The ISC Int'l Journal of Information Security
ISeCure

November 2022, Volume 14, Number 3 (pp. 123–133)
http://www.isecure-journal.org

Presented at the ISCISC'2022 in Rasht, Iran.

An Electronic Voting Scheme Based on Blockchain Technology and Zero-Knowledge Proofs [☆]

Sepehr Damavandi [1], and Sadegh Dorri Nogoorani [1,*]
[1] *Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran.*

ARTICLE INFO.

Keywords:
Applied Cryptography, Blockchain Voting, Blockchain Privacy, Electronic Voting, ZK-SNARKs

Type:
Research Article

doi:
`10.22042/isecure.2022.14.3.13`

dor:
`20.1001.1.20082045.2022.14.3.13.7`

ABSTRACT

Voting is a fundamental mechanism used by many human societies, organizations and nations to make collective decisions. There has been a tremendous effort on making this mechanism fairer, error-free and secure. Electronic voting aims to be a solution to some deficiencies of existing paper-based voting systems. While there have been excellent technical and practical advances in e-voting, and some of them were great in defining the needs and musts of an ideal voting system, there are also severe critics of existing solutions mostly related to end-to-end verifiability and software independence. In this paper, we use blockchain and zero-knowledge proofs for a secure e-voting scheme that satisfies these requirements while preserving the privacy of the voters. We also evaluate our scheme from security and performance aspects.

1 Introduction

Failures in voting systems has caused great damage to societies and governments throughout history. A voting scheme should be able to defend itself. If someone can question its validity and it cannot prove its outcome, then the protesters (right or wrong) have a justified reason to protest. Traditional paper-based voting systems are hard to audit and inefficient when it comes to contestability and being able to prove themselves. Their proof is often based on the same trust assumptions that are being questioned. Traditional voting systems can be made more convenient in security and trust by using methods such as live streaming security cameras in every polling station and absolute open audibility by people and journalists. But that would be expensive. Digitization has been a trend in many traditional physical aspects of our life, and the Internet and cryptography have performed well even in finance which has strict security and privacy requirements. In another dimension, we try to decentralize power and trust, and blockchain technology has been a successful option for decentralization.

Electronic voting has its history of evolution with great advents and also hideous failures [1, 2], but it has been keeping up with all these technical advances, using the privileges they offer to further its practice and even came up with new cryptographic advances which are used in other different applications. Blockchain technology is used in some e-voting systems to bring about decentralization and transparency while being publicly verifiable and not manipulatable. Recently there has been another revolution

* Corresponding author.

☆ The ISCISC'2022 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: `sepehrdamavandi@modares.ac.ir`, `dorri@modares.ac.ir`

in this ecosystem that deals with privacy and scalability, i.e., zero-knowledge proofs which are succinct and practical (ZK-SNARKs) [3]. We use a combination of blockchain technology and ZK-SNARKs to further the position of e-voting towards decentralized and liberating voting systems by offering absolute non-likability.

An evidence-based election that can prove its outcome must have five necessary but insufficient requirements, namely ballot secrecy, software independence, voter-verifiable ballots, contestability and auditing [1]. In addition to that, there are some other important requirements such as eligibility of voters and coercion-resistance that makes electronic voting systems more challenging. One important factor in providing these requirements is the assumptions that they are based on.

Electronic, and more specifically, blockchain voting systems have been argued to be non-functional at the time being for political elections compared to paper-ballot-based voting systems [1]. Nonetheless, these critics do not solve the security and trust assumptions of traditional paper-based elections in which people need to trust some agents that are hired for the election and are susceptible to being corrupted. Governments usually put armed soldiers and warranty the safety of the voters, but that really neither proves the integrity of ballots nor prevents righteously or unsatisfied voters to question the outcome of the vote. The authority that is responsible for running an election or for providing user credentials is considered trusted by the majority of traditional and electronic voting schemes. However, we use blockchain and zero-knowledge proofs to minimize and distribute these trust assumptions and make these authorities responsible for later audits.

Decentralization of authority and power has made a great leap with the advent of blockchain technology [4] which does not need to trust centralized authorities to provide integrity and soundness. This decentralization can harm privacy because everything needs to be clear and verifiable. For example, Bitcoin can be de-anonymized by using the information in its blockchain [5]. Zerocash [6] was the first digital coin that provided privacy for its users using the ZK-SNARK technology. ZK-SNARKs are a fast-growing technology that targets privacy and scalability issues on Blockchains and provides capabilities that are needed in a decentralized structure. They are a new fast kind of zero-knowledge proofs for NP-complete languages that make us able to prove some statements to be true without revealing any more information besides the fact that the statement is true. For example, one can prove that he/she knows a specific pre-image to a one-way function and nothing will be revealed about his/her private inputs. This proof can also be publicly verifiable. In addition, the prover/verifier can do their proof/verification in a fast and efficient way, which makes ZK-SNARKs practical to be leveraged in an e-voting system. For a comprehensive review of the ZK-SNARK technology refer to [3], and to the *AZK GitHub repository* [1] for tools and community reach.

We use blockchain to improve transparency, decentralization, public verifiability, and audibility. In addition, we use ZK-SNARKs to provide maximum privacy for voters without losing scalability or compromising the soundness of the system. Our contributions are:

(1) Proposing a core e-voting scheme that is end-to-end verifiable, completely private, and non-linkable even in the face of corrupted authorities
(2) A modular registration and authentication scheme that ensures voter eligibility and practical authentication using blockchain

The remainder of this paper is organized as follows. Section 2 discusses the related works done in areas related to this paper. In Section 3 we provide the general description of our scheme and figures to make it more comprehensible. Section 4 is where we discuss the security requirements of a voting system and discuss how we achieve them and under what assumptions. In Section 5 we compare our proposal with the most related works. Finally, we conclude our paper in Section 6.

2 Related Works

In this section, we review and classify the related works. The first paper on encrypted e-voting schemes was proposed by Chaum in 1981 [7], although a great topic to start, this scheme had several disadvantages and if only one voter had a failure, the election had to be restarted. Proposals continued their growth and innovations for better security, cryptographic techniques and addressing vulnerabilities also came a long way since.

The development of electronic voting systems can be divided into seven historical phases [8]. The 1990 decade is when e-voting matures and Internet voting is born. The first e-voting proposal for large-scale elections was made by [9] which has been implemented in real-world application scenarios such as Census of Washington University [10]. The hype for e-voting and remote voting continued until 2004 when vulnerabilities of such systems came to atten-

[1] https://github.com/matter-labs/awesome-zero-knowledge-proofs

tion and their failures came to the surface. Then, came the phase for systems guidelines and recommendations in which new concepts and requirements such as software-independence [11], end-to-end verifiability [12], receipt-freeness [13] and coercion resistance [14] were defined and struggles to fulfill them continues till now.

Bitcoin's [15] emergence in 2009 highlighted the power of blockchain technology in decentralizing trust and power in financial applications. Subsequently, the idea of using blockchain technology to enhance electronic voting systems became relevant and proposals for using them appeared. In particular, the blockchain's transparency and traceability can reduce e-voting concerns [16]. There are multiple proposals for blockchain voting systems emphasizing different features-fairness [17], Sybil resistance [18], platform independence [19] and coercion-resistance [20], to name a few. E-voting requirements are advert and sometimes contradictory (e.g., privacy vs. verifiability[21]). Our proposal focuses on privacy and the stronger notion of non-linkability (i.e., there is no way to link a vote to its voter even in the face of corrupted authorities), and at the same time, satisfying other requirements including the contradictory requirements of verifiability and transparency.

The first practical implementation of ZK-SNARKs in *common reference string model* (CRS) was made available by [22] for usage in verifying cloud computations. After that, Zerocash [6] used an enhanced version of them to make a decentralized anonymous payment scheme. The techniques used in this system are also used by us to make a ballot mix. E-voting proposals using blockchain and ZK-SNARKs include [23] in which the Zcash [2] implementation of the Zerocash protocol is used to anonymously transfer vote coins to candidates. In this scheme, candidates should all participate in tallying the votes and if one of them refuses to do so, the election is falsified. In addition, this approach is inferior to performance.

In [19], a mixture of Homomorphic encryption, mix-nets, zero-knowledge proofs, blind signatures and linkable ring signatures is used. Blockchain used by this scheme specifically uses the PBFT consensus protocol to achieve high performance. There is a smart contract administrator which can deploy/terminate the smart contracts used for voting. The voting administrator triggers the tallying and results in the publishing phases and also authenticates users. Voters register with the voting administrator and get their public key verified on the blockchain. In the voting phase, users encrypt their vote with the public key of the ballot and send their vote along with a ZKP stating their vote integrity. A smart contract is deployed here to check these proofs and also prevent double voting. The final sum of the encrypted votes is verifiable by the public when the voting administrator publishes the decryption key of the ballot.

The authors in [20] focus on coercion-resistance and fairness of their system by using time-release and receiver-deniable encryption. Each voter gets two distinct credentials that are not distinguishable by possible coercers. One of these credentials is false and the voter can provide the coercer with that one. Votes that are cast using these false credentials are discarded in the tallying phase. Although the system is evaluated for coercion-resistance by the authors, the case in which a coercer demands the two credentials from a voter and submits his/her desired vote with both of them has not been considered. The option of re-voting is also provided, and only the latest vote up to the election deadline is counted by the system.

In another work [24], the ZK-SNARKs are used in an e-voting scheme implemented by Hawk [25]. Hawk is a SNARK-based smart contract system that maintains the privacy of its users. The authors in [24] enhance the capabilities of Hawk to achieve better performance and remove its trusted manager. This proposal is not yet complete and fundamentally different from ours.

3 Proposed Scheme

Our proposed scheme consists of the following entities:

- An *authority*, A, that is responsible for running the election,
- A *registration authority*, RA, which provides credentials and identity information for eligible voters,
- A set of *registration warranters*, RW, which warranty the information and possible credentials that are provided by RA; this set can be an empty set. RWs can be telecommunication companies if we are using SMS as a method of authentication.
- A set of *authentication warranters*, AW, that provide safe authentication procedures for the election,
- A *consortium* which consists of candidates and A itself plus any other participants deemed necessary. The consortium is responsible for generating the key pair of the ballot box, generating the proof that verifies the election outcome, participating in the consensus protocol of the blockchain, and
- A *consortium blockchain* with public read access.

[2] https://z.cash

The *voters* are mentioned by V_i. The scheme has several phases which we discuss in the following subsections. We will investigate the security parameters of the scheme in the next section. Figure 1 provides a general visual description of the scheme.

Our protocol is inspired by the Zerocash protocol in the laundering function which sounds natural to use in an e-voting system but is different in other functionalities such as its proof sentences, its circuits and transactions between peers of the network. By the laundering function, we mean the process in which ZEC tokens are poured into a pool of tokens and then spent by users in an anonymous and non-linkable manner.

3.1 The Election Smart Contract

The *election smart contract* (denoted by SC) is used in different parts of the scheme and the interactions of different parties with the blockchain are through the SC. We briefly review these interactions here:

- In the setup phase: the SC initiates the election by the Authority giving its needed inputs, it is also responsible for verifying credentials by involved entities.
- In the authentication phase: the SC is responsible for processing the coming requests from the voters and their corresponding results from the authorities, and also it verifies public keys based on these results.
- In the ballot preparation phase: the SC processes the coming requests from the voters who are authenticated in the previous phase and makes sure they put only one commitment in the list.
- In the voting phase: the SC checks the validity of proofs sent with each ballot and collects them.

We will provide details about these functionalities in the description of each phase.

3.2 Setup Phase

The zero-knowledge proofs that we use become non-interactive in the common reference string model [3]. Therefore, we need all the participants other than the voters to take part in a *multi-party computation* (MPC) protocol [26]. The MPC protocol is run by the election authority, A, and generates two keys; namely, the *validation key* and *proving key*. These keys are used in our ZK-SNARKs [27], and as long as one of the participants is honest, the proof system is secure. We call these public parameters of the system, *pp*.

Then, each participant generates a public key pair for blockchain authentication. The consortium starts the blockchain with each of the members running a node and registering and publishing the public keys of the participants other than the voters in the blockchain. We will later explain the voters' keys in the *Authentication Phase*.

After that, A deploys the election smart contract (SC) on the chain and publishes the election details including the list of the candidates and the correct vote values. As we use the multiplicative version of ElGamal [28] in our scheme (additively homomorphic), A also publishes its required public parameters in the smart contract on chain. In particular, A chooses a large prime p, a subgroup of prime order q from Z_p and a generator g of this subgroup, and submits this information to the smart contract (SC).

In this step, the consortium members collectively generate the public key of the ballot box, PK_{bb}, as follows. Each of them chooses a random integer α_j as their private key and $\beta_j = g^{\alpha_j} (mod\ p)$ as their public key. They submit this public key to the election smart contract. The contract multiplies them all together and generates the final public key of the ballot box, the zero-knowledge common reference string, the validation key and the proving key.

With that being done, RA and RWs (if present) come into play to register the list of eligible voters (see Figure 2).

In particular, the RA provides the list of eligible voters to A and RWs in a table that we call RR. The RA also commits to each row of this table, which we call the election RollUp. The commitments are simple SHA256 hashes that are published on-chain:

$$RollUp_i = SHA256(RR_i).$$

where the ith entry of RR is denoted by RR_i, and stored in a hashmap by the election smart contract (SC). RR_i is composed of the necessary information about voter V_i which is at least his/her *name*, *national code* (or SSN), *birth date* (for age), and *postal code* (for local elections). After that, RWs verify the validity of the information in each entry of RR and submit their confirmation in a transaction to the election smart contract. If any inconsistencies are detected, the RA shall correct RR until the RollUp is confirmed by all RWs. This process will provide the system with voter eligibility and makes vote stuffing very hard even if A and RA collude.

3.3 Authentication Phase

In this phase, we authenticate the voters and make sure that they are who they claim to be (see Figure 3). Voter V_i who wants to vote chooses a key pair (PK_i, PR_i) calculates the $RollUp_i$ according to his/her information, and a random 256-bit identifier which we call $Request_i$, and submits them to the elec-

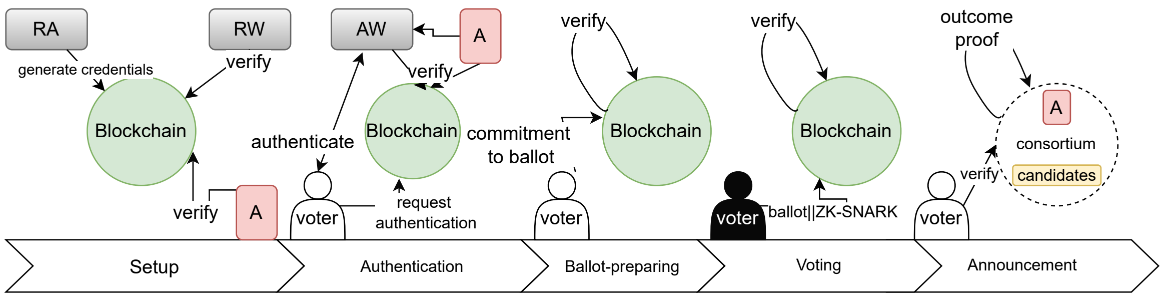

Figure 1. A visual description of our scheme (The voter is anonymous in the voting phase, so he/she is colored black)

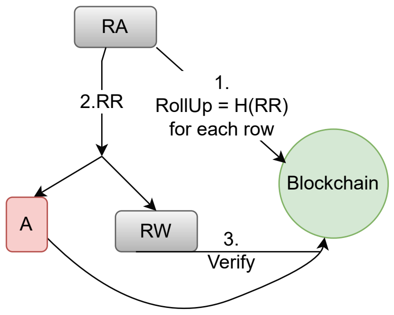

Figure 2. The authority and the registration entities verify and commit to the list of eligible voters

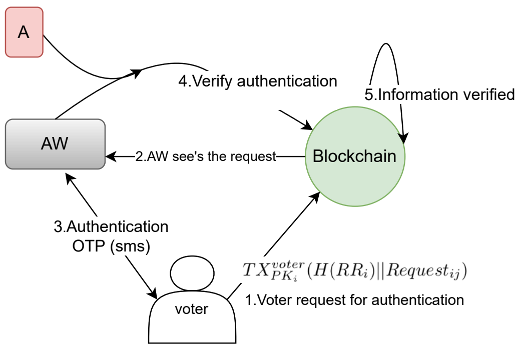
A
4.Verify authentication
5.Information verified
AW
2.AW see's the request
Blockchain
3.Authentication OTP (sms)
$TX^{voter}_{PK_i}(H(RR_i)||Request_{ij})$
1.Voter request for authentication
voter

Figure 3. Upon receiving a request from a voter, AW will perform authentication and verify its success or failure with the authority's supervision

tion smart contract (SC) in a transaction signed by PR_i:

$$V_i \rightarrow SC : voter(PK_i, H(RR_i), Request_i), \sigma_i \quad (1)$$

By $\dot{voter()}$ we mean the *voter* method of the SC is called in a transaction, and σ_i is the voter's signature on the transaction. The transaction states that he/she wants to vote. Upon receipt of each *voter* transaction by the smart contract, A requests an authentication service with AW's or they can directly process coming requests. In either case, the authentication between V_i and AW will be performed, and if it is successful, AW will confirm $Request_i$ with public key PK_i corresponding to $RollUp_i$ on the chain and hence we say PK_i has got a right to vote (VTZ).

3.4 Ballot Preparation Phase

Voter V_i who is authenticated in the previous phase and has a VTZ can now use that VTZ to get a ballot. In this stage V_i chooses a random ballot serial (sn_i), and calculates a statistically hiding commitment scheme COMM over sn_i:

$$cm_i = COMM_{r_i}(sn_i) = \mathcal{H}(r_i||sn_i) \quad (2)$$

in which r_i is the commitment trapdoor (another random number). We use a SHA256 compression function for COMM, which maps a 512-bit input to a 256-bit output. We call this function $\mathcal{H}(.)$. Using $\mathcal{H}$ simplifies the construction of the circuit for the ZK-SNARK used in the voting phase (C^{vote}).

Then, V_i sends the commitment in a transaction signed by PR_i to the election smart contract (SC):

$$V_i \rightarrow SC : issue(PK_i, cm_i), \sigma_i \quad (3)$$

By $\dot{issue()}$ we mean the *issue* method of the SC is called in a transaction, and σ_i is the voter's signature on the transaction. The smart contract verifies that the public key requesting to add its commitment has a right to vote (VTZ) and that it does this once. Then, puts cm_i in a Merkle tree (CMTree) over the list of commitments (CMList). Doing this reduces the time and space complexity of the proof validation algorithm from linear to logarithmic. We use the collision-resistance function $\mathcal{H}(.)$ to avoid an explicit representation of CMList by maintaining an append-only Merkle tree over the growing list CMList. By using a Merkle tree of depth 32 we can support over 400 million votes.

3.5 Voting Phase

In this phase, each voter uses his/her ballot to cast his/her vote anonymously. To do so, the voter V_i chooses a secret key pair (SPK_i, SPR_i). Then, he/she sends his/her vote to the election smart contract (SC) in a transaction signed by SPR_i:

$$V_i \rightarrow SC : vote(sn_i, c_i, \Pi^{vote}_i), \sigma_i \quad (4)$$

By $\dot{vote()}$ we mean the *vote* method of the SC is called in a transaction, and σ_i is the voter's signature using the anonymous private key. Each candidate has a rank in the candidate list which is announced by A in the setup phase. We denote this rank by j $(1 \leq j \leq n)$. We reserve the first bit of the voter's ballot to announce abstaining from the election, and the next bits are for candidates in the same order announced by A. Each candidate choice must be encrypted separately by the voter, and the concatenation of these choices will make the final ballot. For example, if a voter's choice is candidate three, and we have four candidates, his/her vote will be: $enc(0)||enc(0)||enc(0)||enc(1)||enc(0)$. The transaction should be sent over an anonymous network like TOR, and consists of the following items:

- The ballot serial (sn_i)
- The vote choice (c_i) which is the votes (m_i^j) encrypted by the public key of the ballot-box (PK_{bb}) with randomness t_i^j and concatenated together in order defined by A:

$$c_i = c_i^0||c_i^1||\ldots||c_i^n \tag{5}$$

  where

$$c_i^j = enc_{PK_{bb}}(m_i^j||t_i^j) \tag{6}$$

- A ZK-SNARK (Π_i^{vote}) for the following statements: (I) the ballot is fine, that is, it includes a valid vote, and (II) I know an r_i such that $COMM_{r_i}(sn_i)$ is a leaf of the CMTree over CMList.

The ballot serial, sn_i which has not been revealed in the last phase, is used here as a nullifier, assuring that each voter can only submit one vote to the chain. Therefore, the smart contract checks that sn_i is not used before to cast a vote, and verifies Π_i^{vote}. If these checks are fine, c_i is stored in the ballot box.

Note that V_i uses an anonymous channel, a secret key pair, and neither reveals r_i nor the particular commitment cm_i which was added to the tree in the last phase. Therefore, no one can identify him/her.

Our ZK-SNARK will be defined for the arithmetic circuit satisfiability problem, and we should define and build this circuit and its public parameters pp accordingly. Such circuits have some publicly known inputs and some private inputs called the *witness*. The witness is composed of the secrets kept with the prover (i.e., the voter). In that regard the public inputs of the C^{vote} circuit will be:

- Public parameters (pp) of the circuit,
- The Merkle tree (CMTree) over the list of commitments and its root (rt),
- The public key of the ballot box (PK_{bb}),
- The ballot serial (sn_i) provided by the voter in the *vote* transaction, and
- Voter's ballot, c_i provided by the voter with the *vote* transaction.

And the witness is:

- The plain text of the voter's choice to check its validity (m_i^j),
- The commitment (cm_i) and its trap door (r_i),
- A valid authentication path for cm_i in CMTree, and
- The randomness (t_i^j) used to encrypt each m_i^j using ElGamal cryptosystem.

The circuit checks the following items:

- The voter's commitment is valid, i.e., cm_i, r_i, and sn_i match Equation 2;
- The authentication path is correct with regard to cm_i, CMTree, and rt;
- The voter's choices (m_i^j) are valid—i.e., only one or zero, and the total sum of the choices is equal to one; and
- Checks that the vote is valid, i.e., c_i, t_i^j, PK_{bb}, and m_i^j match Equation 5.

3.6 Announcement and Auditing

After the voting phase ends, the consortium runs another round of MPC in a private setting and reveals their secret shares of the ballot box private key to A. Then, A aggregates all the encrypted votes by homomorphic summation and publishes the outcome along with a ZK-SNARK to prove it. This adds another layer of security against coercion. There should be a period considered for public auditing before finalizing the outcome. At this time, everyone can use the publicly readable transactions of the blockchain, calculate the aggregate vote, and verify the proof to ensure that everything is okay.

The MPC used to retrieve the private key of the Ballot box can be retrieved by a smart contract and then used by it to construct the proof. Projects are working on efficient private delegation of ZK-SNARK provers [3] which can be used to construct this proof without actually constructing the private key of the ballot box. But in its most simple form, A can retrieve the private shares of consortium members after the election and construct the proof itself. The public inputs of this ZK-SNARK will be:

- The final aggregated votes in encrypted form $(C_i \ (0 \leq i \leq n))$,
- The final result of the election (M) claimed by A, and
- The public key of the ballot box (PK_{bb}).

And the witness is:

- The private key of the ballot box (PR_{bb}).

The circuit verifies that the following items are correct:

[3] See https://youtu.be/iT_s92f3wds for example.

- The public and private keys of the ballot box match, and
- The result is correct. That is:

$$M_i = dec_{PR_{bb}}(C_i). \tag{7}$$

4 Analysis

The core goal in our scheme is to provide an *evidence-based election*. The principle of *evidence-based elections* is that election officials should not only find the true winner(s) of an election but also provide convincing evidence about that [29]. We discuss our case against five minimal goals/requirements for an evidence-based election and then discuss other requirements from [30].

Ballot secrecy: Ballot secrecy is the privacy of votes during and after the election. Our scheme is completely private and this essential requirement is fulfilled by using ZK-SNARKs at two stages. In the voting phase, we remove the relation between a voter and his/her vote which makes the votes private and not linkable even against corrupted authorities. In the announcement stage, we use a ZK-SNARK to prove the outcome and do not reveal the private key of the ballot box. Since voters cast their vote with a completely anonymous public key and this public key nor the ballot of the voter cannot be traced or marked by any entity, this property is achieved.

Software independence: By software independence, it is meant that the work of any software-based piece of the system (including the auditing components) is verifiable and the system should produce an evidence trail with an associated verification procedure to check that the system (i) has recorded votes as intended, (ii) has collected them as recorded, and (iii) has counted them as collected, in any given execution [1]. Therefore, software independence does not imply refraining from using the software at all and provides for *end-to-end (E2E) verifiability*. Our scheme fulfills the (ii) and (iii) parts of these requirements by using blockchain and ZK-SNARKs. Verified and authenticated voters sign their transactions with their public key, so no one else can cast a vote on their behalf, and since every transaction on the chain is publicly visible everyone can confirm that the summation of encrypted votes is announced soundly and that each vote is indeed submitted by a verified voter. Ensuring the first part requires voter and public participation. Since the details of our system are known, the software that the voters use to generate their votes and proofs (i.e., the front end) can be open-source software. In addition, the voters can also use any other software that they trust and verify its correctness as they like.

Voter-verifiable ballots: This is similar to condition (i) of E2E verifiability above, so we argue the same. Voters are responsible for making sure that their prepared ballot reflects their intended choice. Yes, this is more complicated in electronic voting systems compared to traditional paper-based ballots; but democracy and civil rights always come with a price, voters should be able to participate in ensuring the soundness of the system. Any real-world implementation of a remote voting system should be open-sourced so that different possible vendors can offer their software to their user base.

Contestability: This refers to the extent to which the system can support its users when certain types of error occur. Whichever of the three main components of the information security model, namely confidentiality, integrity and availability is harmed, the voters should be able to convince the others, and there should be remedies and strategies in place to provide support in that case. The transparency of blockchain technology along with our registration and authentication architecture which is performed on the temper-proof and future auditable chain regards great attention to this property. Other e-voting schemes which do not rely on a blockchain, commonly have servers installed by authorities that communicate with voters and other entities to perform different functionalities. Transactions interchanged between these two parties are not visible to anyone else so in case some voter states that the server maliciously did not accept my transaction, there is no proof to either verify or dismiss this claim. Furthermore using this architecture makes the final result announced publicly verifiable. Historically this assurance of the final result is the most challenging aspect of elections.

Eligibility: In traditional paper-based voting systems and the majority of the proposed and implemented electronic voting systems, this property is held by the assumption of a trustable registration authority and election supervisor. This opens up the possibility for vote stuffing and credential misuse by them. Our two sets of warranters provide distribution of this trust assumption. They are also responsible for future audits of their participation.

Registration warranters (RWs) can be organizations and institutions which are already provided with the data they are verifying. The *National Organization for Civil Registration* in Iran is responsible for issuing identity certificates at birth time. Many organizations such as banks and telecommunication companies already have this information and can take part as private-sector participants in this set. Usually, the *executive* branch of a government is responsible for running elections which can be considered to be the authority (A) in our scheme. The other two *leg-*

islative and *judicial* branches of the government can be the other candidates to be in the RW set.

Receipt-freeness: Roughly stated, receipt-freeness is the inability of a voter to prove to an attacker that he/she has voted in a particular manner, even if the voter wishes to do so [14]. This property doesn't let any interaction between the voter and the adversary during the election. Since the final private key of the ballot box is never revealed and is protected by a ZK-SNARK proving the outcome, voters are not able to prove how they have voted even if they reveal all the credentials used by them during the election.

Coercion resistance: Coercion resistance is stronger than receipt-freeness. An adversary can interact with the voters during the election process, and a scheme is coercion-resistant if it is infeasible for the adversary to determine whether a coerced voter complies with the demands. There are some schemes like JCJ [14] and Civitas [31] which achieve coercion resistance by some non-trivial assumptions. For example, Civitas assumes that there is an untappable channel between the voters and the registrars which cannot be achieved even with cryptography and require physical registration. Furthermore, privacy in these schemes is dependent on the trustability of the registration and the tallying authorities, and there should be at least one stage (usually the registration stage) in which the adversary cannot control the voters. We do achieve some level of coercion resistance with simpler assumptions and complications and argue that in comparison, it is simpler to just assume that the voters maintain their integrity to vote. We will discuss this matter further in the discussions of *Trust Assumption 3.*

There are some schemes like JCJ [14] and Civitas [31] which achieve coercion resistance by some non-trivial assumptions. For example, Civitas assumes that there is an untappable channel between the voters and the registrars which cannot be achieved even with cryptography and require physical registration. Furthermore, privacy in these schemes is dependent on the trustability of the registration and the tallying authorities, and there should be at least one stage (usually the registration stage) in which the adversary cannot control the voters. We do achieve some level of coercion resistance with simpler assumptions and complications and argue that in comparison, it is simpler to just assume that the voters maintain their integrity to vote.

Performance: The Zerocash protocol, and its implementation Zcash, have a bigger and more complicated circuit and network structure than we do. Their *pour* transaction which is comparable to our *vote* transaction has a proof time of two minutes and 55 seconds on an Intel Core i7-2620M @2.70GHz with 12GB of RAM and on one thread. The verification time on the same system is 8.5 milliseconds and the proof size is 288 bytes [6]. Considering an average system similar to the mentioned setup to be used by the voters, we can estimate that our proving time can hardly be more than three minutes [32]. The voters can perform the computations on their own time and this does not affect the block generation time. Our scheme is blockchain-independent and can be implemented using any kind of consortium blockchain as long as it can run smart contracts. So efficiency and performance are mainly tied to the underlying consensus algorithm used by the chain.

In addition to the mentioned goals/requirements, we have several simplifying assumptions. We will discuss their feasibility and provide some insights into when they are neglected.

Trust assumption 1. *Voters trust the computer they are using to vote.* This is to ensure the end-to-end verifiability of the system. There has been little research about the open problem of a *secure platform* for electronic voting, and the software, the firmware and even the hardware can be corrupted. The voters can use different software and devices to make sure about the final ballot they are preparing.

Trust Assumption 2. *The Internet is available to the voters.* This ensures the availability of our system. If the Internet is deliberately made unavailable in an area of targeted users, that prevents them from participating in the election. Nevertheless, the authority cannot use the credentials of these voters to impersonate them because we authenticate each voter by RWs.

Trust Assumption 3. *The voters have a private time and space to authenticate themselves, calculate and send their commitments, calculate the ZK-SNARK, and submit their votes.* This assumption is analogous to a physical private space for filling a paper vote, and protects the voters from being vulnerable to coercion in the following obvious cases:

- *Coercion during the authentication phase:* This results in the coercer's key pair getting the victim's right to vote.
- *Coercion during the ballot-preparation phase:* In this case, the attacker misuses the victim's credentials to calculate and submit his/her commitment.
- *Coercion during the voting phase:* In this case, the attacker misuses the victim's commitment and trap door (cm_i, r_i) to submit a vote.

Trust Assumption 4. *If the Authority is not* *trusted, at least one of the Authentication Warranters is trustful.* In other words, we do not expect a case in which the Authority and all the Authentication Warranters are malicious. The collection of public and private sector representatives in our scheme minimizes this risk. Telecommunication companies do this authentication for many applications such as e-commerce, social networking, email services, etc. It is practical to use such a service that is available and fast.

For elections that are more sensitive and risky, we should use more secure ways of authentication. Other options include: using a smart card to store private keys, providing other hardware-enforced security options, or using image processing tools and techniques along them to further enhance the assurance.

Trust Assumption 5. *The consortium will not generate the private key of the ballot box before the voting phase ends, and will not reveal it thereafter.* The first part of this assumption is related to the *fairness* of the system and the second part is required to have *future receipt-freeness.* While there are options such as *time-release encryption [20]* to technically guarantee the first part of this assumption, that comes with some performance penalty. The second part of this assumption is to ensure receipt-freeness of the scheme. Since in receipt-freeness there is no interaction between the adversary and voters, if this part of the assumption is falsified, voters can, later on, deny their knowledge of credentials used during the election.

5 Comparison with the Related Works

We compare the properties of our scheme with the four most related blockchain-based voting schemes in Table 1. Two of these schemes, namely [23] and [24], use ZK-SNARKs in their construction, specifically, the Zerocash protocol. In [23] ZEC tokens are distributed to the voters and they transfer these tokens to their candidate of choice. In this case, the voters may decide to keep their tokens as they have financial value in the network. In addition, if a candidate does not cooperate in the tallying phase, the election is falsified. The authors in [24] try to overcome the latter issue by using a customized form of Hawk protocol which doesn't need a trustable manager for privacy-preserving smart contract execution. This smart contract is used to tally the votes and provides for better performance in proof confirmation. Neither [23] nor [24] provide a precise evaluation of their performance. However, they are using the Zerocash protocol with its complicated circuit and different kinds of addresses and security mechanisms which are not needed for e-voting. Our scheme is more efficient in this regard as our proofs are more fine-tuned. In addition, we use a special-purpose consortium blockchain which is also more efficient than Zerocash (based on the Bitcoin blockchain).

We already discussed the first five properties in Table 1. The non-linkability property is another property that does not hold when ballot secrecy (the privacy of the voters) is based on the trustworthiness of some election authorities. In this case, these authorities can discover the vote of each voter. Our proposal along with the other schemes that use the Zerocash protocol are not-linkable thanks to the ZK-SNARKs. However, the two Zerocash-based proposals [23, 24] do not implement end-to-end verifiability, contestability, eligibility and receipt-freeness. Our scheme has five of the six desired properties and does not provide for coercion resistance. Coercion resistance is somehow contradictory to non-linkability since we need to mark the ballots to be able to distinguish the ones cast with false credentials, and these marks are later recognizable by the entities that issued them for each specific voter. Furthermore, schemes that have coercion-resistance have other non-realistic assumptions such as untappable channels [14], physical registration phase [31], or limitations on the behavior of adversaries [20].

6 Conclusions and Future Work

We proposed a non-linkable voting scheme with the necessary security requirements satisfied under acceptable assumptions. Scalability and practicality are reserved by using fast SNARK proofs. Electronic voting systems can benefit both from the blockchain and ZK-SNARKs technology, and although system security concerns remain an issue, other financial and less critical applications fail when it comes to security and theft. Nevertheless, by comparing to traditional systems, electronic voting systems are superior and more future-proof. In future work, we aim to implement our scheme and optimize its performance. Furthermore, we aim to extend our scheme with re-authentication and re-voting mechanisms to make our scheme more resistant to coercion.

References

[1] Sunoo Park, Michael Specter, Neha Narula, and Ronald L Rivest. Going from bad to worse: from internet voting to blockchain voting. *Journal of Cybersecurity*, 7(1):tyaa025, 2021.

[2] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J Alex Halderman. Attacking the washington, dc internet voting system. In *International Conference on Financial Cryptography and Data Security*, pages 114–128. Springer, 2012.

Table 1. Comparison of our scheme with the most related works

Property	[23]	[19]	[20]	[24]	Our Scheme
Ballot Secrecy	Yes	Trusted VA	Trusted RA	Yes	Yes
Contestability	No	No	No	No	Yes
Eligibility	No	No	Trusted RA	No	Distributed trust
Receipt-freeness	No	Yes	Yes	No	Yes
Coercion-resistance	No	No	Conditional	No	No
Non-linkability	Yes	No	No	Yes	Yes

[3] Justin Thaler. Proofs, arguments, and zero-knowledge, 2022.

[4] Ruhi Taş and Ömer Özgür Tanrıöver. A systematic review of challenges and opportunities of blockchain for e-voting. *Symmetry*, 12(8):1328, 2020.

[5] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[6] 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*, pages 459–474. IEEE, 2014.

[7] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[8] M Mesbahuddin Sarker and Tajim Md Niamat Ullah Akhund. The roadmap to the electronic voting system development: a literature review. *International Journal of Advanced Engineering, Management and Science*, 2(5):239465, 2016.

[9] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.

[10] Lorrie Faith Cranor and Ron K Cytron. Sensus: A security-conscious electronic polling system for the internet. In *Proceedings of the Thirtieth Hawaii International Conference on system sciences*, volume 3, pages 561–570. IEEE, 1997.

[11] Ronald L Rivest and Madars Virza. Software independence revisited. In *Real-World Electronic Voting*, pages 19–34. Auerbach Publications, 2016.

[12] Josh Benaloh, Ronald Rivest, Peter YA Ryan, Philip Stark, Vanessa Teague, and Poorvi Vora. End-to-end verifiability. *arXiv preprint arXiv:1504.03778*, 2015.

[13] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553, 1994.

[14] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections*, pages 37–63. Springer, 2010.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[16] Patricia Baudier, Galina Kondrateva, Chantal Ammi, and Eric Seulliet. Peace engineering: The contribution of blockchain systems to the e-voting process. *Technological Forecasting and Social Change*, 162:120397, 2021.

[17] Sebastian Müller, Andreas Penzkofer, Darcy Camargo, and Olivia Saa. On fairness in voting consensus protocols. In *Intelligent Computing*, pages 927–939. Springer, 2021.

[18] Yuxi Cai, Georgios Fragkos, Eirini Eleni Tsiropoulou, and Andreas Veneris. A truth-inducing sybil resistant decentralized blockchain oracle. In *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 128–135. IEEE, 2020.

[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*, pages 369–386. Springer, 2018.

[20] Kaili Ye, Dong Zheng, Rui Guo, Jiayu He, Yushuang Chen, and Xiaoling Tao. A coercion-resistant e-voting system based on blockchain technology. *Int. J. Netw. Secur*, 23:791–806, 2021.

[21] Alisa Pankova and Jan Willemson. Relations between privacy, verifiability, accountability and coercion-resistance in voting protocols. In *Applied Cryptography and Network Security*, pages

313–333, 2022.

[22] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 626–645. Springer, 2013.

[23] Pavel Tarasov and Hitesh Tewari. Internet voting using zcash. *Cryptology ePrint Archive*, 2017.

[24] Aritra Banerjee. A fully anonymous e-voting protocol employing universal zk-snarks and smart contracts. In *International Congress on Blockchain and Applications*, pages 349–354. Springer, 2021.

[25] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[26] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *Cryptology ePrint Archive*, 2017.

[27] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 781–796, 2014.

[28] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *International Conference on Cryptology in India*, pages 61–72. Springer, 2004.

[29] Justin Parkhurst. *The politics of evidence: from evidence-based policy to the good governance of evidence*. Taylor & Francis, 2017.

[30] Alexander Schneider, Christian Meter, and Philipp Hagemeister. Survey on remote electronic voting. *arXiv preprint arXiv:1702.02798*, 2017.

[31] Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368. IEEE, 2008.

[32] Kashif Mehboob Khan, Junaid Arshad, and Muhammad Mubashir Khan. Investigating performance constraints for blockchain based secure e-voting system. *Future Generation Computer Systems*, 105:13–26, 2020.

Sepehr Damavandi is a member of the Blockchain Laboratory at the Electrical and Computer Engineering Faculty of Tarbiat Modares University. He holds a Bachelor's degree in Computer Engineering from IAU (Tabriz branch). His research interests include Web3 and cutting-edge cryptographic components for preserving privacy like ZK-SNARKs that help in advancing blockchain technology and Web3.

Sadegh Dorri Nogoorani is an assistant professor and the head of the Blockchain Laboratory at the Electrical and Computer Engineering Faculty of Tarbiat Modares University. He holds an M.Sc. in Computer Networks and a Ph.D. in Computer Engineering from the Sharif University of Technology. His research interests are blockchain and distributed ledger technology, security, privacy, trust, and distributed systems design in general.