trento, Department of Mathematics (2018)
13. Meneghetti, A., Sala, M., Taufer, D.: A Survey on PoW-based Consensus. Annals of Emerging Technologies in Computing (AETiC) **4**(1) (2020)
14. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4**, 161–174 (1991). https://doi.org/10.1007/BF00196725
15. Victor Shoup, J.A.: Σ-Protocols Continued and Introduction to Zero Knowledge (2007), https://cs.nyu.edu/courses/spring07/G22.3220-001/lec3.pdf
16. 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)