Advances in Mathematics of Communications doi:10.3934/amc.2021005

A COERCION-RESISTANT BLOCKCHAIN-BASED E-VOTING PROTOCOL WITH RECEIPTS

Chiara Spadafora, Riccardo Longo* and Massimiliano Sala

Department of Mathematics, University Of Trento
38123 Povo, Trento, Italy

(Communicated by Daniele Bartoli)

Abstract. 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 achieve decentralization using blockchain technology. 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. We prove the security of the protocol under the standard DDH assumption on the underlying prime-order cyclic group (e.g. the group of points of an elliptic curve), as well as under standard assumptions on blockchain robustness.

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 [11], 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 [32]. Usually, e-voting protocols are requested to 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.

2020 *Mathematics Subject Classification:* Primary: 94A60; Secondary: 94B99, 68W40.
Key words and phrases: E-voting, coercion-resistance, cryptography, Diffie-Hellman assumption, blockchain.

* Corresponding author: riccardolongomath@gmail.com.

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, with a proof similar to that in [25] but with subtle modifications to conform to our protocol's requirements, 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. We provide a proof of security in Section 4 where we show vote indistinguishability with a simulation-based proof and show that our protocols satisfies the desirable properties of a voting system. In Section 5, we briefly discuss the generalization of our protocol to multiple candidates. Finally, in Section 6 we draw some conclusions.

1.1. RELATED WORK. The research in the field of e-voting is constantly growing, with a lot of protocols proposed. *Civitas* [9] deals with coercion allowing voters to generate, with their designated private key, fake credentials and then erasing all the votes submitted through them. *Helios* [1] was designed mainly for the purpose of low-coercion applications. It highlights the verifiability in remote e-voting systems. In Helios, voters do not need to be authenticated until they cast votes. Under such a situation, anyone can participate and test the system. *Caveat Coercitor* [15] is a unique voting system that, instead of preventing coercion, allows it, while recording unforgeable evidence of said 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* [6] 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.

Concerning DLT voting systems, *Crypto-voting* [12] is a blockchain based e-voting system developed by the University of Cagliari in collaboration with Net Service SPA. It exploits the use of sidechains and guarantees vote privacy thanks to an approach based on Shamir's secret sharing scheme. *Agora* [13] runs on a custom blockchain with three main components: Skipchain, Cotena and Valeda. The first manages consensus, Cotena anchors the system to the Bitcoin blockchain, periodically storing a hash of the most recent Skipblock in a Bitcoin transaction. Valena validates both Skipchain and Cotena data by the use of cryptographic proofs. *Vocdoni* [30] is handled by a Tendermint blockchain called *vochain*. Data integrity is provided by the Ethereum blockchain. *Voatz* [28] is used by several counties and states in the USA. It is based on an application which is able to perform biometric identifcation of the voter. The system runs on the Voatz blockchain which was built using the Hyperledger framework. On August 2020, the *US Patent and Trademark Office* [14] published a patent application that outlines various blockchain-enabled approaches for potential vote-by-mail systems. In one suggested implementation, the voter receives a paper ballot on which there is printed an individualized, computer-readable code. The voter could then scan the code with a mobile device, verify his identification, and then cast the ballot digitally. This system separates the voter ID from the actual vote to maintain anonymity, with the votes stored on the blockchain. Other interesting systems are: Netvote [17], OV-net [21], Follow My Vote [29] , Polys [2] and Colony [23]. A more comprehensive comparison of voting protocols can be found in [18], [31].

A preliminary (weaker) version of the protocol presented here can be found in [27].

2. Preliminaries

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

Definition 1 (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 [20].

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

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

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

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

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

Definition 2 (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. Zero-Knowledge proofs were first conceived in 1989 by Shafi Goldwasser, Silvio Micali, and Charles Rackoff [5]. Together with the work of Laszlo Babai and Shlomo Moran [4], 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 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 3 (Completeness)**.** If $x \in \mathcal{L}$ (i.e. the prover is honest) then the verifier should accept the proof with probability 1.

Definition 4 (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 5 (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 $x \in \mathcal{L}$.

2.2.1. *Equality of discrete logarithms.* The simplest of ZKPs is due to Schnorr [24]: the proof of knowledge of a *discrete logarithm*. Here we present a variation of the Schnorr interactive protocol, similar to that in [25], 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:*

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

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 $\tau = \text{poly}(\log_2(p))$ times, i.e. the number of repetitions is polynomial in the length of p (the security parameter).*

Proposition 1 (Completeness of Protocol 1)**.** *Under the DDH assumption, the Protocol 1 satisfies the completeness property, as per Definition 3.*

Proof. To show that this protocol is correct, it suffices to verify that the equations of steps 3 and 4 hold when s is computed correctly. □

Proposition 2 (Soundness of Protocol 1)**.** *Under the DDH assumption, the Protocol 1 satisfies the soundness property, as per Definition 4.*

Proof. To show soundness, first note that the prover can guess all τ values of the challenges c only with probability $2^{-\tau}$ 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. □

Proposition 3 (Zero-Knowledge of Protocol 1)**.** *Under the DDH assumption, the Protocol 1 satisfies the Zero-Knowledge property, as per Definition 5.*

Proof. This property states 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 $i = \tau$ 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 2τ 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. □

From this proof the following corollary is immediately derived:

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.*

Further discussion on Zero-Knowledge proofs and simulations can be found in [19]. In the proof of Theorem 15 we will need also the following lemma:

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.*

Proof. It is possible to retrieve from $\mathcal{A}$ the secret exponent ω as follows:

1. $\mathcal{A}$ sends $(t, \bar{t})$.
2. We respond sending $c = 0$.
3. $\mathcal{A}$ then responds with $s = r + c \cdot \omega = r$.
4. If $u^s = t$, $\bar{u}^s = \bar{t}$, we rewind to Item 2, otherwise we abort.
5. This time we send $c = 1$, so $\mathcal{A}$ responds with $s' = r + \omega$, and since r is the same we can compute $\omega = s' - s = r + \omega - r$. Again if $u^\omega \neq z$ or $\bar{u}^\omega \neq \bar{z}$ we abort.

Clearly, if the protocol does not abort we recovered ω. □

2.3. Commitment scheme. A commitment scheme [7] 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 prevent some possible malicious choice of parameters, specifically we want that the authorities choose their values independently. However, this kind of suspicious behavior does not affect Vote-Indistinguishability (see Definition 14) thanks to the hardness of DLOG problem, so in our analysis commitments are not directly involved in the proof of Theorem 15. 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 containing a list of *transactions* 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 reorder, delete or modify past transactions, but it can add new transactions.

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 by reaching a consensus (e.g. [22]) on the transaction to append and by publishing the updated state.

In our protocol the main users are the voters, the miners are roughly equivalent to election officers, while 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 representing 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[1] (so the internal PKI will guarantee the identification);
- to use a *permissionless* blockchain[2] with either a smart contract and/or an external oracle.

Remark 1. For any e-voting protocol the problem of voter registration and authentication is very hard to solve, as shown in [8, 16], however it is out of scope of this paper.

Finally, we need that miners enforce some consistency rules:

- transactions are properly authorized by the user who owns the token, i.e. a user's tokens cannot be spent by anybody else;

[1]We are implementing a prototype with HyperLedger Fabric [3], the source code will be published on github.

[2]We are also implementing a prototype with Quadrans [10], the source code will be published on github.

- 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);
- no *double spending* of tokens (each token can be spent only once).

This abstract model can be implemented using one of the many blockchain constructions already available.

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 this section we define such properties, we will prove that our proposed protocol satisfies them in Section 4.2.

Definition 6 (Correctness)**.** A voting system is *correct* if an adversary cannot alter or cancel the votes of the honest voters (i.e. the one that are not coerced), and cannot cause voters to cast ballots resulting in double voting (i.e. use one credential to vote multiple times).

Definition 7 (Fairness)**.** A voting system is *fair* if no information about how many votes each candidate has received can be learned until the voting results are published. Any participant cannot gain knowledge of the voting result before its final publication.

Definition 8 (Transparency)**.** A voting system is *transparent* if the voters are able to understand the voting system in all its components.

Definition 9 (Privacy)**.** A voting system is *private* if no entity involved in the voting process can link a cast ballot to the voter who cast it.

Definition 10 (Verifiability)**.** A voting system is *verifiable* if it satisfies the following two properties:

1. **Universal Verifiability**: the correctness of elections results can be verified by all observers;
2. **Individual Verifiability**: every voter can check that their vote has been cast correctly and has been accurately counted in tallied results.

A secure voting scheme should posses some counter attacks requirements to enhance the security of the model.

Definition 11 (Vote-coercion resistance)**.** Voters should be able to cast their ballots as they want, even if someone tries to coerce them.

In some cases, voters are willing to sell their vote to obtain various material or immaterial goods. However, in order to actually prove a vote, the voter has to provide an evidence of the vote (like a picture of the ballot or a receipt of the vote) or sell directly their credentials. The latter situation is applicable only 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 12 (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 11, 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. Two-candidate voting system

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 valid, the other is fake, but only the voter knows which is which. When voting, every voter expresses its preference assigning the valid *v-token* to the chosen candidate and the fake one to the other. The voter gets a vote receipt on which both transactions will be displayed. In the final tally the fake *v-tokens* are discarded[3] 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:

- **Setup.** The authorities, knowing a list of eligible voters, generate the values for the creation of the *v-tokens* and the masks associated to the candidates.
- **Registrar.** The voter creates a wallet (that no one else has control over) and registers it with the two authorities, which then proceed to create two indistinguishable *v-tokens*, one valid and one 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 one authority does not see what the other is 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.** Both *v-tokens* 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 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 *v-tokens* are processed (see 3.1.4) 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 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, anyone can request a series of ZKPs to assure that the *v-tokens* have been correctly processed during the tallying phase.

[3]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 received by the candidate.

3.1. PROTOCOL DESCRIPTION. The key components involved in the protocol are:

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

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

3.1.1. *Setup.* The first authority, $\mathcal{A}_1$, selects a secure group $\mathbb{G}$ of prime order p in which the DDH assumption holds, along with a generator $g \in \mathbb{G}$, then it publishes $\mathbb{G}$, g, p.

Then, $\mathcal{A}_1$ performs the following operations:

1. Chooses at random a value $\alpha'_l \in \mathbb{Z}_p^*$ for $l \in \{1, 2\}$ (i.e. one for each candidate), with $\alpha'_1 \neq \alpha'_2$. Those will be the first half of the masks for the votes.
2. Chooses uniformly at random two values $x'_i, y'_i \in \mathbb{Z}_p^*$ for every voter v_i, with $x'_i \neq x'_j$, $y'_i \neq y'_j$ for all $i \neq j \in \{1 \ldots N\}$.
3. Chooses two random values k and λ in $\mathbb{Z}_p^*$. $\mathcal{A}_1$ knows that the *v-tokens* computed using k are valid, while the ones computed using λ are fake, but this information is kept secret.
4. Finally, $\mathcal{A}_1$ commits (see Section 2.3) to the values g^k, g^λ, $g^{\alpha'_1}$, $g^{\alpha'_2}$, and for every $i \in \{1 \ldots N\}$ it commits to the pairs $(v_i, g^{x'_i})$, $(v_i, g^{y'_i})$.

It is important that all the values x'_i, y'_i, α'_l k, λ remain private.

Similarly, the second authority, $\mathcal{A}_2$, performs the following operations:

1. Chooses at random a value $\alpha''_l \in \mathbb{Z}_p^*$ for each candidate, with $\alpha''_1 \neq \alpha''_2$. Those will be the second half of the masks for the votes.
2. Chooses uniformly at random two values $x''_i, y''_i \in \mathbb{Z}_p^*$ for every voter v_i, with $x''_i \neq x''_j$, $y''_i \neq y''_j$ for all $i \neq j \in \{1 \ldots N\}$.
3. Finally, $\mathcal{A}_2$ commits (see Section 2.3) to the values $g^{\alpha''_1}$, $g^{\alpha''_2}$, and for every $i \in \{1 \ldots N\}$ it commits to the pairs $(v_i, g^{x''_i})$, $(v_i, g^{y''_i})$.

It is important that all the values x''_i, y''_i, α''_l remain private.

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

- $\mathcal{A}_1$ publishes the decommitments for the values g^k, g^λ, $g^{\alpha'_1}$, $g^{\alpha'_2}$, and the pairs $(v_i, g^{x'_i})$, $(v_i, g^{y'_i})$ for every $i \in \{1 \ldots N\}$.
- $\mathcal{A}_2$ publishes decommitments for the values $g^{\alpha''_1}$, $g^{\alpha''_2}$, and the pairs $(v_i, g^{x''_i})$, $(v_i, g^{y''_i})$ for every $i \in \{1 \ldots N\}$.

3.1.2. *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.

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

For every voter v_i, 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 both authorities without fear that any adversary can eavesdrop.
2. Alice proves to both authorities that she controls her wallet (e.g. signing a challenge message with the wallet's private key), and the authorities associate the wallet address to v_i in their respective voters lists.
3. $\mathcal{A}_1$ creates the preliminary ballot:

$$\bar{b}_i = \left(g^{y_i'(x_i'+k)}, g^{y_i'(x_i'+\lambda)}\right) \tag{4}$$

   and sends it to Alice.
4. $\mathcal{A}_1$ gives Alice the values $g^{x_i'y_i'}$, $g^{y_i'k}$, $g^{y_i'\lambda}$, and proves the correctness of the preliminary ballot with the Schnorr ZKP presented in Section 2.2.1:
   (a) First $\mathcal{A}_1$ proves that $g^{x_i'}$, $g^{y_i'}$, and $g^{x_i'y_i'}$ are correctly related using:

$$\omega = x_i', \qquad u = g, \qquad z = g^{x_i'}, \qquad \bar{u} = g^{y_i'}, \qquad \bar{z} = g^{y_i'x_i'}, \tag{5}$$

   and:

$$\omega = y_i', \qquad u = g, \qquad z = g^{y_i'}, \qquad \bar{u} = g^{x_i'}, \qquad \bar{z} = g^{y_i'x_i'}. \tag{6}$$

   (b) Then $\mathcal{A}_1$ proves that $g^{y_i'k}$ and $g^{y_i'\lambda}$ are correctly linked to $g^{y_i'}$ and to g^k and g^λ respectively, using:

$$\omega = k, \qquad u = g, \qquad z = g^{k}, \qquad \bar{u} = g^{y_i'}, \qquad \bar{z} = g^{y_i'k}, \tag{7}$$

   and:

$$\omega = \lambda, \qquad u = g, \qquad z = g^{\lambda}, \qquad \bar{u} = g^{y_i'}, \qquad \bar{z} = g^{y_i'\lambda}. \tag{8}$$

   (c) Finally Alice can check that:

$$\bar{b}_{i,1} = g^{y_i'x_i'} \cdot g^{y_i'k}, \tag{9}$$

$$\bar{b}_{i,2} = g^{y_i'x_i'} \cdot g^{y_i'\lambda}. \tag{10}$$

5. $\mathcal{A}_2$ gives Alice the value $g^{x_i''y_i''}$ and proves that it is correct with the Schnorr ZKP presented in Section 2.2.1 and using:

$$\omega = x_i'', \qquad u = g, \qquad z = g^{x_i''}, \qquad \bar{u} = g^{y_i''}, \qquad \bar{z} = g^{y_i''x_i''}, \tag{11}$$

   and:

$$\omega = y_i'', \qquad u = g, \qquad z = g^{y_i''}, \qquad \bar{u} = g^{x_i''}, \qquad \bar{z} = g^{y_i''x_i''}. \tag{12}$$

6. $\mathcal{A}_2$ sends $g^{y_i''x_i''}$ to $\mathcal{A}_1$. Defining $y_i := y_i' \cdot y_i''$, $\mathcal{A}_1$ computes $g^{y_ix_i''} = (g^{y_i''x_i''})^{y_i'}$, sends it to Alice, and proves to Alice its correctness using:

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

7. Alice flips a random coin $c_i \in \{0,1\}$ and defines $\tilde{b}_i = (\bar{b}_{i,1}, \bar{b}_{i,2}) = \bar{b}_i$ if $c_i = 0$, $\tilde{b}_i = (\bar{b}_{i,2}, \bar{b}_{i,1})$ if $c_i = 1$, and sends $\tilde{b}_i$ and $g^{y_ix_i''}$ to $\mathcal{A}_2$. Note that, thanks to c_i, $\mathcal{A}_2$ does not know which token is valid and which is fake.

8. $\mathcal{A}_2$ contacts $\mathcal{A}_1$ to have confirmation that $g^{y_i x_i''}$ is correct and that $\tilde{b}_i$ is a permutation of $\bar{b}_i$ (without learning which component is valid and which is fake), then computes $(\tilde{b}_{i,l})^{y_i''}$ for $l \in \{1,2\}$ and proves to Alice their correctness using:

$$\omega = y_i'', \qquad u = g, \qquad z = g^{y_i''}, \qquad \bar{u} = \tilde{b}_{i,l}, \qquad \bar{z} = (\tilde{b}_{i,l})^{y_i''}. \tag{14}$$

9. Finally, $\mathcal{A}_2$ flips a random coin $c_i' \in \{0,1\}$ and sends to Alice's wallet her final ballot b_i that is defined as:

$$b_i = \begin{cases} \left((\tilde{b}_{i,1})^{y_i''} \cdot g^{y_i x_i''}, (\tilde{b}_{i,2})^{y_i''} \cdot g^{y_i x_i''}\right) & \text{if } c_i' = 0, \\ \left((\tilde{b}_{i,2})^{y_i''} \cdot g^{y_i x_i''}, (\tilde{b}_{i,1})^{y_i''} \cdot g^{y_i x_i''}\right) & \text{if } c_i' = 1. \end{cases} \tag{15}$$

If we define $x_i := x_i' + x_i''$ then for $l \in \{1,2\}$ we have that $b_{i,l} = g^{y_i(x_i+\sigma_{i,l})}$, with $\sigma_{i,l} \in \{k, \lambda\}, \sigma_{i,1} \neq \sigma_{i,2}$. Note that Alice, thanks to the proofs and the knowledge of the intermediate values, knows which one is the valid token (the one with $\sigma_{i,l} = k$), but thanks to c_i, c_i' neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ can distinguish the tokens unless they collude.

3.1.3. *Voting Phase.* Voters express their preference sending the valid token to the preferred candidate, and the fake token to the other candidate. The two *v-tokens* are sent with a transaction on the blockchain to the respective candidates. As stated in Section 2.4, blockchain rules allow only votes that send one token to the first candidate and one to the second, 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), moreover the assumed properties of the blockchain guarantee that no vote is changed or deleted.

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

In order to count the votes, the authorities have to process the tokens received by each candidate, substituting the *voter's mask* y_i with the appropriate *candidate mask* $\alpha_l = \alpha_l' \alpha_l''$. 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. 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 associated person.

This phase is symmetrical for the two candidates, and note that it is sufficient to count the preferences collected by only one candidate since the number of total votes is known, so we will describe just the tallying of the first candidate's votes.

For $1 \leq i \leq T$, let l_i the index of the token that the voter v_i sent to the first candidate, that is $b_{i,l_i} = g^{y_i(x_i+\sigma_{i,l_i})}$, then the authorities process the token b_{i,l_i} performing the following steps:

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

$$\bar{t}_{1,i} = b_{i,l_i}^{\frac{\alpha_1'}{y_i'}} = \left(g^{y_i(x_i+\sigma_{i,l_i})}\right)^{\frac{\alpha_1'}{y_i'}} = g^{\alpha_1' \cdot y_i''(x_i+\sigma_{i,l_i})}, \tag{16}$$

and registers it on the blockchain.

2. Any observer could ask for a proof that this computation is correct. $\mathcal{A}_1$ proves that $\bar{t}_{1,i}$ is correct using:

$$\omega = \frac{\alpha_1'}{y_i'}, \qquad u = g^{y_i'}, \qquad z = g^{\alpha_1'}, \qquad \bar{u} = b_{i,1}, \qquad \bar{z} = \bar{t}_{1,i}. \tag{17}$$

3. $\mathcal{A}_2$ then computes the final vote $t_{1,i}$ as:

$$t_{1,i} = \bar{t}_{1,i}^{\frac{\alpha_1''}{y_i''}} = \left(g^{\alpha_1' \cdot y_i''(x_i+\sigma_{i,l_i})}\right)^{\frac{\alpha_1''}{y_i''}} = g^{\alpha_1(x_i+\sigma_{i,l_i})}, \tag{18}$$

and registers it on the blockchain (remember that $\alpha_1 := \alpha_1' \cdot \alpha_1''$).

4. Again, any observer could ask for a proof that this computation is correct. $\mathcal{A}_2$ proves that $t_{1,i}$ is correct using:

$$\omega = \frac{\alpha_1''}{y_i''}, \qquad u = g^{y_i''}, \qquad z = g^{\alpha_1''}, \qquad \bar{u} = \bar{t}_{1,i}, \qquad \bar{z} = t_{1,i}. \tag{19}$$

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 $T = R_1 + F_1 = R_2 + F_2$, $R_1 = F_2$, $R_2 = F_1$. The count R_1 can be computed with the following steps:

1. $\mathcal{A}_1$ computes and publishes $g^{\alpha_1} = \left(g^{\alpha_1''}\right)^{\alpha_1'}$, $g^{\alpha_1 k}$, $g^{\alpha_1 \lambda}$.
   $\mathcal{A}_1$ proves that g^{α_1} is correct using:

$$\omega = \alpha_1', \qquad u = g, \qquad z = g^{\alpha_1'}, \qquad \bar{u} = g^{\alpha_1''}, \qquad \bar{z} = g^{\alpha_1}, \tag{20}$$

   then proves that $g^{\alpha_1 k}$ is correct using:

$$\omega = k, \qquad u = g, \qquad z = g^{k}, \qquad \bar{u} = g^{\alpha_1}, \qquad \bar{z} = g^{\alpha_1 k}, \tag{21}$$

   and finally that $g^{\alpha_1 \lambda}$ is correct using:

$$\omega = \lambda, \qquad u = g, \qquad z = g^{\lambda}, \qquad \bar{u} = g^{\alpha_1}, \qquad \bar{z} = g^{\alpha_1 \lambda}. \tag{22}$$

2. Then $\mathcal{A}_1$ computes $\sum_{i=1}^{T} x_i'$, and publishes $g^{\alpha_1 \sum_{i=1}^{T} x_i'}$.
3. Note that any observer could 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 $g^{\alpha_1 \sum_{i=1}^{T} x_i'}$ is correct using:

$$\omega = \sum_{i=1}^{T} x_i', \quad u = g, \quad z = g^{\sum_{i=1}^{T} x_i'}, \quad \bar{u} = g^{\alpha_1}, \quad \bar{z} = g^{\alpha_1 \sum_{i=1}^{T} x_i'}. \tag{23}$$

4. Similarly, $\mathcal{A}_2$ computes $\sum_{i=1}^{T} x_i''$ and publishes $g^{\alpha_1 \sum_{i=1}^{T} x_i''}$.
5. Again, any observer could 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 $g^{\alpha_1 \sum_{i=1}^{T} x_i''}$ is correct using:

$$\omega = \sum_{i=1}^{T} x_i'', \qquad u = g, \quad z = g^{\sum_{i=1}^{T} x_i''}, \quad \bar{u} = g^{\alpha_1}, \quad \bar{z} = g^{\alpha_1 \sum_{i=1}^{T} x_i''}. \tag{24}$$

6. Note that any observer could compute the value:

$$g^{\alpha_1(\sum_{i=1}^{T} x_i + R_1 k + F_1 \lambda)} = \prod_{i=1}^{T} t_{1,i}, \tag{25}$$

given that:

$$g^{\alpha_1 \sum_{i=1}^{T} x_i} = g^{\alpha_1 \sum_{i=1}^{T} (x'_i + x''_i)} = g^{\alpha_1 \sum_{i=1}^{T} x'_i} \cdot g^{\alpha_1 \sum_{i=1}^{T} x'_i}, \tag{26}$$

anyone can compute:

$$\left(g^{\alpha_1 k}\right)^{R_1} \cdot \left(g^{\alpha_1 \lambda}\right)^{F_1} = \left(g^{\alpha_1 \sum_{i=1}^{T} x_i}\right)^{-1} \cdot g^{\alpha_1 (\sum_{i=1}^{T} x_i + R_1 k + F_1 \lambda)}. \tag{27}$$

7. Finally note that now R_1 and F_1 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

In this section we prove the security of the proposed protocol and its validity as a voting system.

4.1. Vote indistinguishability.

The goal is to prove that an adversary cannot distinguish between valid 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 by observing the results. Therefore, we assume that the adversary controls one authority and all but two voters, and 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.

4.1.1. *Security Model.* The security of the protocol will be proven in terms of vote indistinguishability (VI), as it will be detailed in Definition 14.

The security of the protocol will be proved in the presence of at least one honest authority, so the simulator in the proof will take on the roles of both the honest authority and the two voters that the adversary does not control. To simplify our analysis we assume that the adversary $\mathcal{A}$ controls one authority, which is in favor of the adversary, and $\mathcal{A}$ does not intentionally fail decommitments or ZKPs, otherwise the protocol would abort and $\mathcal{A}$ would not win the security game with non-negligible advantage (in fact, since it outputs its guess only once the protocol has correctly terminated, $\mathcal{A}$ must run the protocol smoothly).

Definition 13 (Security Game)**.** The security game for a two-candidate protocol proceeds as follows:

- **Init.** The adversary $\mathcal{A}$ chooses the authority and the $N-2$ users that it controls and therefore $\mathcal{A}$ knows which are the valid and fake *v-tokens* of these users. The remaining two are called *free voters*. The challenger $\mathcal{C}$ takes the role of the other authority 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.** $\mathcal{A}$ votes with some or all of the voters it controls.
- **Challenge.** Let C_0 and C_1 be the two candidates, $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and votes with the *v-tokens* of the free voters accordingly: the first free voter votes for C_μ, the second one for $C_{\mu \oplus 1}$.
- **Phase 2.** $\mathcal{A}$ 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 $\mathcal{A}$ can request the ZKPs of the correctness of the computations done by $\mathcal{C}$.
- **Guess.** $\mathcal{A}$ outputs a guess μ' of the coin flip that randomly assigned the voting preferences of the two free voters[5].

Definition 14 (Vote Indistinguishability)**.** A Two-Candidates 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 13), there exists a negligible function η such that:

$$\mathbb{P}[\mu' = \mu] \leq \frac{1}{2} + \eta(\theta). \tag{28}$$

In the following theorem we prove our voting protocol VI-secure under the DDH assumption (Definition 2) in the security game defined above.

Theorem 15. *If the DDH assumption of Definition 2 holds, then the protocol described in Section 3.1 is VI-secure, as per Definition 14.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$ that can guess μ with advantage ε, i.e. $\mathbb{P}[\mu' = \mu] \geq \frac{1}{2} + \varepsilon$. We will show how a simulator $\mathcal{S}$ can play the DDH game with advantage $\frac{\varepsilon}{2}$ interacting with $\mathcal{A}$ (also rewinding $\mathcal{A}$'s execution). The simulator starts with considering a DDH challenge:

$$(g, A = g^a, B = g^b, T), \tag{29}$$

with $T = g^{ab}$ or $T = R = g^{\xi}$. We will consider separately the case when $\mathcal{A}$ controls $\mathcal{A}_2$ and when $\mathcal{A}$ controls $\mathcal{A}_1$.

First we consider the case in which the adversary controls $\mathcal{A}_2$, 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**. The simulator $\mathcal{S}$ chooses uniformly at random in $\mathbb{Z}_p^*$ the values $\bar{x}_1$, $\bar{x}_2$, $\bar{y}_1$, $\bar{y}_2$, k, λ, $\bar{\alpha}_1$, $\bar{\alpha}_2$. Then $\mathcal{S}$ chooses uniformly at random y_i', x_i' for $3 \leq i \leq N$. Finally $\mathcal{S}$ implicitly sets:

$$x_1' = \bar{x}_1 + b, \qquad x_2' = \bar{x}_2 - b, \tag{30}$$

$$y_1' = a \cdot \bar{y}_1, \qquad y_2' = a \cdot \bar{y}_2, \tag{31}$$

$$\alpha_1' = a \cdot \bar{\alpha}_1, \qquad \alpha_2' = a \cdot \bar{\alpha}_2 \tag{32}$$

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 the values $g^{x_1'}$, $g^{x_2'}$, $g^{y_1'}$, $g^{y_2'}$, $g^{\alpha_1'}$, $g^{\alpha_2'}$ as follows:

$$g^{x_1'} = g^{\bar{x}_1} \cdot B, \qquad g^{x_2'} = g^{\bar{x}_2} \cdot B^{-1}, \tag{33}$$

$$g^{y_1'} = A^{\bar{y}_1}, \qquad g^{y_2'} = A^{\bar{y}_2}, \tag{34}$$

$$g^{\alpha_1'} = A^{\bar{\alpha}_1}, \qquad g^{\alpha_2'} = A^{\bar{\alpha}_2}, \tag{35}$$

[5]In other words, for each of the four *v-tokens* outside its control, $\mathcal{A}$ guesses if the *v-token* is `valid` or not.

while the other values can be computed following the protocol, so $\mathcal{S}$ perfectly simulates the actions that $\mathcal{A}_1$ would perform in the setup phase.

- **Registrar Phase**. For the v_i with $3 \leq i \leq N$, $\mathcal{S}$ can simulate $\mathcal{A}_1$ following the protocol, while for v_1 and v_2 the simulation is carried on as follows:
  1. $\mathcal{S}$ computes the preliminary ballot as:

$$\bar{b}_1 = \left(A^{\bar{y}_1(\bar{x}_1+k)} \cdot T^{\bar{y}_1}, A^{\bar{y}_1(\bar{x}_1+\lambda)} \cdot T^{\bar{y}_1}\right), \tag{36}$$

$$\bar{b}_2 = \left(A^{\bar{y}_2(\bar{x}_2+k)} \cdot T^{-\bar{y}_2}, A^{\bar{y}_2(\bar{x}_2+\lambda)} \cdot T^{-\bar{y}_2}\right). \tag{37}$$

     Since it controls the corresponding voter, fortunately $\mathcal{S}$ does not need to perform the ZKPs (which would fail).
  2. $\mathcal{A}$ gives to $\mathcal{S}$ the value $g^{x_i'' y_i''}$ and proves that it is correct with the Schnorr ZKP, so $\mathcal{S}$ can extract the values x_i'', y_i'', thanks to Lemma 1, so $\mathcal{S}$ can compute:

$$g^{y_i x_i''} = A^{\bar{y}_i \cdot y_i'' \cdot x_i''}. \tag{38}$$

  3. $\mathcal{S}$ follows the protocol: it flips a random coin $c_i \in \{0,1\}$, computes $\tilde{b}_i$ accordingly and sends $\tilde{b}_i$ and $g^{y_i x_i''}$ to $\mathcal{A}$.
  4. $\mathcal{S}$ continues to interact with $\mathcal{A}$ to complete the phase. Note that even knowing k and λ the adversary cannot distinguish valid and fake tokens, while $\mathcal{S}$ can tell that the valid tokens in b_i are equal to:

$$A^{\bar{y}_1 \cdot y_1''(\bar{x}_1+x_1''+k)} \cdot T^{\bar{y}_1 \cdot y_1''} \qquad \text{for } v_1, \tag{39}$$

$$A^{\bar{y}_2 \cdot y_2''(\bar{x}_2+x_2''+k)} \cdot T^{-\bar{y}_2 \cdot y_2''} \qquad \text{for } v_2, \tag{40}$$

     while the fake tokens are equal to:

$$A^{\bar{y}_1 \cdot y_1''(\bar{x}_1+x_1''+\lambda)} \cdot T^{\bar{y}_1 \cdot y_1''} \qquad \text{for } v_1, \tag{41}$$

$$A^{\bar{y}_2 \cdot y_2''(\bar{x}_2+x_2''+\lambda)} \cdot T^{-\bar{y}_2 \cdot y_2''} \qquad \text{for } v_2. \tag{42}$$

     Note that if $T = g^{ab}$ then the simulator $\mathcal{S}$ gives a perfect simulation.
- **Phase 1**. The adversary chooses some voters i, $3 \leq i \leq N$ and casts their votes.
- **Challenge**. $\mathcal{S}$ flips a coin μ, and sends the tokens accordingly, as described in Definition 13.
- **Phase 2**. Trivial as Phase 1.
- **Tallying**. Without loss of generality, only the v_is with $1 \leq i \leq T$ voted. For $3 \leq i \leq T$, $\mathcal{S}$ can follow the protocol. $\mathcal{S}$ carries on the simulation as follows:
  1. $\mathcal{S}$ computes and proves the correctness of the preliminary votes following the protocol without problems. In fact, for $i, l \in \{1,2\}$, we have that

$$\frac{\alpha_l'}{y_i'} = \frac{a\bar{\alpha}_l}{a\bar{y}_i} = \frac{\bar{\alpha}_l}{\bar{y}_i} \qquad \forall l \in [M], \tag{43}$$

     and these values are known to $\mathcal{S}$.
  2. Conversely $\mathcal{S}$ can ask $\mathcal{A}$ to prove that the values $t_{l,i}$ are correct, and therefore extract the values $\frac{\alpha_l''}{y_i''}$, and compute α_l'', since the values y_i'' have already been extracted from $\mathcal{A}$.
  3. $\mathcal{S}$ computes and publishes

$$g^{\alpha_1} = A^{\bar{\alpha}_1 \alpha_1''}, \qquad g^{\alpha_1 k} = A^{\bar{\alpha}_1 \alpha_1'' k}, \qquad g^{\alpha_1 \lambda} = A^{\bar{\alpha}_1 \alpha_1'' \lambda}. \tag{44}$$

Since the simulator does not know the actual value of $\omega = \alpha_1'$ to use in the proof of g^{α_1}, then $\mathcal{S}$ has to simulate the Schnorr protocol (possible thanks to Corollary 1). The other two proofs can be performed normally since k and λ are known.

4. $\mathcal{S}$ can compute:

$$\sum_{i=1}^{T} x_i' = \bar{x}_1 + \bar{x}_2 + \sum_{i=3}^{T} x_i', \tag{45}$$

so for the rest of the tallying phase $\mathcal{S}$ follows the protocol. In particular notice that, for $l \in \{1, 2\}$:

$$\prod_{i=1}^{T} t_{l,i} = A^{\bar{\alpha}_l \alpha_l'' (\bar{x}_1 + \bar{x}_2 + x_1'' + x_2'' + \sum_{i=3}^{T} x_i + R_l k + F_l \lambda)}, \tag{46}$$

that is T does not appear in the tally, so it is always correct.

- **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 in $\mathbb{G}$.

Now we consider the case in which the adversary controls $\mathcal{A}_1$. In the following we describe the differences from the first simulation.

- **Setup**. The simulator chooses uniformly at random in $\mathbb{Z}_p^*$ the values $\bar{x}_1$, $\bar{x}_2$, $\bar{y}_1$, $\bar{y}_2$, $\bar{\alpha}_1$, $\bar{\alpha}_2$, then it chooses uniformly at random y_i'', x_i'' for $3 \leq i \leq N$. Finally $\mathcal{S}$ implicitly sets:

$$x_1'' = \bar{x}_1 + b, \qquad x_2'' = \bar{x}_2 - b, \tag{47}$$

$$y_1'' = a \cdot \bar{y}_1, \qquad y_2'' = a \cdot \bar{y}_2, \tag{48}$$

$$\alpha_1'' = a \cdot \bar{\alpha}_1, \qquad \alpha_2'' = a \cdot \bar{\alpha}_2. \tag{49}$$

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 the values $g^{x_1'}$, $g^{x_2'}$, $g^{y_1'}$, $g^{y_2'}$, $g^{\alpha_1'}$, $g^{\alpha_2'}$ as follows:

$$g^{x_1''} = g^{\bar{x}_1} \cdot B, \qquad g^{x_2''} = g^{\bar{x}_2} \cdot B^{-1}, \tag{50}$$

$$g^{y_1''} = A^{\bar{y}_1}, \qquad g^{y_2''} = A^{\bar{y}_2}, \tag{51}$$

$$g^{\alpha_1''} = A^{\bar{\alpha}_1}, \qquad g^{\alpha_2''} = A^{\bar{\alpha}_2}, \tag{52}$$

while the other values can be computed following the protocol, so $\mathcal{S}$ perfectly simulates the actions that $\mathcal{A}_1$ would perform in the setup phase.

- **Registrar Phase**. For $i \in \{1, 2\}$ the simulation is carried on as follows:
  1. From the ZKPs that $\mathcal{A}$ has to perform on behalf of $\mathcal{A}_1$, $\mathcal{S}$ extracts the values x_i', y_i', k, λ.
  2. $\mathcal{S}$ computes:

$$g^{y_1'' \cdot x_1''} = A^{\bar{y}_1 \cdot \bar{x}_1} T^{\bar{y}_1}, \tag{53}$$

$$g^{y_2'' \cdot x_2''} = A^{\bar{y}_2 \cdot \bar{x}_2} T^{-\bar{y}_2}. \tag{54}$$

3. Without loss of generality suppose that $c_i \oplus c'_i = 0$ (i.e. the valid token is in first position), then $\mathcal{S}$ computes the final ballots as:

$$b_1 = \left(A^{y'_1 \cdot \bar{y}_1 (x'_1 + \bar{x}_1 + k)} \cdot T^{y'_1 \cdot \bar{y}_1}, A^{y'_1 \cdot \bar{y}_1 (x'_1 + \bar{x}_1 + \lambda)} \cdot T^{y'_1 \cdot \bar{y}_1}\right), \tag{55}$$

$$b_2 = \left(A^{y'_2 \cdot \bar{y}_2 (x'_2 + \bar{x}_2 + k)} \cdot T^{y'_2 \cdot \bar{y}_2}, A^{y'_2 \cdot \bar{y}_2 (x'_2 + \bar{x}_2 + \lambda)} \cdot T^{-y'_2 \cdot \bar{y}_2}\right). \tag{56}$$

Again, note that if $T = g^{ab}$ then the simulator $\mathcal{S}$ gives a perfect simulation.

- **Tallying**. For $l, i \in \{1, 2\}$, $\mathcal{S}$ carries on the simulation as follows:
  1. $\mathcal{S}$ extracts from the ZKPs of the preliminary votes the values $\frac{\alpha'_l}{y'_i}$, and compute α'_l, since the values y'_i have already been extracted from $\mathcal{A}$, and therefore compute α_l also.
  2. $\mathcal{S}$ computes and proves the correctness of the final votes following the protocol without problems. In fact, for $i, l \in \{1, 2\}$, we have that

$$\frac{\alpha''_l}{y''_i} = \frac{a\bar{\alpha}_l}{a\bar{y}_i} = \frac{\bar{\alpha}_l}{\bar{y}_i} \qquad \forall l \in [M], \tag{57}$$

  and these values are known to $\mathcal{S}$.
  3. $\mathcal{S}$ can compute:

$$\sum_{i=1}^{T} x''_i = \bar{x}_1 + \bar{x}_2 + \sum_{i=3}^{T} x''_i, \tag{58}$$

  so for the rest of the tallying phase $\mathcal{S}$ follows the protocol. In particular notice that, for $l \in \{1, 2\}$:

$$\prod_{i=1}^{T} t_{l,i} = A^{\alpha'_l \bar{\alpha}_l (x'_1 + x'_2 + \bar{x}_1 + \bar{x}_2 + \sum_{i=3}^{T} x_i + R_l k + F_l \lambda)}, \tag{59}$$

  that is T does not appear in the tally, so it is always correct.

The steps omitted are either identical to the first simulation, or can be performed following the protocol because the simulator knows all the necessary values.

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

$$\mathbb{P}[\mathcal{S}(g, A, B, T = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{60}$$

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 from the values that would have been computed 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, then we have:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = R) = 0] = \frac{1}{2}. \tag{61}$$

Therefore, $\mathcal{S}$ can play the DDH game with non-negligible advantage $\frac{\varepsilon}{2}$. As already mentioned, when T is not random the simulator $\mathcal{S}$ gives a perfect simulation in both cases, this means that the advantage is preserved and so it holds that:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{62}$$

On the contrary when T is a random element $R \in \mathbb{G}$ the adversary gains no information, so:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = R) = 0] = \frac{1}{2}. \tag{63}$$

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

4.2. GENERAL PROPERTIES OF THE PROTOCOL. In this section we prove that the protocol described in Section 3.1 satisfies the general properties of a vote system introduced in Section 2.5.

Proposition 4 (Correctness)**.** *If the underlying blockchain is* append-only*, checks the authorization of transactions, accepts only valid votes, and prevents double spending of tokens (as described in Section 2.4), then the protocol is correct, as per Definition 6.*

Proof. This property derives directly from the properties of the underlying blockchain: since it is append-only cast votes cannot be altered or erased, and the tokens of an uncoerced voter cannot be spent by an adversary since the consensus rule accepts only properly authorized transactions. Moreover, the consensus rules assure that only valid votes are cast, and prevents any voter to vote twice. □

Proposition 5 (Fairness)**.** *If the DDH assumption holds, then the protocol is fair, as per Definition 7.*

Proof. Thanks to Theorem 15, if the DDH assumption holds the protocol has vote-indistinguishability. Therefore, the votes cast do not reveal how many preferences each candidate has received until they are processed by the authorities, and this does not happen until the voting phase has ended. Even the processed votes do not reveal such information until the tallying values $g^{\alpha_l \sum x_i}$ are published by the authorities in the steps 2 and 4 of tallying, at which point the end results are computable by anyone and therefore public. □

Proposition 6 (Transparency)**.** *If the underlying blockchain is public (as described in Section 2.4), then the protocol is transparent, as per Definition 8.*

Proof. Since the mathematical background of the whole process is public, everyone can audit it, so every part can be understood. If the blockchain is public then anyone can check the progress of the voting processes. □

Proposition 7 (Privacy)**.** *If the DDH assumption holds, the underlying blockchain is public and append-only (as described in Section 2.4), then the protocol is private, as per Definition 9.*

Proof. Thanks to Theorem 15, if the DDH assumption holds the protocol has vote-indistinguishability, so a vote (even processed for the tallying) does not reveal the preference expressed by the voter, it only reveals that this voter actually voted (note that in normal elections this information is often public). Moreover, if the protocol uses pseudonymous wallets that can not be linked to the real identity of the voter then the privacy is completely preserved. □

Proposition 8 (Verifiability)**.** *If the DDH assumption holds, and the blockchain is public, then the protocol is verifiable, as per Definition 10.*

Proof. Since the blockchain is *public*, anyone can check that every vote has been cast correctly by browsing the blockchain ledger. Moreover, anyone can see the processed votes received by the candidates and count the preferences using the values published by the authorities during the tallying. If the DDH assuption holds the Zero-Knowledge proofs allow everyone to do a consistency check of the computations performed at every step, so universal verifiability holds.

Similarly, the ZKP performed during the Registrar phase allow the voters to verify which of their tokens is valid, then analyzing the blockchain and the tallying values can check that their preferences have been correctly expressed and counted in the final result. □

Proposition 9 (Vote-Selling and Coercion Resistance)**.** *If the DDH assumption holds, then the protocol is vote-selling and coercion resistant, as per Definitions 11 and 12.*

Proof. Thanks to Theorem 15, 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 can do so with the additional information $\zeta = c_i \oplus c'_i$ (i.e. the coin flips that shuffle the ballot during its generation), that is known only to the voter since the ballot generation is performed in a safe environment. Note that $\zeta \in \{0, 1\}$ and both values are admissible and equiprobable, thus the voter can easily fabricate a value ζ' that possibly flips the vote to fake compliance with the coercer's choice. 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. Generalization to multiple candidates

The protocol presented here is limited to the case of two candidates, however the same approach can be generalized to elections with multiple candidates, but extra care is required in ballot creation.

With $M > 2$ distinct candidates, the ballot b_i must comprehend M *v-tokens*, one valid and the others fake. This means that there are multiple fake tokens with the same $(x_i + \lambda)$, so the voter masks y_i must become lists $(y_{i,1}, \ldots, y_{i,M})$ with $y_{i,l} \neq y_{i,l'}$ for all $l \neq l' \in \{1, \ldots, M\}$ in order to properly conceal fake tokens.

This obviously complicates the shuffling of ballots, but with the introduction of a third authority and the addition of a couple of rounds in the Registrar phase it is possible to construct a generalization of the protocol that securely supports multiple candidates, with a computational cost that scales linearly with the number of candidates.

We are currently finalizing its construction and security proofs. We are confident in our construction since a preliminary version of this voting protocol, including the generalization to multiple candidates, has been developed complete with the security proofs, for the MSC thesis [26] of the first author (supervised by the other two authors).

Finally, note that with a multiple-candidate protocol we allow voting with a blank ballot, simply by adding a dummy candidate that represents a blank choice.

6. Conclusions and final remarks

In the protocol proposed here, an underlying blockchain infrastructure and a system of ZKPs ensure transparency and full auditability of the whole process. In

fact, every computation involving secret material is proved correct, and the public ledger given by the blockchain assures immutability of the past and a public trail.

Although this system can be implemented on any blockchain, one should note that the most common solutions require a sizeable amount of cryptocurrency to run the protocol (e.g. for initialising and transferring the voting tokens). In our application context, these costs could discourage voters to express their preferences, so an ad-hoc ledger is probably preferable.

The protocol also achieves extensive security properties, including *coercion* and *vote-selling* resistance, while retaining receipts. Indeed, our approach disguises valid and fake votes, which remain indistinguishable even after tallying and even considering a corrupt authority. Since only the actual voter can access the authenticated and secure environment in which the tokens are proven valid or false, and the proof transcript can be forged to state anything (as per Zero-Knowledge properties), then no one else can be sure that a token is actually valid or fake. This means that the voter cannot sell the vote and adversaries cannot coerce a specific vote, unless they are content with expressing a random preference.

The passage through a secure and authenticated environment is quite of a trouble, but to the best of our knowledge there is no other way to achieve coercion resistance, since we need a moment in which voter and adversary are separated. However, compared to traditional voting booths the trouble is arguably reduced: in fact the Registrar phase could begin many months before the election and last until voting ends, giving voters much more flexibility.

The work carried on by the two authorities is essential to the execution of the protocol, so it may seem vulnerable to DOS attacks. However, the Setup and Registrar phases can be executed well ahead of the election. Moreover, the Setup can be done asynchronously by the two authorities, and the Registrar phase can last a long time, and since it is infeasible to sustain an effective DOS attack for extended periods of time, the voters should have ample opportunities to register themselves. Note also that the authorities are not involved in the Voting phase, which (relying on the decentralised blockchain) is difficult to disrupt on a large scale. In the end, as long as the authorities are able to eventually go online for tallying, we can get the election results.

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 between various pairs of independent authorities, each 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 damage in case both authorities are 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.

Acknowledgments

The core of this work is derived from the results contained in the MSC thesis of the first author, who thanks her supervisors (the other two authors). The second author is a member of the INdAM Research group GNSAGA.
We would like to thank Vincenzo di Nicola for the initial input, Alessio Meneghetti for his support, Carla Mascia for her revision and feedback, and huge thanks to Gaetano Russo that worked on the implementation and provided valuable feedback.

References

[1] B. Adida, Helios: Web-based open-audit voting, in *USENIX Security Symposium*, **17** (2008), 335–348.

[2] R. Alyoshkin, *Polys Whitepaper*, Available from: https://docs.polys.me/en/collections/699457-technology-whitepaper.

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich et al., Hyperledger fabric: A distributed operating system for permissioned blockchains, in *Proceedings of the Thirteenth EuroSys Conference*, ACM, (2018), 1–15.

[4] L. Babai and S. Moran, Arthur-merlin games: A randomized proof system, and a hierarchy of complexity class, *Journal of Computer and System Sciences*, **36** (1988), 254–276.

[5] M. Blum, P. Feldman and S. Micali, Non-interactive zero-knowledge and its applications, in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC, ACM, (1988), 103–112.

[6] J.-M. Bohli, J. Müller-Quade and S. Röhrich, Bingo voting: Secure and coercion-free voting using a trusted random number generator, in *Proceedings of International Conference on E-Voting and Identity*, Springer, 2007, 111–124.

[7] G. Brassard, D. Chaum and C. Crépeau, Minimum disclosure proofs of knowledge, *Journal of Computer and System Sciences*, **37** (1988), 156–189.

[8] A. Cardillo, N. Akinyokun and A. Essex, Online voting in ontario municipal elections: A conflict of legal principles and technology?, in *Electronic Voting* (eds. R. Krimmer, M. Volkamer, V. Cortier, B. Beckert, R. Küsters, U. Serdült and D. Duenas-Cid), Springer International Publishing, Cham, (2019), 67–82.

[9] M. R. Clarkson, S. Chong and A. C. Myers, Civitas: Toward a secure voting system, in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP, IEEE Computer Society, (2008), 354–368,

[10] D. Costa, F. Fiori, P. Milan, M. Sala, A. Vitale and M. Vitale, *Quadrans Whitepaper*, 2019. Available from: https://quadrans.io/content/files/quadrans-white-paper-rev01.pdf.

[11] L. Fouard, M. Duclos and P. Lafourcade, *Survey on Electronic Voting Schemes*, 2007.

[12] F. Fusco, M. I. Lunesu, F. Pani and A. Pinna, Crypto-voting, a blockchain based e-voting system, in *10th International Conference on Knowledge Management and Information Sharing*, (2018), 223–227.

[13] N. Gailly, P. Jovanovic, B. Ford, J. Lukasiewicz and L. Gammar, *Agora: Bringing our Voting Systems into the 21st Century*, 2018.

[14] D. Goswami, *United States Postal Service Secure Voting System*, US Patent Application Publication, 2020,

[15] G. Grewal, M. Ryan, S. Bursuc and P. Ryan, Caveat Coercitor: Coercion-Evidence in Electronic Voting, in *Proceedings - IEEE Symposium on Security and Privacy*, IEEE, (2013), 367–381.

[16] S. B. Khairnar, P. S. Naidu and R. Kharat, Secure authentication for online voting system, in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, (2016), 1–4.

[17] K. Koštál, R. Bencel, M. Ries and I. Kotuliak, Blockchain e-voting done right: Privacy and transparency with public blockchain, in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, (2019), 592–595.

[18] H. Li, A. Kankanala and X. Zou, A taxonomy and comparison of remote voting schemes, in *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, IEEE, (2014), 1–8.

[19] Y. Lindell, *Tutorials on the Foundations of Cryptography, Chapter How to Simulate It – A Tutorial on the Simulation Proof Technique*, Springer, 2017.

[20] R. Longo, *Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols*, PhD thesis, University Of Trento, Department of Mathematics, 2018.

[21] P. McCorry, S. Shahandashti and F. Hao, A smart contract for boardroom voting with maximum voter privacy, in *21st International Conference on Financial Cryptography and Data Security (FC17), Sliema, Malta*, (2017), 357–375.

[22] A. Meneghetti, M. Sala and D. Taufer, A Survey on PoW-based Consensus, *Annals of Emerging Technologies in Computing (AETiC)*, **4**, 11 pp.

[23] A. Rea, A. F. D. Kronovet and J. du Rose, *Colony Technical Whitepaper*, 2020. Available from: https://colony.io/whitepaper.pdf.
[24] C. Schnorr, Efficient signature generation by smart cards, *Journal of Cryptology*, **4** (1991), 161–174.
[25] V. Shoup and J. Alwen, Σ-*Protocols Continued and Introduction to Zero Knowledge*, 2007. Available from: https://web.archive.org/web/20170830043326/https://cs.nyu.edu/courses/spring07/G22.3220-001/lec3.pdf.
[26] C. Spadafora, *A New Blockchain-Based Secure E-Voting Protocol*, Master's thesis, University of Trento, Department of Mathematics, Academic Year 2018/2019.
[27] C. Spadafora, R. Longo and M. Sala, *Coercion-Resistant Blockchain-Based E-Voting Protocol*, Cryptology ePrint Archive, Report 2020/674 version 20200605:195929, 2020. Available from: https://eprint.iacr.org/eprint-bin/getfile.pl?entry=2020/674&version=20200605:195929&file=674.pdf.
[28] M. A. Specter, J. Koppel and D. Weitnzer, 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.
[29] F. D. Team, *Introducing a Secure and Transparent Online Voting Solution for Modern Age: Follow My Vote*, Available from: https://followmyvote.com/.
[30] V. D. Team, *Vocdoni Documentation*, Available from: https://docs.vocdoni.io/.
[31] K.-H. Wang, S. K. Mondal, K. Chan and X. Xie, A review of contemporary e-voting: Requirements, technology, systems and usability, *Data Science and Pattern Recognition*, **1** (2017), 31–47.
[32] S. Xiao, X. A. Wang, W. Wang and H. Wang, Survey on blockchain-based electronic voting, in *Advances in Intelligent Networking and Collaborative Systems* (eds. L. Barolli, H. Nishino and H. Miwa), Springer, (2020), 559–567.

Received August 2020; revised February 2021.

E-mail address: c.spadaf@libero.it
E-mail address: riccardolongomath@gmail.com
E-mail address: maxsalacodes@gmail.com