Efficient, Coercion-free and Universally Verifiable Blockchain-based Voting

Tassos Dimitriou, Senior Member, IEEE
tassos.dimitriou@ieee.org

Abstract

Most electronic voting systems today satisfy the basic requirements of privacy, unreusability, eligibility and fairness in a natural and rather straightforward way. However, receipt-freeness, incoercibility and universal verifiability are much harder to implement and in many cases they require a large amount of computation and communication overhead. In this work, we propose a blockchain-based voting system which achieves all the properties expected from secure elections without requiring too much from the voter. Coercion resistance and receipt-freeness are ensured by means of a randomizer token – a tamper-resistance source of randomness which acts as a black box in constructing the ballot for the user. Universal verifiability is ensured by the append-only structure of the blockchain, thus minimizing the trust placed in election authorities. Additionally, the system has linear overhead when tallying the votes, hence it is scalable and practical for large scale elections.

Keywords: Internet voting, blockchains, coercion resistance, receipt-freeness, universal verifiability, commitments, *zkSNARKs*.

1. Introduction

Voting is an essential part of any democratic society, delivering the will of people in a fair and trustworthy manner. Compared to traditional voting systems which have a large overhead in both time and money, electronic voting (e-voting) systems present an alternative that aims to be secure, efficient, convenient and less error-prone. However, e-voting systems introduce several concerns, the most important being the potential for large scale fraud and attacks on privacy.

Existing electronic voting systems mainly fall in two categories [1]. In the first category one finds systems that use a private voting booth where voters can cast their ballots in private (eg. [2, 3, 4], to name a few). Assuming that it is difficult to tap the communication line between the voting booth and the collection server, coercion and vote selling can be prevented. These systems, however, are still not convenient, requiring the presence of a voter in a polling station as in traditional paper-ballot systems. Systems of the second category are internet-based. After an initial registration period, people can cast their votes from any location as long as there is internet access availability ([5, 6, 7, 8, 9]). They use a combination of cryptographic primitives and protocols, such as blind signatures, threshold cryptosystems, mix networks, homomorphic encryption, etc. to satisfy the properties expected from secure elections. While voting of this kind aims to increase user participation and make the whole process more efficient, it also carries with it the potential of abuse at a larger scale.

A voting system is considered secure if it satisfies the following properties:

- *Privacy*: It should be impossible to link a vote to a voter.
- *Completeness*: All valid votes are counted correctly.
- *Soundness*: Invalid votes should be easy to detect and discard.
- *Eligibility*: Only legitimate voters should be able to take part in the election.

- *Unreusability*: Voters can vote only once, thus double-voting is prevented.
- *Fairness*: Early results should not be obtained, as they could influence the vote of the remaining voters.

The basic requirements listed above are relatively easy to implement and are satisfied by most electronic voting systems. Hence, a lot of research has focused on providing a list of extended requirements that, if not satisfied, can undermine the election results. These include:

- *Universal verifiability*: Anyone should be able to verify the fairness of the election and the correct computation of the final tally from submitted ballots.
- *Receipt-freeness*: A voter neither obtains nor is able to construct a receipt on how she voted. This would also prevent vote selling.
- *Coercion-resistance*: A voter cannot be coerced into casting a particular vote as desired by a coercer.

These last two properties are considered very crucial and must be addressed in a fair and democratic election process. While coercion and vote buying also occur in traditional voting systems, these threats are more dangerous in Internet voting schemes as an attacker may exercise influence and control at a larger scale [7].

Coercion is a strong attack on privacy, in which an adversary may instruct voters to reveal their keys or vote in a particular way. On the other hand, a coercion-free system is one in which a voter deceives an adversary into believing she behaved as instructed. This implies receipt-freeness since a voting proof can be used as an avenue for coercion. From a usability point of view it is best if such a system does not demand any strategy on the voter side (e.g. lie, create fake credentials, etc.) This is satisfied in our system; all the voter has to do is simply vote.

Our work uses Bitcoin's blockchain infrastructure to realize an e-voting system that satisfies the properties described above. Participants that use the blockchain network have a single view of all transactions, requiring no trust on any particular entity. Hence the blockchain may be used to provide verifiability which is at the heart of all voting systems, minimizing at the same time the trust placed in election authorities.

Contributions: Compared to existing work, our system has a number of advantages that we highlight here: *i)* it is simple in the sense that all a voter has to do is vote; *ii)* the tallying overhead is linear, making this a scalable and practical system for large scale elections; *iii)* the system remains universally verifiable; *iv)* voter privacy is guaranteed; privacy only depends on the existence of simple anonymous channels where users can submit their votes, and *v)* the protocol ensures receipt freeness and coercion-resistance.

It is important to note that coercion resistance (and by implication receipt-freeness) can be achieved by our scheme without asking the voters to vote multiple times in order to cancel possibly coerced submitted votes ([6, 7, 10]), issue fake credentials ([7, 8]), or rely on the existence of re-randomization networks and re-encryption mixes ([7, 8, 9, 11, 12]). All we need to assume is that the attacker cannot influence the randomness used by the voter to encrypt her vote or authenticate herself. This is achieved through the use of a *randomizer token*, a tamper resistant device that can be instantiated with simple smart cards or TPM enabled devices. Coercion resistance is then possible because the randomness used in the creation of a ballot is *not* under the control of the voter. Thus, the voter cannot use it to re-create a ballot and show how she voted nor can she be coerced to use specific randomness in order to have her vote tracked.

Thus, our scheme does not demand any strategy on the voter side. The voter does not need to lie or produce fake credentials nor does she have to re-vote or deceive the coercer in any way. The voter simply has no way to prove how she voted. Additionally, the use of a blockchain as a public medium to post the ballots helps eliminate trust to election authorities with its built-in verifiability and integrity properties.

Organization: The remainder of the paper is structured as follows. In the next section we review related work on receipt freeness and coercion resistance with emphasis on blockchain voting schemes. Section 3 discusses our voting model and the assumptions we use throughout the paper, while Section 4 highlights the cryptographic primitives used in our proposal. In Section 5, we detail the voting system; its security and efficiency properties are then analyzed and discussed in Section 6. Finally, Section 7 concludes this work.

2. Related work

Electronic voting systems can be thought as end-to-end systems that consume voters' encrypted ballots and produce a final tally along with a proof that the result of the election was computed correctly. Most voting systems use a combination of blind signatures, mix networks and homomorphic encryption techniques. Voting schemes based on blind signatures are generally simple and more efficient since voters can have their ballots blindly signed by the election authority and then submit them through an anonymous channel. These systems, however, are not receipt-free since voters can use the blind factors in their ballots to prove how they voted. Alternatively, a coercer may dictate the factor to be used. Voting schemes based on mix or re-encryption networks are generally less efficient due to the larger number of intermediate steps involved, while systems based on homomorphic encryption require the cooperation of a set of trustworthy tallying authorities to produce the final result.

As mentioned in the introduction one important requirement of e-voting systems is *receipt-freeness*. The concept was introduced in [13] to capture security against vote selling. In a receipt-free system, the voter should not be able to get or construct a receipt proving the contents of her vote. Yet an even more desirable property which implies receipt-freeness is *coercion resistance* . This concept was formalized in [7] by defining an adversary who can demand participants to vote in a specific way. The system was based on a credential mechanism in which a voter could cast more than one ballots under different (fake) credentials, thus making it hard for an adversary to tie a vote to a particular credential (voter's key pair). Each ballot, however, had to go through a series of mixes to ensure coercion-resistance, causing the overhead for the tallying authorities to be quadratic in the number of votes. The Civitas [8] system is an enhancement of this idea.

Another important property of voting systems is *verifiability* which is about ensuring that the outcome of the election reflects correctly the voter choices. In a verifiable system, anyone should be able to verify that submitted ballots are tallied correctly. A popular system Helios [6] is verifiable but it is not receipt-free. On the other hand systems like [7] and its variants that depend on the trustworthiness of the talliers are not really verifiable. Although a basic assumption in all works is that not all talliers can be corrupt, the authors in [1] argue that a verifiable system should not be built on the existence of a trustworthy set of tallying authorities.

The emergence of blockchains introduced a new way to construct systems which have less inherent security vulnerabilities. Blockchains act as distributed databases, where the complete set of data stored in the database is shared among all participants in the network. This concept is the basic element of the Bitcoin[1] protocol, the largest peer-to-peer payment system that was developed as an alternative to conventional, centralized money systems. Blockchains have already been used in voting systems to increase the level of transparency and verifiability offered. However, many of these solutions fail to satisfy important properties expected from secure election systems as explained below.

In [14], a fund transfer system is developed based on Bitcoin transactions. Correct voter behavior is enforced through the use of deposit, however, malicious voters can still forfeit the voting process by refusing to cast their votes. Voting in this scheme is reduced to transferring funds, similar to lottery protocols, but applicability is limited to choosing between two candidates.

A protocol based on the work of [5] and the use of blind signatures is developed in [15]. Anonymity is maintained through the use of PBCs (Prepaid Bitcoin Cards) which are given to voters after registration, however the scheme lacks receipt-freeness. In [16], a blockchain-based voting platform is proposed to conduct national elections. This system relies upon the existence of a trusted third party to hide the user's vote from the election authorities, which negates any of the benefits introduced by the use of blockchains.

Recently, a smart-contract voting system was proposed in [17]. A person's vote is distributed among peers who must tally the votes submitted to them using homomorphic encryption. However, the system does not guarantee fairness as dishonest peers may intentionally "drop" votes. Such behavior is only prevented by incentivizing users to behave correctly, which does not necessarily mitigate malicious behavior. Boardroom voting based on smart contracts was described in [18]. Yet the protocol is not coercion resistant and is

[1] "Bitcoin: A peer-to-peer electronic cash system." http://bitcoin.org/bitcoin.pdf

limited to elections with two options (yes/no). Another recent smart-contract based proposal was given in [19]. It is based on homomorphic encryption and the use of linkable ring signatures to provide a voting framework that is independent from the security and privacy features of the underlying blockchain platform.

The works in [20] and [21] make direct use of the ZeroCoin [22] and ZeroCash [23] protocols developed to anonymize bitcoin transactions. Here one anonymous coin is basically exchanged for one vote. However, while the latter protocols help ensure the anonymity of bitcoin transactions, they cannot be used for voting per se as the security requirements are different. For example, the commitments used to hide the coins in [22] and [23] can help break receipt-freeness in voting schemes when used "as is". Hence the protocols [20] and [21] are not coercion-resistant.

A number of commercial systems ([24, 25]) have also been proposed that use the blockchain as a ballot box. Although these systems claim to achieve verifiability and accessibility, they suffer from other shortcomings. In particular, [24] requires a trusted authority to ensure voter confidentiality and to hide the correspondence between the voter's identity and their voting key. In [25], voter anonymity is protected through the use of a mixing phase, yet no mechanism is anticipated to protect voters from coercion.

In this work we develop a blockchain-based voting protocol that aims to remove dependence on trusted third parties or election authorities in tallying and publishing election results. Having an on-blockchain election system enables direct observation and verifiability of the outcome using available tools and scripts running on the blockchain. Our proposal combines the benefits of using blockchain technologies with a number of cryptographic tools such as Pedersen commitments and *zkSNARKs* to achieve the properties expected from secure elections. Our tools and assumptions are described in the sections that follow.

3. Model

The main entities in our system are the voters $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ which are the entities participating in the election administered by a registration authority $\mathcal{R}$. The role of $\mathcal{R}$ is mainly to authenticate voters, distribute secret credentials and post their public credentials in a blockchain $\mathcal{B}$ set up for voting. Once users are registered they can participate in the election. The blockchain is used to provide universal verifiability. It is an append-only mechanism whose goal is to create a consistent view of the information posted which, once posted, it cannot be removed. Finally, the tallying authority $\mathcal{T}$ is responsible for processing the submitted ballots and publish the final result. The role of $\mathcal{T}$ in our proposal is minimal since its role can be assumed by any interested party, including the voters themselves. The registrar and the tallier(s) can also be part of the same authority which is termed election authority ($\mathcal{EA}$).

3.1. Definition of voting scheme

Voting in our system consists of each voter submitting a single vote v which captures the choices of the election (e.g. whether there is one or many candidates, whether the vote is a simple 'Yes' or 'No', and so on). We abstract away these details under the submission of a vote v whose format is well specified by the election authority $\mathcal{EA}$. Any invalid votes will be rejected during tallying.

The main operations of our proposal are listed below. These enable users to register, vote, verify the tally, and so on. Each voter carries a *token randomizer* (defined in the next section) that helps her with some of these tasks.

- $\mathsf{Setup}(1^\kappa) \rightarrow params$ is a probabilistic algorithm that on input a security parameter κ generates the election authority's public and private keys along with other system parameters.
- $\mathsf{Register}(\mathcal{V}_i, \mathcal{R}) \rightarrow (\mathcal{V}_i, C_i, \sigma_i)$ is a protocol executed between a voter $\mathcal{V}_i$ and the registrar $\mathcal{R}$. The outcome of this protocol is a pair of public/private keys for the voter along with a signed commitment C_i on values s_i, r_i generated by $\mathcal{V}_i$'s token randomizer. The commitment is signed with the voter's private key to produce σ_i. Once this phase is over, the voter is authorized to vote and $(\mathcal{V}_i, C_i, \sigma_i)$ is added to the public list.

- $\mathsf{Vote}(params, \mathcal{V}_i, s_i, r_i, v_i, K_i) \rightarrow b_i$ is executed by $\mathcal{V}_i$'s randomizer to produce ballot b_i. At a high level, $\mathcal{V}_i$ constructs a vote v_i and then encrypts it with a one-time key K_i to produce $E_{K_i}(v)$. She then reveals s_i and produces a proof π_i that she knows the secret number r_i used in the commitment C_i, without however revealing either of r_i or C_i. The ballot $b_i = (\pi_i, s_i, E_{K_i}(v))$ is then sent to the blockchain $\mathcal{B}$ though an *anonymous* channel.

- $\mathsf{Valid}(params, \mathcal{B}, b) \rightarrow \{\bot, \top\}$ checks the validity of a ballot b posted on the blockchain $\mathcal{B}$. It returns $\top$ for valid ballots and $\bot$ for invalid ones (e.g. containing proofs that fail to verify, containing numbers s that have already been posted in the blockchain, and so on).

- $\mathsf{PostKey}(\mathcal{V}_i) \rightarrow \langle s_i, K_i \rangle$ is used by voter $\mathcal{V}_i$ to *anonymously* post her key K_i in the blockchain, while s_i is also sent to facilitate matching the key to her previously transmitted encrypted vote $E_{K_i}(v_i)$.

- $\mathsf{Tally}(\mathcal{T}, \mathcal{B}, \{b_i\}_i, \{K_j\}_j) \rightarrow \tau$ is run by talliers $\mathcal{T}$ to produce the final election result. The talliers apply $\mathsf{Valid}(params, \mathcal{B}, b_i)$ to check the validity of ballots b_i and use the keys to decrypt the votes and account them in the tally τ.

- $\mathsf{VerifyTally}(\mathcal{B}, \{b_i\}_i, \{K_j\}_j, \tau) \rightarrow \{\bot, \top\}$ is used to check the election tally τ. Any interesting party (the public, the voters, etc.) can verify if each ballot carries a correct proof, and if each encrypted vote has been decrypted and incorporated only once in the final tally. If the result equals τ, the function outputs $\top$, otherwise it outputs $\bot$.

Our voting scheme $\mathcal{VS}$ is the collection of the functions defined above and is denoted as $\mathcal{VS} = \{\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{PostKey}, \mathsf{Tally}, \mathsf{VerifyTally}\}$.

3.2. Assumptions

3.2.1. On election authorities and setup

Our voting scheme places minimal trust in election authorities in the sense that they can collaborate with each other to reveal a person's vote. For example a registrar $\mathcal{R}$ can leak to an adversary information revealed during the registration process as an attempt to infer the vote of a user. Alternatively, an adversary may coerce a voter to retain transcripts of the registration process. Similarly, talliers can collaborate with $\mathcal{R}$ to modify or miscount votes. Our protocol is resistant to these adversarial behaviors.

However, we must assume that token randomizers (described in detail below) handed to the users by the election authorities are not malicious and can be trusted. This is similar to the issuance of smart cards to be used in e-voting schemes for secure identification, secure storage and for processing parts of the e-voting scheme such as signing and encrypting messages and/or votes [26]. We thus take it as given that independent verification and analysis can vouch for the correct operation and trustworthiness of the randomizers.

Additionally, the *zkSNARKs* (see Section 4) require a trusted party to generate the common reference string (CRS) for the production and the verification of the proofs. However, a malicious party (e.g. the election authorities) can provide a CRS that allows it to break the ZK property and learn information about the voter's secret parameters. This attack can be prevented if the prover (voter) checks that the CRS is correctly formed. Hence no trust is really placed on the election authorities. Alternatively, the use of Subversion-NIZK [27] ensures that the ZK property is preserved even under maliciously chosen CRS.

3.2.2. On coercion-resistance and token randomizer properties

Coercion can come in many flavors, from simple, the coercer issues some instructions and asks for a proof, to full-on: the coercer is with the voter to make sure that she makes the right voting decisions (e.g. the coercer might be watching through an online video feed). Defending against the latter case is clearly impossible. Our goal is to produce an election scheme that mitigates threats of the former case, while satisfying secure election requirements with as few assumptions as possible.

In this work we introduce the notion of a *token randomizer* ($\mathcal{TR}$), a tamper-resistant device which acts as a black-box on behalf of the user. Receipt-freeness and coercion resistance now will be based on the difficulty of tampering with $\mathcal{TR}$, in particular disclosing randomness produced internally. Randomization

services are not new in the literature of e-voting. They go as back as [28], in which the authors proposed a receipt-free voting scheme based on a trusted, third-party randomizer, and [29, 30, 31] in which the role of the external randomizer is assumed by a tamper-resistant device. Despite the use of such device, however, these voting protocols remain rather involved, requiring the use of threshold cryptosystems and trustful talliers.

Here, our expectations from the $\mathcal{TR}$ are much simpler. Below we list the main assumptions and operations we anticipate from such token randomizer.

- *TRA-1*: $\mathcal{TR}$ is programmed in advance to perform certain cryptographic operations on data submitted to it through pre-defined API calls. Additionally, $\mathcal{TR}$ has its own source of randomness which can be used to generate cryptographic keys or other random values required by $\mathcal{TR}$'s built-in logic. Such randomness is *retained* with the $\mathcal{TR}$'s module and is safely erased once it is not needed any more for subsequent operations. However, the $\mathcal{TR}$ may return such data to higher level programs if this is dictated by its predefined logic. Thus, the user does not have access to $\mathcal{TR}$'s state unless otherwise specified by the algorithms "hardwired" in the $\mathcal{TR}$.

- *TRA-2*: $\mathcal{TR}$ may internally store values which can be returned to the user once a predefined interval has passed. This assumes the existence of an internal counter that can be used to release information at selected intervals only. These intervals are also hardwired in the $\mathcal{TR}$ *before* the election starts and cannot be modified at will.

The first assumption, *TRA-1*, is typical of tamper-resistant devices such as smart cards or TPM-enabled chips. As far as we know, there is also nothing to prevent the feasibility of *TRA-2*. The logic of $\mathcal{TR}$ will be captured by the following operations.

- $\mathcal{TR}.\mathsf{Commit}() \rightarrow C$. The randomizer returns a Pedersen (Section 4) commitment $C = g^s h^r \mod p$ for values s and r chosen at random by $\mathcal{TR}$. Only C is returned to the user while both r and s are stored internally.

- $\mathcal{TR}.\mathsf{Ballot}(v) \rightarrow \{\langle s, E_K(v), \pi\rangle, \perp\}$. The randomizer creates a one-time key K and uses it to encrypt the vote v supplied as input by the user. Embedded in its logic is the proof system of the *zkSNARK* generator (Section 4) which uses only *public* information and the randomness r retained by $\mathcal{TR}$ to compute a zero knowledge proof of knowledge for the value r used in the commitment C. Once π is computed, r is permanently destroyed. The randomizer posts the tuple $\langle s, E_K(v), \pi\rangle$ to the blockchain but stores K internally. Subsequent calls of $\mathcal{TR}.\mathsf{Ballot}(v)$ will return the value $\perp$ as r has been erased. Additionally, premature calls of this function (i.e. before the internal counter reaches the correct value signifying the beginning of the voting period) also return the value $\perp$.

- $\mathcal{TR}.\mathsf{Reveal}() \rightarrow \{\langle s, K\rangle, \perp\}$. Once the internal clock of the randomizer reaches the correct value, signifying the end of the voting period, it posts to the blockchain the value s and the key K used in the encryption $E_K(v)$ of the vote. Then s, K are permanently destroyed. Premature calls of this function return the value $\perp$ as well as subsequent ones.

The intervals used in the $\mathcal{TR}.\mathsf{Ballot}$ and $\mathcal{TR}.\mathsf{Reveal}$ operations are *hardwired* and cannot change at election time. These values have been specified by the election authorities in *advance*. Finally, the randomizer is presented to the user by the registrar $\mathcal{R}$. *It is directly connected to the voting system, it has a limited set of communication interfaces and acts on behalf of the voter during voting* (similar assumptions have been used in [30]). All messages produced by the $\mathcal{TR}$ are digitally signed with its Direct Anonymous Attestation[2]

[2]In the context of vote submission, our goal is to be able to verify that a submitted ballot or key is authenticated but without tying it to a particular randomizer device. For completeness, we briefly summarize how DAA can be adopted in this setting; the interested reader may look at how the scheme has been used for remote, anonymous authentication of Trusted Platform Modules (TPMs) [32]. In our setting the DAA signer is the token randomizer/TPM, a DAA issuer is the manufacturer of the randomizers, while a verifier can be the election authorities or any other external party. At the core of the scheme is

(DAA) signing key. Messages posted to the blockchain must bear the DAA-signature of the $\mathcal{TR}$ to be considered valid.

As mentioned in the beginning of the section, we assume that independent verification and analysis can verify the correct operation of the randomizer.

Remark 1. *There is an obvious attack that can be used to break coercion-resistance of the system; the user may be forced to simply hand-over its token randomizer. In Section 6.1.2, we explain how the basic functionalities of $\mathcal{TR}$ can be extended to defend against this type of attack.*

3.2.3. On side channels and DoS

Finally, we assume that ballots cannot be distinguished based on side channels like time of submission or origin. As the interval between posting a transaction to the P2P blockchain network and seeing the transaction stored in some block may be large, this is a reasonable assumption to expect, especially for large scale elections. Additionally, we ask voters to cast their ballots through *anonymous* communication channels since if the IP address of the voter is visible when she casts a vote then it can be easily linked to the identity submitted during the registration phase (this is a standard assumption in all works where users may directly interact with an adversarial authority). One mechanism that can be used to ensure that a network connection remains anonymous is the anonymity network TOR.

We do not consider Denial-of-Service (DoS) attacks in this work. As the network is open, malicious users may attempt to flood the network with invalid ballots or other data as required by the various phases of the protocol. We are assuming that such erroneous data can easily be filtered. Alternatively, to avoid machine generated posts, users may required to solve a CAPTCHA or, as proposed in [7], solve a cryptographic puzzle before submitting their vote; however this is beyond the scope of this work.

4. Main tools

In what follows we describe the main tools we will be using in our proposal. We assume the existence of two groups $G = \langle g \rangle$ and $\mathbf{G} = \langle \mathbf{g} \rangle$ of prime order $q = O(2^\kappa)$, with κ being the security parameter, that have an efficiently computable bilinear pairing e.

Pedersen commitments

A commitment scheme is a pair of algorithms (Commit, Open) executed between a committer and a receiver. Commit takes a message m and a random number r, and produces a commitment $C = \mathsf{Commit}(m, r)$. In the opening phase, the committer sends (m, r) to the receiver which checks whether the opening algorithm $\mathsf{Open}(C, m, r)$ returns *Accept*. A commitment scheme is secure if it is both binding and hiding. The "hiding" property ensures that the receiver can learn no information about m before the opening phase, while the "binding" property ensures that, once committed, a malicious user cannot find different m' or r' such that $\mathsf{Open}(C, m', r') = \mathit{Accept}$. The Pedersen commitment scheme can be used to commit to a message m by computing $\mathsf{Commit}(m, r) = h^r g^m \mod q$, where g is a generator of a group G of prime order q.

QAPs and zkSNARKs

We will base our constructions to the zero-knowledge Succinct Non-interactive ARguments of Knowledge (*zkSNARKs*) as developed in [34]. Such arguments can be used to prove NP statements about Quadratics Arithmetic Programs (QAPs) without revealing anything about the corresponding witnesses. After taking a QAP Q as input, a one-time setup results in two public keys: an evaluation key EK_Q and a verification

the certification of the DAA key by the issuer, which however learns nothing about the key. Signing with such a key provides anonymity-preserving assurance that the randomizer has a valid DAA key. Yet, neither the verifier nor the issuer, even if they collude with each other, can tell which randomizer signed a message, only that the signature comes from a valid randomizer. In short, the TPM can transform the original credential into new a credential that cannot be linked to original one. The reader may consult [32] for more explanations.

key EV_Q. The evaluation key allows an untrusted prover to produce a proof π regarding the validity of the QAP NP statement.

Informally, a *zkSNARK* for a QAP Q is a triple of (randomized) algorithms (KeyGen, Compute, Verify):

- $\mathsf{KeyGen}(Q, 1^\kappa) \rightarrow (EK_Q, VK_Q)$. On input a security parameter 1^κ and a QAP Q, this function produces a public evaluation key EK_Q and a public verification key VK_Q.
- $\mathsf{Compute}(EK_Q, x, w) \rightarrow \pi_Q$. On input a public evaluation key EK_Q, a $x \in L_Q$, where L_Q is the NP decision language defined by the QAP, and a corresponding witness w, this function produces a proof π_Q that w is a valid witness for x.
- $\mathsf{Verify}(VK_Q, x, \pi_Q) \rightarrow \{0, 1\}$. On input a public verification key VK_Q, x and a proof π_Q, it outputs 1 if $x \in L_Q$ and 0 otherwise.

The properties expected by *zkSNARKs* are completeness, soundness and zero knowledge. The proof returned by algorithm Compute can be turned into a signature scheme by making the message m to be signed part of the challenges exchanged while constructing the proof [36]. We will denote this by the notation

$$\pi_Q \leftarrow zkSNARK[m]((S) : P),$$

where S insides parentheses denotes private information known only to the prover while P constitutes public information available to the verifier.

Let d be the degree of the QAP and let $q = 4d + 4$. In terms of security, the constructions above are secure under the q-power knowledge of exponent (d-PKE), the q-power Diffie-Hellman (q-PDH) and the $2q$-strong Diffie-Hellman ($2q$-SDH) assumptions [34].

Blockchains

We assume limited familiarity with Bitcoin and blockchains, for more information the reader is referred to [37]. A blockchain is a linked-list data structure in which data is organized as blocks, and blocks are connected together through hash pointers (pointer to the hash value of the previous block) to form a chain. Maintaining a hash pointer instead of a simple pointer turns the blockchain into an append-only data structure. As all nodes point to the hash of the previous node, updating a node will result in a chain of updates all the way until the first node in the list. Thus any change to an earlier node can be detected by maintaining the first node's hash value. This property allows the blockchain to maintain its integrity.

The process of extending the blockchain is called mining. Miners compete against each other to extend the blockchain with new blocks. This competition ensures that the network always maintains the largest chain through appropriate consensus mechanisms such as proof-of-work or proof-of-stake. As a result, a nodes' dishonest behavior will be detected and prevented by other nodes.

In this work, we envision the use of the blockchain as a substitute to a public board that should be secure, anonymous and transparent. In the proposed system, the trust to other authorities will be minimized. This way, authorities cannot perform malicious actions because transactions are communicated in public and the append-only character of the blockchain ensures that all transactions are being considered. For more information on blockchains and their use for voting, please see Appendix A.

5. Voting protocol

The protocol works in a P2P, distributed fashion without the need of centralized server. There is a one-time setup phase for the *zkSNARK* algorithms which can be used for multiple elections.

Before we delve into the details of the protocol, we give a rough outline of the main phases of our voting scheme. During an initial registration phase each user is equipped with a token randomizer and public-private key pair which will be used to sign a commitment during the pre-voting period. The randomizer will be responsible for carrying out the main voting actions on behalf of the voter.

During the actual voting period, a registered voter can cast an encrypted vote, authenticating herself using a *zkSNARK* constructed with the previously submitted commitment. Essentially the voter proves that

she knows the secret values bound in *one* of the commitments submitted in the previous phase.Auditing and verification can be done by any interested party once the counting phase is over. The details of each phase are shown in the sections that follow.

Finally, the authorities define a list of intervals to ensure that the election progresses in a timely manner.

- T_{Voting}: This signifies the most important phase of the election in which voters submit their ballots.
- $T_{PostVoting}$: At this point no more ballots will be accepted by the system. When this phase begins, voters submit the keys used to encrypt their votes.
- $T_{Tallying}$: The election is over and tallying can begin.

These intervals must be active for a sufficient amount of time to allow voters to commit and cast their votes. These timers are embedded in the token randomizers as explained below.

5.1. One-time setup

The system is initialized with a call to $\mathsf{KeyGen}(1^\kappa)$, where κ is the security parameter. Once the group parameters $(q, G, \mathrm{G}, g, \mathrm{g}, e)$ are created, the election authority picks a public/private key pair that will be used to sign voters' public keys during registration.

5.2. Pre-Registration and preparations

To participate in the election, users must register first. This typically occurs by presenting valid credentials such as a national ID or passport. Additionally the voter generates a public/private key pair (PK_i, SK_i) and submits her public key to be signed by the registration authority. The authenticated public key will be included in the list of eligible voters along with the user ID.

Each voter is then presented with a token randomizer whose role is to facilitate the rest of the phases of the election. Timer T_{Voting} is associated with operation $\mathcal{TR}.\mathsf{Ballot}()$, preventing voters to have access or cast their ballots before time T_{Voting}. Similarly, $T_{PostVoting}$ is associated with $\mathcal{TR}.\mathsf{Reveal}()$ which is used to post the ballot keys to the blockchain.

5.3. Registration

$\mathsf{Register}(\mathcal{V}_i, \mathcal{R})$ is executed between a voter $\mathcal{V}_i$ and the registrar $\mathcal{R}$. Every pre-registered user who wishes to participate in the election will send to the registrar a commitment $C_i \leftarrow \mathcal{TR}_i.\mathsf{Commit}()$ for a secret value s_i masked with a random value r_i to be used in the actual voting process. The voter does not have access to the values s_i, r_i which constitute *internal* state of her $\mathcal{TR}_i$.

The commitment is then signed by $\mathcal{V}_i$ using her private key SK_i to produce a signature σ_i on C_i. An entry $\langle \mathcal{V}_i, C_i, \sigma_i \rangle$ will be created by $\mathcal{R}$ and posted in the blockchain or in a public database which will hold these triples for all legitimate voters and their commitments. Legitimacy of the signed commitment can be verified by checking whether the associated public key belongs to one of the voters that have been certified. It is clear that submission of the pair (C_i, σ_i) does not have to be anonymous. These values should be verifiable by any entity for the sake of transparency and validity of the voting process.

Notice that while the pre-registration and registration periods have been described as two separate phases, they can also be merged to one if voters already possess valid credentials. These preparations usually take place before the actual voting to give enough time to users to fulfill the necessary election requirements.

5.4. Voting period

A registered voter $\mathcal{V}$ uses $\mathsf{Vote}(params, \mathcal{V}, s, r, v, K)$ to *anonymously* cast a ballot b, authenticating herself as a legitimate voter using a *zkSNARK*. In this phase, each voter reveals the secret number s used in her commitment C and proves knowledge of the secret value r used. The release of s prevents double-voting, since s is stored in the blockchain and cannot be re-used. The whole process is facilitated by the voter's $\mathcal{TR}$. The exacts details are described below.

The *zkSNARK* is constructed by the $\mathcal{TR}$, making use of the pairing system developed in [34, 35] which encodes computations as QAPs. The quadratic program for vote submission is based on the following two observations:

- Checking whether a commitment C belongs to the set of commitments $\{C_0, C_1, \ldots, C_{n-1}\}$ submitted by voters is equivalent to checking whether $\Pi_i(C - C_i) = 0$.
- To prove knowledge of r in the commitment $C = g^s h^r$, one can argue as follows. Let $r = r_{\kappa-1}2^{\kappa-1} + \cdots + 2r_1 + r_0$ be the binary representation of r, with κ being the security parameter of the commitment scheme. Then $C = g^s h^{r_{\kappa-1}2^{\kappa-1}+\cdots+2r_1+r_0} = g^s \Pi_{j=0}^{\kappa-1} h^{r_j 2^j}$. Thus, instead of proving knowledge of r, one can prove knowledge of $h_0, h_1, \ldots, h_{\kappa-1}$, with h_j being equal to 1 or h^{2^j}, depending on the value of bit r_j, such that $C = g^s \Pi_{j=0}^{\kappa-1} h_j$.

Combining the two requirements, we obtain the following witness relation R that can be captured by a QAP:

$$\big((C_0, C_1, \ldots, C_{n-1}, g^s), (h_j)_{j=0}^{\kappa-1}\big) \in R \Leftrightarrow$$
$$\forall j (h_j - 1)(h_j - h^{2^j}) = 0 \wedge \Pi_i (g^s \Pi_{j=0}^{\kappa-1} h_j - C_i) = 0$$

To vote, the user makes a call $\mathcal{TR}.\mathsf{Ballot}(v)$ to its token randomizer, where v captures the user's choice. The randomizer picks a fresh key K to produce $E_K(v)$. It also computes g^s and $h_j = h^{r_j 2^j}$, where $r = \sum_j r_j 2^j$. This is possible since s and r have been stored internally in the $\mathcal{TR}$. Then it feeds these values to the proof algorithm (recall Section 4) which is again part of the $\mathcal{TR}$ to get $\pi = \mathsf{Compute}((C_0, C_1, \ldots, C_{n-1}, g^s, (h_j)_{j=0}^{\kappa-1})$. This proof is essentially a signature for the encrypted vote by including $E_K(v)$ to the random values used by the prover. Thus, $\mathcal{TR}$ creates a ballot which consists of the encryption of the vote $E_K(v)$, the number s and the proof

$$\pi = zkSNARK[E_K(v)]((h_j)_{j=0}^{\kappa-1}) : ((C_i)_{i=0}^{n-1}),$$

where the h_j's are part of the internal state of the $\mathcal{TR}$. Once the proof is created, r and the h_j's are permanently erased. The key K and s, however, are retained until the next phase. Finally, the ballot

$$b = \langle s, E_K(v), \pi \rangle$$

is posted anonymously to the blockchain.

5.5. Post-voting

When the election authorities signal the end of the voting period, no more encrypted votes will be accepted and voters can release the keys used to encrypt their votes. Notice that for some elections it might be ok for the vote to be unencrypted and counting to begin as soon as the votes arrive. In that case the proof π above would just contain the user's vote v in plain form.

However, this would expose the progress of the election while still in the voting phase. In most election systems this would violate the fairness property as early results may influence the decision of late voters. Hence the need for the encrypted vote. Encryption keys are posted to the blockchain through calls to $\mathsf{PostKey}(\mathcal{V}_i)$, which is making use of the voter's randomizer functionality $\mathcal{TR}.\mathsf{Reveal}()$. The pair $\langle s, K \rangle$ is again posted to the blockchain through an anonymous channel.

5.6. Tallying and Verifying

Once the key-submission phase is over, keys will be matched to encrypted votes and counting can begin using $\mathsf{Tally}(\mathcal{T}, \mathcal{B}, \{b_i\}_i, \{K_j\}_j)$. This phase is straightforward as all the information is available in the blockchain and can be carried out by the talliers $\mathcal{T}$ or any other interested party. Hence anybody can audit and verify the tally τ using $\mathsf{VerifyTally}(\mathcal{B}, \{b_i\}_i, \{K_j\}_j, \tau)$.

To facilitate decryption, the talliers may store encrypted votes in a hashtable as soon as they are posted. The key to search the hashtable is the value s contained in the ballot and sent along with the key. Thus, keys and encrypted votes can be matched easily, so decryption takes $O(1)$ per vote, or $O(n)$ overall.

6. Analysis

In this section we analyze the security and efficiency aspects of our proposal, with emphasis on coercion resistance.

6.1. Security

Eligibility: IDs of registered voters are published together with their commitments C_i. When voters obtain their randomizers $\mathcal{TR}_i$, election authorities know that submitted ballots are authorized although they cannot link a ballot to a voter. Therefore only legitimate voters can participate in voting.

Unreusability: Each voter, by means of its randomizer, posts an encrypted ballot which is tied to one of the commitments through the associated *zkSNARK* and the serial s_i used in C_i. Double voting is prevented since no second vote can be submitted with the same s_i. Thus, every voter can only vote once.

Universal Verifiability, Completeness and Soundness: Our protocol provides universal verifiability since both the outcome of the election and the submitted ballots can easily be checked by any other interested party. In particular, one can

- Check that all submitted commitments C_i have been properly signed by certified voter keys.
- Verify whether the *zkSNARKs* in submitted ballots b_i are properly constructed. These proofs essentially show that they have been posted by a legitimate voter, who has submitted a commitment in the previous phase and can be accounted only once (through their serial number s_i).
- Match they keys to encrypted votes using the serials s_i.
- Recount the decrypted votes using $\mathsf{VerifyTally}(\mathcal{B}, \{b_i\}_i, \{K_j\}_j, \tau)$.

Thus, if these checks are performed, one can be sure that all ballots posted on the blockchain are correctly tallied and accounted in the final result. Hence the protocol has completeness. As invalid ballots are discarded and not considered in the tally, the protocol also satisfies the soundness property.

6.1.1. Receipt-freeness and Coercion-resistance:

In this work we are assuming that the attacker cannot constantly monitor the voter. Making a system resistant to this type of attack is impossible if the coercer and the voter are side-by-side during the whole voting period. Note that re-voting does not solve this problem as this assumes that the voter is given a chance to vote again in private. Our work considers attackers that will issue instructions and ask for proof of compliance. Matching these instructions to some form of receipt is the only way for the coercer to check if a voter has complied.

Typically, these instructions may require the voter to use a specific encryption key or randomness that may link the outcome of a cryptographic operation to the coerced vote. In our case, the only places where this can occur is (i) during registration, in the construction of the initial commitment C, and (ii) during voting, in the encryption of the vote with the one-time key K.

The randomness r used in the commitment $C = h^r g^s$ is never revealed by the randomizer and is safely erased once the proof π is constructed during the operation $\mathcal{TR}.\mathsf{Ballot}(v)$. Assuming the tamper-resistance of randomizer $\mathcal{TR}$, the voter cannot obtain any information on $\mathcal{TR}$'s internal state (r and K). As the voter cannot prove any correlation between the submitted ballot (voting phase) and the commitment (registration), no receipt can constructed from the protocol messages. These observations will be captured by the model defined below.

Remark 2. *Note that a coerced voter may try to submit an invalid vote. Although this would reveal the link between the vote and the voter upon decryption, such a vote would not be useful to the coercer. This "attack" can be prevented by making sure that only well-constructed votes are posted by the randomizer during operation* $\mathcal{TR}.\mathsf{Ballot}(v)$.

Remark 3. *A voter may try to sell her vote by bypassing $\mathcal{TR}$ entirely. The voter may generate the commitment herself and submit the* zkSNARK *proof during voting. This would definitely be possible since $\mathcal{TR}$ only uses public algorithms to construct the proof. However, any submission during the voting phase must bear the DAA-signature of a registered $\mathcal{TR}$. The tamper-resistance of the $\mathcal{TR}$ ensures that a voter does not have access to the embedded DAA key.*

To model coercion, we follow the definition of privacy introduced in recent works [12, 33]. These works are mostly used to model protocols where the tally does not produce just a result but also proofs of correct tallying (in our case, tallying is just plain counting of votes). Coercion is formalized as a game in which an adversary $\mathcal{A}$ has to distinguish between two worlds: one in which coercion succeeds and one in which it doesn't. In both worlds, all information submitted is contained in ballot boxes BB_0 and BB_1 that start out empty. Box BB_0 corresponds to the real election (that will be tallied) while BB_1 is a fake ballot box which the adversary has to distinguish from BB_0. In both worlds, the adversary can see the information posted in the ballot boxes BB_β, where $\beta = \{0, 1\}$.

Using this definition our scheme would be deemed insecure since an adversary could easily distinguish between the two cases using the encryption keys posted in the blockchain. Yet these keys have only been used to introduce fairness to the election so that votes are not visible prematurely. They do not present any new information to an observer, so they should not be used to judge privacy. Additionally, using these models, one has to make certain assumptions about the distribution of votes (as in [1, 7]) in order to preclude trivial distinguishing attacks such as when honest nodes vote for candidates a and b but the coerced voter votes for c.

To show that our system is coercion-free, we use the experiment shown in Figure 1 that closely matches the workings of our protocol. In this setting, voters participating in the voting system $\mathcal{VS}$ interact with an adversary $\mathcal{A}$. In particular, two voters $\mathcal{V}_0$, $\mathcal{V}_1$ are involved in the following game with $\mathcal{A}$. After the voters are registered, they are "coerced" to create a ballot $b_\beta = \langle s_\beta, E_{K_\beta}(v), \pi_\beta \rangle \leftarrow \mathcal{TR}_\beta.\mathsf{Ballot}(v)$, for a vote v chosen by $\mathcal{A}$ and $\beta = 0, 1$. The adversary is then given access to one of the ballots and is asked to guess which voter does it correspond to. $\mathcal{A}$ wins the game if its guess is correct.

If the adversary cannot distinguish the two cases with probability significantly more than random guessing, we say that the scheme provides coercion-resistance as per the definition below:

Definition 1 (CR). *Let $\mathcal{VS} = \{\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{PostKey}, \mathsf{Tally}, \mathsf{Verify\text{-}Tally}\}$ be our voting scheme. We say that $\mathcal{VS}$ is coercion-resistant if no PPT adversary $\mathcal{A}$ can distinguish between the games $\mathsf{Exp}^{cr,0}_{\mathcal{A},VS}$ and $\mathsf{Exp}^{cr,1}_{\mathcal{A},VS}$ defined by the experiment in Figure 1; that is, for any PPT adversary $\mathcal{A}$ the following is negligible in the security parameter κ:*

$$\left| \Pr\left[\mathsf{Exp}^{cr,0}_{\mathcal{A},VS} = 1\right] - \Pr\left[\mathsf{Exp}^{cr,1}_{\mathcal{A},VS} = 1\right] \right|.$$

We now show that our voting system is coercion-resistant.

Theorem 1. *Our voting system $\mathcal{VS}$ is coercion-resistant if the commitment scheme is hiding and the QAP proof system is sound and zero knowledge.*

Proof: We prove the theorem by showing that from an adversary $\mathcal{A}$ that attacks coercion-resistance we can construct adversaries $\mathcal{B}$ and $\mathcal{C}$ against the commitment scheme and the QAP proof system such that

$$\mathsf{Adv}^{cr}_{\mathcal{A},\mathcal{VS}} \leq \mathsf{Adv}^{hide}_{\mathcal{B}} + \mathsf{Adv}^{pok-extr}_{\mathsf{POK},\,\mathsf{C}} + \mathsf{Adv}^{ZK}_{\mathsf{POK},C},$$

where $\mathsf{Adv}^{hide}_{\mathcal{B}}$ denotes the hiding advantage of the commitment scheme and $\mathsf{Adv}^{pok-extr}_{\mathsf{POK},\mathcal{C}}$, $\mathsf{Adv}^{ZK}_{\mathsf{POK},\mathcal{C}}$ the extractability and zero-knowledge advantages of the underlying *zkSNARK* system, respectively.

For the setup phase described in Figure 1, coercion-resistance reduces to the Discrete Log (DL) assumption used to prove the hiding property of the commitment scheme as no polynomial time algorithm can distinguish between $h^r g^s \mod q$ and another random quantity $t \in Z_q$. If there is a polynomial time adversary $\mathcal{A}$ that breaks coercion-resistance in our voting scheme then that adversary could be used by an

Experiment $\mathsf{Exp}^{cr,\beta}_{\mathcal{A},VS}$:

Setup phase: The system is initialized with a call to $\mathsf{KeyGen}(1^\kappa)$, where κ is the security parameter. A set $\mathcal{V}$ of n voters is registered into the system. Each voter $\mathcal{V}_i$ generates a key pair (PK_i, SK_i) and is given a randomizer $\mathcal{TR}_i$ which is used to produce a commitment $C_i \leftarrow \mathcal{TR}_i.\mathsf{Commit}()$. Commitments are signed by the voters. The collection of signed commitments $\{\langle C_i, \sigma_i\rangle\}_i$ is given to adversary $\mathcal{A}$.

Challenge phase:

- $\mathcal{A}$ selects two voters $\mathcal{V}_0$ and $\mathcal{V}_1$ and a vote v. $\mathcal{A}$ is given access to a voting oracle.
- $\beta \xleftarrow{R} \{0,1\}$. The oracle gives $\mathcal{A}$ the ballot b_β which is equal to $\langle s_\beta, E_{K_\beta}(v), \pi_\beta\rangle \leftarrow \mathcal{TR}_\beta.\mathsf{Ballot}(v)$.
- $\mathcal{A}$ outputs a guess bit β'.

Exp is successful if $\beta = \beta'$.

Figure 1: Coercion-resistance experiment $\mathsf{Exp}^{cr,\beta}_{\mathcal{A},VS}$.

adversary $\mathcal{B}$ to determine if its input is a Pedersen commitment in polynomial time. This would break the DL assumption of the commitment scheme.

Alternatively, an adversary $\mathcal{A}$ may try to break the soundness property of the underlying *zkSNARK*. $\mathcal{A}$ can try to extract a witness from all verifying proofs of knowledge $\pi_\beta = \mathsf{Compute}((C_0, C_1, \ldots, C_{n-1}, g^{s_\beta}, (h_j)_{j=0}^{\kappa-1})$ communicated during the challenge phase. If $\mathcal{A}$ can do this, then it can guess the bit β used in the experiment of Figure 1. However, the success of $\mathcal{A}$ depends on the outcome of an extractor algorithm E for the QAP system whose advantage is bounded by $\mathsf{Adv}^{pok-extr}_{\mathsf{POK},\mathcal{C}}$.

Finally, $\mathcal{A}$ may try to break the zero knowledge property of the associated *zkSNARK*. A *SNARK* for an NP language L with a corresponding NP relation R is zero-knowledge, if there exists a simulator S that can produce a simulated proof that is indistinguishable from the the real proof π [36]. This property guarantees that the proof reveals nothing more than the correctness of the statement. Thus, an adversary $\mathcal{A}$ that breaks coercion-resistance in our voting scheme implies an adversary $\mathcal{C}$ that breaks the zero-knowledge property of the NIZK proof system with advantage $\mathsf{Adv}^{ZK}_{\mathsf{POK},\mathcal{C}}$.

Adversary $\mathcal{C}$ works as follows. It runs adversary $\mathcal{A}$ as a subroutine by simulating the security experiment exactly as in Figure 1. When $\mathcal{C}$ has to decide between a simulated proof π_s which is provided by the simulator S and a real proof π as implied by the NIZK security experiment, it constructs the following ballots for voters $\mathcal{V}_1$ and $\mathcal{V}_2$. It sets $\pi_0 = \mathsf{Compute}((C_0, C_1, \ldots, C_{n-1}, g^{s_0}, (h_j)_{j=0}^{\kappa-1})$, $\pi_1 = \pi_s$ and constructs the corresponding ballots b_0, b_1. Then it picks a random bit β and gives A the ballot b_β. If $\mathcal{A}$ can distinguish between the two cases, then $\mathcal{B}$ breaks the zero-knowledge property of the underlying QAP system.

This is not possible, however, as was shown in [36]. Hence under the q-PDH and d-PKE assumptions, our voting scheme instantiated with the QAP defined in Section 5.4 is sound and zero knowledge. □

6.1.2. Other real world attacks

We close this section by summarizing a number of real-world attacks our protocol defends against.

In a *randomized ballot attack*, an adversary coerces a voter to submit a randomly composed ballot. While both the attacker and the voter have no idea about which candidate this voter casts the ballot for, the purpose of this attack is to nullify the ballots submitted by the voter. This attack does not work in our system since the voter cannot bypass the $\mathcal{TR}$ nor submit an invalid ballot as explained in Remarks 2 and 3 above.

In a *ballot buying attack*, the adversary asks for proof that the voter complied and voted as instructed. This is prevented in our system since the randomness r used to produce the commitments during registration and the proofs in the voting phase is never revealed. Thus nothing can link the ballot to the identity of the voter.

In a *credential leaking attack*, an attacker can force the voter to give away her secret credentials. The purpose of this attack is either to allow the coercer vote on behalf of the user, or find out what the user voted for. This attack is also prevented by our system since the actual ballot posted by the user is not linked to her private key used to sign her commitment in the registration phase. A more applicable attack is for the coercer to ask for the voter's token randomizer. We treat this attack below.

What if attacker asks for the user's token randomizer? A coercion-free system is one in which a voter deceives an adversary into believing she behaved as instructed, even if she was asked to reveal her secret keys or vote in a particular way. In our case, the analogue of giving away secret keys would be to hand-over the token randomizer. To defend against this hand-over attack, we suggest a simple modification to $\mathcal{TR}$'s functionality. An implicit assumption here is that coercion takes place remotely, i.e. the adversary does not continuously watch over the shoulder of a voter, monitor her hard-drive, etc. Hence the voter has a moment of privacy (see also [7] for a similar assumption).

In particular, we let the voter specify a vote v that will be *locked* in the $\mathcal{TR}$'s state. Then during the voting stage, even if the user (or the coercer) specifies a different vote v', operation $\mathcal{TR}.\mathsf{Ballot}(v')$ will result in posting the ballot $\langle s, E_K(v), \pi \rangle$ to the blockchain. This ballot remains unlinkable and the coercer cannot tell what the $\mathcal{TR}$ posted to the blockchain.

Locking a vote v can occur any time after registration. The user interacts with the $\mathcal{TR}$ as if she was ready to vote. However, if the $\mathcal{TR}$'s internal counter signifies that the voting stage has not yet begun, the value v is stored internally and becomes a *locked vote*. This locked vote is then released when voting is allowed. As the $\mathcal{TR}$ already has the capability of internally storing secret values (recall operational assumption *TRA-2*), this functionality can easily be supported.

6.2. Efficiency aspects

One important aspect of our system is that tallying has only a linear overhead with respect to number of voters n (as a comparison the system in [7] requires $O(n^2)$ work). This is due to the fact that the encrypted votes in ballots and the keys to open them bear the same identifier s (the serial used in the initial commitment). Hence tallying and producing the final result is a very efficient procedure.

Additionally, the characteristics of the proof system for arithmetic circuits developed in [34] translate directly to our voting system. Nonetheless, significant performance gains can be achieved by maintaining the voter commitments in a Merkle tree instead of an explicit list L_{com} (the idea was used in the Zerocash implementation [23] for anonymous Bitcoin payments). Doing so, reduces the time and space complexity from linear in the size of L_{com} to just logarithmic. Additionally, the NP statement used in the QAP construction of knowing the value r of one of the commitments C_i maintained in L_{com} now becomes "*I know r such that C_i appears as a leaf in a Merkle tree with root r*." This modification increases exponentially the number of users (and their commitments) that can be supported by the system.

The above, together with the state-of-the-art implementation of *zkSNARKs* for arithmetic circuits results in a prover (the token randomizer's) running time of a few minutes and a proof verification to just a few milliseconds. In particular, the findings in [23] show that a proof can be generated in under 3 minutes in a single threaded 2.7GHz CPU. Verifying a proof takes less than 9ms, while proof size is restricted to 288 bytes.

The above numbers suggest the practicality of our system and its use in large-scale elections. Our system does not require any heavy computational power on behalf of the talliers since the only important operation required to produce the final tally is verifying proofs in submitted ballots. Hence our system remains verifiable in a strong sense as anyone can check the validity of the election outcome.

7. Conclusions

As existing blockchain voting systems cannot provide the comprehensive list of security features needed for secure elections, we have proposed a blockchain-based voting scheme that ensures voter privacy with minimal involvement on the user side. Our protocol uses the blockchain as an infrastructure for universally verifiable elections along with a number of cryptographic primitives that can be used to achieve all basic

properties expected from secure election systems. Our protocol achieves coercion resistance (and hence receipt-freeness) by means of a randomizer token whose purpose is to act as a black-box on behalf of the user. As no complicated operations like re-voting, managing fake credentials, or lying to coercer are expected by the user, the protocol achieves usability, which is another important requirement of e-voting systems. Additionally, the protocol requires no trust on election authorities; even if the adversary corrupts tallying authorities, ballots cannot be forged. Finally, the protocol is scalable and has linear tallying overhead which makes it practical for large-scale elections.

References

[1] X. Yi and E. Okamoto. "Practical internet voting system." Journal of Network and Computer Applications 36, no. 1, pp. 378–387, 2013.

[2] J. Benaloh. "Simple verifiable elections", in Proceedings of th eelectronic voting technology workshop (EVT06), 2006.

[3] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. "Scantegrity: End-to-end voter-verifiable optical-scan voting." IEEE Security & Privacy 6, no. 3, 2008.

[4] T. Moran and M. Naor. "Split-ballot voting: everlasting privacy with distributed trust." ACM TISSEC 13, no. 2 (2010): 16.

[5] A. Fujioka, T. Okamoto, and K. Ohta. "A practical secret voting scheme for large scale elections." In International Workshop on the Theory and Application of Cryptographic Techniques, pp. 244–251. Springer, Berlin, Heidelberg, 1992.

[6] B. Adida. "Helios: Web-based Open-Audit Voting." In USENIX security symposium, vol. 17, pp. 335–348, 2008.

[7] A. Juels, D. Catalano, and M. Jakobsson. "Coercion-resistant electronic elections." In ACM workshop on Privacy in the electronic society, pp. 61–70. ACM, 2005.

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

[9] Peter YA Ryan, Peter B. Ronne, and Vincenzo Iovino. "Selene: Voting with transparent verifiability and coercion-mitigation." In International Conference on Financial Cryptography and Data Security, pp. 176-192, 2016.

[10] G. V. Post. "Using re-voting to reduce the threat of coercion in elections," Electronic Government, an International Journal 7 (2) (2010) 168-182.

[11] G. S. Grewal, M. D. Ryan, S. Bursuc, and P. YA Ryan. "Caveat coercitor: Coercion-evidence in electronic voting." IEEE Symposium on Security and Privacy, 2013.

[12] V. Cortier, G. Fuchsbauer, and D. Galindo. "BeleniosRF: A Strongly Receipt-Free Electronic Voting Scheme." IACR Cryptology ePrint Archive 2015 (2015): 629.

[13] Josh Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections." In 26th annual ACM Symposium on Theory of Computing, pp. 544–553, 1994.

[14] Z. Zhao, and T-H.H. Chan. "How to vote privately using bitcoin." In International Conference on Information and Communications Security, pp. 82–96, 2015.

[15] Jason Paul Cruz, and Yuichi Kaji. "E-voting System Based on the Bitcoin Protocol and Blind Signatures." In IPSJ Transactions on Mathematical Modeling and its Applications, Vol. 10 No. 1 14-22, 2017.

[16] K. Lee, J. James, T. Ejeta, and H. Kim, "Electronic Voting Service Using Block-Chain," J. Digit. Forensics Secur. Law, 2016.

[17] Wenbin Zhang, Yuan Yuan, Yanyan Hu, Shaohua Huang, Shengjiao Cao, Anuj Chopra, and Sheng Huang. "A Privacy-Preserving Voting Protocol on Blockchain." In 11th IEEE Inter. Conference on Cloud Computing, pp. 401–408, 2018.

[18] Patric McCorry, Siamak F. Shahandashti, and Feng Hao. "A smart contract for boardroom voting with maximum voter privacy." In International Conference on Financial Cryptography and Data Security, pp. 357–375, 2017.

[19] Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. "Platform-independent secure blockchain-based voting system." In International Conference on Information Security, pp. 369–386, 2018.

[20] Yu Takabatake, Daisuke Kotani, and Yasuo Okabe. "An anonymous distributed electronic voting system using Zerocoin." 2016.

[21] Pavel Tarasov and Hitesh Tewari. "Internet Voting Using Zcash." 2017.

[22] I. Miers, C. Garman, M. Green, and A. D. Rubin. "Zerocoin: Anonymous distributed e-cash from bitcoin." IEEE Symposium on Security and Privacy, 2013.

[23] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin." In 2014 IEEE Symposium on Security and Privacy, 2014.

[24] Follow my vote. Available at https://followmyvote.com/

[25] Tivi voting. Available at https://tivi.io/

[26] J. Budurushi, S. Neumann, M. Volkamer. "Smart cards in electronic voting: lessons learned from applications in legally-binding elections and approaches proposed in scientific papers." In 5th International Conference on Electronic Voting 2012 (EVOTE2012).

[27] M. Bellare, G. Fuchsbauer, and A. Scafuro. "NIZKs with an untrusted CRS: security in the face of parameter subversion." In ASIACRYPT 2016.

[28] Martin Hirt and Kazue Sako. "Efficient receipt-free voting based on homomorphic encryption." In International Conference on the Theory and Applications of Cryptographic Techniques, pp. 539–556. Springer, Berlin, Heidelberg, 2000.

[29] Emmanouil Magkos, Mike Burmester, and Vassilis Chrissikopoulos. "Receipt-freeness in large-scale elections without untappable channels." In *Towards The E-Society*, pp. 683–693. Springer, Boston, MA, 2001.
[30] B. Lee and K. Kim. "Receipt-free electronic voting scheme with a tamper-resistant randomizer." In International Conference on Information Security and Cryptology, pp. 389–406. Springer, Berlin, Heidelberg, 2002.
[31] G. S. Grewal, M. D. Ryan, L. Chen, and M. R. Clarkson. "Du-vote: Remote electronic voting with untrusted computers." In the 28th IEEE Computer Security Foundations Symposium (CSF), pp. 155–169, 2015.
[32] E. Brickell, J. Camenisch, and L. Chen. "Direct anonymous attestation." In 11th ACM CCS, pp. 132–145, 2004.
[33] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. "A comprehensive analysis of game-based ballot privacy definitions." In IEEE Security and Privacy 2015.
[34] B. Parno, J. Howell, C. Gentry, and M. Raykova. "Pinocchio: Nearly practical verifiable computation." IEEE Symposium on Security and Privacy, 2013.
[35] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. "Pinocchio coin: building zerocoin from a succinct pairing-based proof system." In ACM workshop on Language support for privacy-enhancing technologies, 2013.
[36] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. "Quadratic span programs and succinct NIZKs without PCPs." In International Conference on the Theory and Applications of Cryptographic Techniques, 2013.
[37] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.
[38] J. Benaloh. "Rethinking voter coercion: The realities imposed by technology." The USENIX Journal of Election Technology and Systems 82 (2013).
[39] Y. Nasser, C. Okoye, J. Clark, and P. YA Ryan. "Blockchains and Voting: Somewhere between hype and a panacea."
[40] Vitalik Buterin. "On Public and Private Blockchains". Ethereum Blog. Available at: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/
[41] United Nations. "Election Observation". Available at: http://www.un.org/womenwatch/osagi/wps/publication/ Chapter7.htm

Appendix A – On the use of Blockchains for Voting

All voting systems assume the existence of a bulletin board (BB), an append-only broadcast channel where ballots or other data get posted, without however specifying how this can be implemented. In real life, the bulletin board would have the form of website maintained by the election authorities, where data appear for public use, after being replicated to various storage servers to protect against accidental failures.

In an end-to-end verifiable voting system, the main role of the bulletin board is to ensure the fairness and correctness of the election process. To meet these objectives, bulletin boards must have a number of desirable properties, such as:

- They must be distributed to withstand DOS attacks and accidental failures.
- They must support the chronological logging of events.
- They must be integrity protected to avoid manipulation attacks such as adding, removing or altering posted data.
- They should be verifiable, i.e. that information posted is authentic.

In real life, however, bulletin boards seldom satisfy these properties; election authorities control the bulletin board so they can add/remove information or even censor messages to be published. Consider, for example, the extreme case where authorities want to know what Alice voted for. They can simply replace all other ballots in the bulletin board by encryptions of valid votes and then during tallying the preference of Alice will be revealed. What this "attack" demonstrates is that you cannot expect to have full privacy when bulletin boards are controlled by election authorities. Essentially, all protocols based on the use of BBs pre-suppose that election authorities can be trusted.

The answer to "whether a blockchain is really necessary" is not clear. In principle, a bulletin board can be used instead considering the trust assumptions mentioned previously. But currently, the blockchain is the best implementation we have for a bulletin board [39]. It is widely distributed, it is maintained by a collection of entities (not necessarily affiliated with election authorities) which are incentivized to include all posted transactions, in a highly consistent manner.

Use of a public blockchain comes with a number of disadvantages too, the most important one being that every transaction requires a fee. So, who is going to pay for this? The election authorities may give each

user a small amount but nothing prevents the user from taking the money and spent it elsewhere. Even if users are equipped during registration with blindly signed tokens that can be exchanged with ballots, such transactions still require a small fee. Another disadvantage is throughput. A public blockchain like the one used by the Bitcoin network processes one transaction every few seconds, so it would take many days to handle the load generated by a large scale election.

A solution to the above might be the use of *permissioned* blockchains. A blockchain of this kind is controlled by a group of nodes that determine who can transact on the blockchain and who can serve the network by extending the chain with new blocks. That blockchain will not require a fee if a transaction is posted by a legitimate voter. Another variant, a *consortium* blockchain might be closer to the needs of a democratic election. It is a "partially decentralized" blockchain [40], where the consensus process is controlled by a pre-selected set of nodes. Consider a consortium of 10 local and international institutions, in which 6 must sign every block in order for the block to be added to the blockchain. The right to read the blockchain can be public or restricted to the participants. These blockchains would process transactions at a greater rate, thus serving the needs of real-life elections. They would also be more resistant to spam and DoS attacks as they could easily filter transactions. They would, however, re-introduce trust in the authorities running these blockchain, hence the need for impartial observers.[3]

[3]Election observation is a tool already used by the United Nations [41] when domestic observer organizations do not have the strength or resources to organize impartial elections. Observation protects the civil and political rights of participants and can help build confidence in the honesty of the electoral processes. One can envision a similar service to be used when restricted blockchains are used to run large scale elections. These observers may be running mining nodes, requiring their agreement and participation to extend the blockchain with new valid blocks.