Platform-Independent Secure Blockchain-Based Voting System

Bin Yu[1], Joseph K. Liu[1(✉)], Amin Sakzad[1], Surya Nepal[2], Ron Steinfeld[1], Paul Rimba[2], and Man Ho Au[3]

[1] Monash University, Melbourne, Australia
joseph.liu@monash.edu
[2] CSIRO, Sydney, Australia
[3] The Hong Kong Polytechnic University, Kowloon, Hong Kong

Abstract. Cryptographic techniques are employed to ensure the security of voting systems in order to increase its wide adoption. However, in such electronic voting systems, the public bulletin board that is hosted by the third party for publishing and auditing the voting results should be trusted by all participants. Recently a number of blockchain-based solutions have been proposed to address this issue. However, these systems are impractical to use due to the limitations on the voter and candidate numbers supported, and their security framework, which highly depends on the underlying blockchain protocol and suffers from potential attacks (e.g., force-abstention attacks). To deal with two aforementioned issues, we propose a practical platform-independent secure and verifiable voting system that can be deployed on any blockchain that supports an execution of a smart contract. Verifiability is inherently provided by the underlying blockchain platform, whereas cryptographic techniques like Paillier encryption, proof-of-knowledge, and linkable ring signature are employed to provide a framework for system security and user-privacy that are independent from the security and privacy features of the blockchain platform. We analyse the correctness and coercion-resistance of our proposed voting system. We employ Hyperledger Fabric to deploy our voting system and analyse the performance of our deployed scheme numerically.

Keywords: Evoting · Blockchain · Ring signature
Homomorphic encryption

1 Introduction

Voting plays a significant role in a democratic society. Almost every local authority allots a significant amount of budget on providing a more robust and trustworthy voting system. Cryptographic techniques like homomorphic encryption and Mix-net [7] are usually applied in contemporary electronic voting systems to achieve the voting result verifiability while preserving voters' secrecy. However, incidents like a security flaw that has erased 197 votes from the computer

L. Chen et al. (Eds.): ISC 2018, LNCS 11060, pp. 369–386, 2018.
https://doi.org/10.1007/978-3-319-99136-8_20

database in the 2008 United States elections [34] and the compromise of 66,000 electronic votes in the 2015 New South Wales (NSW) state election in Australia [30] raise the public concerns on the security of electronic voting systems. For voting systems based on bulletin (e.g., [1,9]), one of the major concerns is whether the voting result that is published on the bulletin can be trusted. Blockchain with the growing popularity and remarkable success in cryptocurrency provides a new paradigm to achieve the public verifiability in such electronic voting systems.

Recently a number of blockchain-based electronic voting systems have been developed by exploiting its inherent features. These systems can be classified into three broad categories. (1) Cryptocurrency based voting systems (e.g., [21,33,43]). The ballots to a candidate are based on the payment he/she receives from the voters; the problem with such systems are malicious voters may refuse to "pay" the candidates to retain the money. Furthermore, a centralised trusted party, who coordinates the payment between the candidates and voters must exist. (2) Smart contract based voting system [28], which only supports two candidates and the voting is restricted to limited number of participants. Furthermore, it requires all voters to cast their ballots before reaching an agreement on the voting result. (3) Using blockchain as a ballot box to maintain the integrity of the ballots [11,35].

In summary, the security of these blockchain-based systems highly depends on the specific cryptocurrency protocol they employed. Additionally, these voting systems can only work with a specific blockchain platform, and support a limited number of candidates and voters. Based on our observations and studies, we believe that any blockchain-based voting systems should have the following three features: (1) Platform-independence — this means the changes in the underlying blockchain protocols should not affect the voting schemes. (2) Security framework — the voting system should be implemented with comprehensive security features (the detail of security features are discussed in Sect. 5). The nature of the blockchain allows everyone to obtain the data on it; thus, the comprehensive security features have critical importance to ensure that the ballots are fully secured on the blockchain. (3) Practical — it should be scalable, which means the large amount of ballots casting and tallying can be finished in a reasonable time.

Our Contributions: In this paper, we propose an electronic voting (evoting) system that supports the above identified three features as follows.

1. *Our voting system does not depend on a centralised trusted party for ballots tallying and result publishing.* Compared with traditional voting systems, which highly depend on a centralised trusted party to tally the ballots and publish the result, our voting system takes the advantage of a blockchain protocol to eliminate the need for a centralised trusted party. The details of the blockchain trustworthiness and voting system trust assumptions are discussed in Sects. 4 and 5, respectively.
2. *Our voting system is platform-independent and provides comprehensive security assurances.* Existing blockchain-based voting systems highly depend on the underlying cryptocurrency protocols. Receipt-freeness [31] and correctness

of the voting result are hard to achieve (we analyse the blockchain-based voting system explicitly in Sect. 2). The security of our voting system is achieved by cryptographic techniques provided by our voting protocol itself, thus, our voting system can be deployed on any blockchain that supports smart contract. To achieve the goal of providing a comprehensive security, we employ Paillier system to enable ballots to be counted without leaking candidature information in the ballots. Proof-of-knowledge is employed to convince the voting system that the ballot casted by a voter is valid without revealing the content of the ballot. Linkable ring signature is employed to ensure that the ballot is from one of the valid voters, while no one can trace the owner of the ballot. The detail of security features that we achieved are discussed in Sect. 5.

3. *Our voting system is scalable and applicable.* In order to support voting scalability, we propose two optimised short linkable ring signature key accumulation algorithms which are described in our full paper [41] to achieve a reasonable latency in large scale voting. We evaluate our system performance with 1 million voters to show the feasibility of running a large scale voting with the comprehensive security requirements.

The rest of the paper is organised as follows: we discuss the cryptographic techniques applied in voting systems and analyse some typical voting systems in Sect. 2. Cryptographic primitives and our voting protocol are presented in Sects. 3 and 4, respectively. We analyse the correctness and security of our voting system in Sect. 5. In Sect. 6, we deploy our voting system and analyse its performance.

2 Related Work

Secure evoting is considered as one of the most difficult problems in security literature as it involves many security requirements. To satisfy these security requirements, cryptographic techniques are mostly applied in constructing a secure evoting system. In this section, we discuss the existing voting systems based on traditional public bulletin and blockchain technology.

2.1 Public Bulletin Based Voting Systems:

In the following, we outline the key cryptographic techniques used in public bulletin based voting systems and how such techniques address the corresponding security requirements.

– **Homomorphic encryption:** Homomorphism feature allows one to operate on ciphertexts without decrypting them [13]. For a voting system, this property allows the encrypted ballots to be counted by any third party without leaking any information in the ballot [10,14,18]. Typical cryptosystems applied in a voting system are Paillier encryption [32,40] and ElGamal encryption [19,21].

- **Mix-net:** Mix-net was proposed in 1981 by Chaum [7]. The main idea of mix-net is to perform a re-encryption over a set of ciphertexts and shuffle the order of those ciphertexts. Mix node only knows the node that it immediately received the message from and the immediate destination to send the shuffled messages to. The voting systems proposed in [1,17,20] apply mix-net to shuffle the ballots from different voters, thus the authority cannot relate a ballot to a voter. For the mix-net based voting systems, they need enough amount of mix nodes and ballots to be mixed.
- **Zero-knowledge proof:** Zero-knowledge proof is often employed in a voting system [9,29,39] to let the prover to prove that the statement is indeed what it claimed without revealing any additional knowledge of the statement itself. In a voting system, the voter should convince the authority that his ballot is valid by proving that the ballot includes only one legitimate candidate without revealing the candidate information.
- **Blind signature and linkable ring signature:** Voting systems like [12,16, 22] employ blind signature [12] to convince the tallying centre that the ballot is from a valid voter while not revealing the owner of the ballot. Simultaneously, the authority who signs the ballot learns nothing about the voter's selections. In blind signature, both voters and tallying centre must trust the signer. If the signer is compromised, the signature scheme may stop working. Unlike blind signature, linkable ring signature [3–5,8,23–27,36,42] is proposed to avoid the untrusted signer. Instead, it needs a certain number of voters to participate in the signing process. By comparing the linkability tag, the authority can easily tell whether this voter has already voted. When the voter signs on the ballot, he/she needs to include other voters' public keys to make his/her signature indistinguishable from other voters' signatures.

2.2 Blockchain-Based Voting Systems

The blockchain-based voting systems can be discussed under three broad categories as follows.

- **Voting systems using cryptocurrency:** In [43], authors propose a voting system based on Bitcoin. In their voting system, the ballot does not need to be encrypted/decrypted as they employ the protocol for lottery. Random numbers are used to hide the ballot that are distributed via zero-knowledge proof. Making deposit before voting may keep the voters to comply with the voting protocol while the malicious voters can still forfeit the voting by refusing to vote. However, only supporting "yes/no" voting may restrict the adoption of this voting system.

  In [33], authors proposed a voting system on the Zcash payment protocol [15] without altering the inner working of Zcash protocol. The voter's anonymity is ensured by the Zcash address schemes. The correctness of the voting is guaranteed by the trusted third-party and the candidates. In this system, the authority, who manages the Zcash and voter status database should be

trusted. If the authority is compromised, double-voting or tracing the source of the ballot is possible.

- **Voting systems using smart contract:** In [28], the authors claim that their open voting network is the first implementation of a decentralised and self-tallying Internet voting protocol with maximum voter privacy using Blockchain. They employ smart contract as a public bulletin to achieve self-tallying.[1] However, their voting system can only work with two candidates voting (yes/no voting) and the limitation of 50 voters makes it impractical for a real large scale voting system.
- **Voting systems using blockchain as a ballot box:** Tivi and Followmyvote [11,35] are commercial voting systems which employ the blockchain as a ballot box[2]. They claim to achieve verifiability and accessibility anytime anywhere, while the voters' privacy protection in these systems is hard to evaluate.

To summarize, most traditional voting systems need a centralised trusted party to coordinate the whole voting process. In these systems, the centralised trusted party plays a critical role in storing the ballots, counting the ballots, and publishing the voting result. Although the existing blockchain-based voting systems take advantage of blockchain public verifiability, the system security and user privacy of these systems depend on the features provided by the underlying blockchain platform. Our proposed approach not only takes the benefits of a decentralised trust offered by the blockchain technology to remove the need of a centralised trusted party to do the ballots tallying, voting result decoding and publishing, but also considers key security primitives proposed in the literature including traditional evoting systems to build a practical platform independent secure evoting protocol that can be deployed to any blockchain platforms that support smart contract.

3 Cryptographic Primitives

In this section, we describe the cryptography primitives borrowed from traditional voting systems and apply in our evoting system. Note that the syntaxes, correctness conditions, and security models of a linkable ring signature and a public key encryption are given in Appendix A and Appendix B, respectively in our full paper [41].

3.1 Message Encode and Decode

Before the voting starts, we must encode the candidate ID to make it suitable for vote tallying. The encode/decode functions are defined as follows:

[1] Self-tallying means that after the casting phase, voters can count the ballots themselves.

[2] The authors call the storage of the ballot as the ballot box.

- **Candidate encoding:** We define $\zeta := \mathsf{Encode}(m) \in \mathbb{Z}$ as the candidate encoding function. For ρ candidates, each labeled with an ID from $\mathcal{P} = \{1, 2, \ldots, \rho\}$, $\beta = 2^{\rho+1}$ be the basis of encoding operation. We encode the m^{th} candidate as $\zeta = \beta^m$ where $m \in \mathcal{P}$. We choose 2 as the basis of β as the division operation can be replaced by the CPU register right shift instruction to achieve a better performance.
- **Candidate decoding:** Let $k = k_t\beta^t + \cdots + k_n\beta^n + k_{n-1}\beta^{n-1} + \cdots + k_1\beta + k_0$ be the representation of k base β, $k \in \mathbb{Z}$, then we define the right shift k with n positions as $\mathsf{rsh}(k, n) = k_t\beta^{t-n} + k_{t-1}\beta^{t-n-1} + \cdots + k_{n+1}\beta + k_n$. Let $\mathsf{sum} = s_\rho\beta^{\rho-1} + s_{\rho-1}\beta^{\rho-2} + \cdots + s_2\beta + s_1$ be the addition of all the ballots where s_j is the total number of ballots that the candidate j^{th} acquires. We define $s_j := \mathsf{Decode}(\mathsf{sum}, j)$ for $1 \leq j \leq \rho$ and is computed as $s_j = \mathsf{rsh}(\mathsf{sum}, \beta^{j-1}) \bmod \beta$.

3.2 Paillier Encryption System [37]

Paillier Encryption system is employed to enable our voting system to tally the encrypted ballots. In our voting system, we implement the following functions in Paillier system and the detail of these functions are described in Appendix B.1 in the full paper [41].

- **Key Generation:** $(\mathsf{sk}_{\mathsf{Paillier}}, \mathsf{pk}_{\mathsf{Paillier}}) := \mathsf{Gen}_{\mathsf{Paillier}}(K_{\mathsf{len}})$ is the function to generate the secret key $\mathsf{sk}_{\mathsf{Paillier}}$ and the corresponding public key $\mathsf{pk}_{\mathsf{Paillier}}$ with the given key length K_{len}. The voting administrator invokes this function to generate the key pair and uploads the public key $\mathsf{pk}_{\mathsf{Paillier}}$ to the blockchain.
- **Encryption:** $C_{\mathsf{Ballot}} \leftarrow \mathsf{Enc}_{\mathsf{Paillier}}(\zeta_{\mathsf{Ballot}}, \mathsf{pk}_{\mathsf{Paillier}})$ where $\zeta_{\mathsf{Ballot}} \in \mathbb{Z}_n$ is the plaintext ballot to be encrypted. Voters download the $\mathsf{pk}_{\mathsf{Paillier}}$ from the blockchain and call this function to encrypt their ballots. This function generates the encrypted ballot C_{Ballot}.
- **Decryption:** $\zeta_{\mathsf{Res}} := \mathsf{Dec}_{\mathsf{Paillier}}(C_{\mathsf{Res}}, \mathsf{sk}_{\mathsf{Paillier}})$ where $C_{\mathsf{Res}} \in \mathbb{Z}_n^*$ is the encrypted voting result; the voting administrator invokes this function to decrypt the voting result.
- **Message Membership Proof of Knowledge** [6]**:** $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(C_{\mathsf{Ballot}}, \Upsilon)$ where C_{Ballot} is the encrypted ballot, Υ is the set of the encoded candidates. When a voter publishes his/her ballot, he/she invokes this function to generate the proof $\{v_j, e_j, u_j\}_{j\in\mathcal{P}}$ that demonstrates his/her ballot encrypts only one of the encoded candidates in Υ.
- **Decryption Correctness Proof of Knowledge:** $(\zeta_{\mathsf{Res}}, r) := \mathsf{PoK}(C_{\mathsf{Res}}, \mathsf{sk}_{\mathsf{Paillier}})$, where ζ_{Res} is the plaintext and r is the random factor that generate the encrypted voting result C_{Res}. After publishing the voting result, the voting administrator invokes this function to generate a unique value pair $(\zeta_{\mathsf{Res}}, r)$ that constructs the C_{Res} to prove that he/she decrypts the voting result C_{Res} correctly.

3.3 Linkable Ring Signatures

Linkable ring signature (LRS) can be applied in our system to protect the privacy of the voters. In practice, we apply the short linkable ring signature (SLRS) [2]

which extends all the SLRS features to make the signature size constant with the growth of voter numbers, it has the following features: (1) every ballot that is accepted by the system is from one of the valid users, (2) the voter can check whether his ballot is counted by the blockchain, (3) the size of the signature is constant to support scalability and (4) double-voting is prevented. In our voting system, we implement the function tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$. The details of these functions are explained in Appendix A.2 in the full paper [41].

- **Setup:** $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$ is a function that takes λ as the security parameter and generates system-wide public parameters param such as the group of quadratic residues modulo a safe prime product N (explained in Appendix A.2 in the full paper [41]) denoted as $\mathsf{QR}(N)$, the length of the key, and a random generator $\tilde{g} \in \mathsf{QR}(N)$.
- **Key Generation:** $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is a function to generate a key pair for each voter i.
- **Signature:** $\sigma \leftarrow \mathsf{Sign}(\mathcal{Y}, \mathsf{sk}, \mathsf{msg})$ is a function to generate the signature σ using all voters' public keys $\mathcal{Y} = \{y_1, y_2, \ldots, y_b\}$, the message msg to be signed, and the voter's secret key sk. Voters should invoke this function to sign on his/her encrypted ballot.
- **Verification:** $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(\sigma, \mathcal{Y}, \mathsf{msg})$; our voting system invokes this function to test the validity of every ballot. Based on the input of the encrypted ballot itself, the voter's signature and all the voters' public keys, the blockchain accepts or rejects this ballot to be put on the chain.
- **Linkability:** $\mathsf{Link}(\sigma_1, \sigma_2) \rightarrow \mathsf{linked/unlinked}$. When a voter casts his/her vote, our voting system invokes this function to check whether this voter has already casted his vote. If this function returns linked, our voting system rejects this ballot; otherwise, the ballot is recorded on the chain.

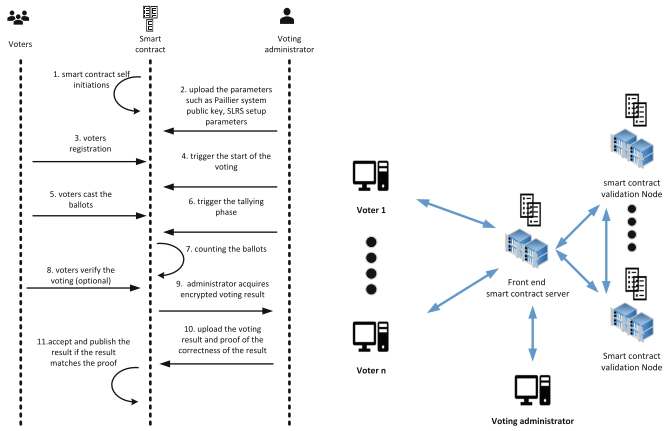
4 The Voting Protocol

In this section, we first provide an overview of the whole voting protocol and then discuss each step in details. The whole voting process can be divided into 11 steps as shown in Fig. 1(a). Except smart contract administrator, three entities are involved: voters, smart contract, and voting administrator.

The voting system consists of one front-end smart contract and several validation nodes as shown in Fig. 1(b). The role of a validation node is to replicate the execution of the smart contract codes to ensure its correct execution. For a practical voting, the validation nodes could be held by different entities/stakeholders, thus all ballots on the blockchain have been verified by different entities/stakeholders. As all the entities/stakeholders have the agreement on the data stored on the blockchain, the blockchain built on the servers owned by different entities can be regarded as trustworthy.

4.1 Entities in the Voting Process

Four entities should be involved in our voting system shown in Fig. 1(a), and details are explained as follows:



(a) The voting protocol diagram. (b) The blockchain evoting system.

Fig. 1. The general voting protocol and how entities are connected.

- **smart contract administrator:** he/she has the ability to access the smart contract platform to deploy/terminate smart contract. In Hyperledger fabric, this account is authorised by the membership service and a permission is granted to deploy/terminate the Smart contract. In our voting system, we need at least one smart contract administrator to deploy the voting smart contract.
- **voting administrator:** The role of voting administrator is to organize the vote by setting up the voting parameters and triggering the tallying and result publishing phase. Although there are underlying mechanisms in hyperledger to authenticate users, we use SLRS to prevent administrator from linking the ballots with the users.
- **smart contract:** The role of smart contract include (1) store the encrypted ballots. (2) verify the validity of the ballots. (3) count the encrypted ballot. (4) verify the correctness of the voting result. (5) publish the voting result and provide the platform for the voters to verify the voting process.
- **voters:** Voters are the people who have the rights to cast their vote. They need to register into the voting system before they cast their vote.

4.2 Voting System Set Up

During the system set up, the voting administrator uploads the following three parameters to the blockchain:

- The public key of the Paillier system $\mathsf{pk}_{\mathsf{Paillier}}$.
- A set of encryptions of zero denoted as T, generated by the administrator's Paillier public key $\mathsf{pk}_{\mathsf{Paillier}}$. For voting with 1 million voters, we suggest the set

should contain enough elements to make the randomised pool large enough and the detail of T is discussed in receipt-freeness analysis[3].
– The SLRS scheme parameters, param.

4.3 Voter Registration

Bob must register this voting system with his identity. The registration information could be: (1) email address with a desired password, or (2) the identity number with a desired password, or (3) an invitation URL sent by the administrator with a desired password. After Bob passes the identity check conducted by the smart contract, he can login with the desired password to download the SLRS param and the administrator's Paillier public key $\mathsf{pk}_{\mathsf{Paillier}}$, then generates the SLRS key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ by calling $\mathsf{KeyGen}(\mathsf{param})$; Bob then uploads the public key pk_i to the smart contract (Bob's secret key is kept off-chain by himself). If the smart contract accepts his SLRS public key, the smart contract puts his public key pk_i on the blockchain to complete his registration phase.

4.4 Vote Casting Phase

During this phase, Bob casts his vote as follows: (1) Bob chooses one of the candidates $m \in \mathcal{P}$ and encodes it as $\zeta := \mathsf{Encode}(m)$. (2) Bob invokes the Paillier encryption function $C \leftarrow \mathsf{Enc}_{\mathsf{Paillier}}(\zeta, \mathsf{pk}_{\mathsf{Paillier}})$. (3) Bob needs to prove that C is an encryption of an element in $\{\zeta_1, \ldots, \zeta_\rho\}$ (set of all encoded candidates) by calling $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(C, \Upsilon)$; hence he sends $\pi = \{C, \{v_j, e_j, u_j\}_{j\in\mathcal{P}}\}$ to the smart contract.[4] (4) Upon receiving $\{v_j, e_j, u_j\}_{j\in\mathcal{P}}$, the smart contract verifies the validation of the encrypted ballot C. We denote ϕ as a mapping of this transaction's session id to T domain. If C is valid, the smart contract takes an encryption of zero at ϕ^{th} position. Let ϵ be the addition of $T[\phi]$ and C. The smart contract signs on ϵ denoted as s and sends (ϵ, s) back to Bob. (5) If Bob accepts s, he invokes $(v, \tilde{y}, \sigma') := \mathsf{Sign}(\epsilon, (\mathsf{pk}, \mathsf{sk}), \mathcal{Y})$ to generate the $\mathsf{Sign}_{\text{bob}}$ on ϵ and sends $(\epsilon, \mathsf{Sign}_{\text{bob}})$ to the smart contract. (6) If the smart contract detects that $\mathsf{Sign}_{\text{bob}}$ has already been recorded on the blockchain or cached in the memory, it rejects Bob's vote; otherwise, $(\epsilon, \mathsf{Sign}_{\text{bob}})$ is put on the blockchain.

4.5 Ballots Tallying and Result Publishing

Due to the Paillier system's homomorphic feature, counting the vote is as simple as fetching the encrypted ballots from the blockchain and adding them together. The result publishing mechanism is described in the following steps: (1) Let E_{sum}

[3] To avoid requiring the administrator to upload the encryption of zero pool, coin flipping protocol can be applied to generate the consistent encryption of zero on all the validation nodes and this is our future work.

[4] Bob can choose none of the actual candidates by casting his ballot to a dummy candidate. When the smart contract publishes the voting result, it ignores all the ballots that the dummy candidate gets.

be the sum of all the encrypted votes and Sign_s be the signature signed by the smart contract on E_{sum}. The smart contract sends $(E_{\mathsf{sum}}, \mathsf{Sign}_s)$ to the administrator. (2) The administrator invokes $\mathsf{sum} := \mathsf{Dec}_{\mathsf{Paillier}}(E_{\mathsf{sum}}, \mathsf{sk}_{\mathsf{Paillier}})$ to compute plaintext sum, which encodes the ballots of each candidate. The administrator also invokes $(\mathsf{sum}, r) := \mathsf{PoK}(E_{\mathsf{sum}}, \mathsf{sk}_{\mathsf{Paillier}})$ to calculate the random r that constructs this E_{sum}, and sends (sum, r) to the smart contract. (3) The smart contract verifies the correctness of (sum, r) by checking if $E_{\mathsf{sum}} \stackrel{?}{=} g^{\mathsf{sum}} r^n$ (g is one of the elements of $\mathsf{pk}_{\mathsf{Paillier}}$). (4) If the smart contract accepts the sum, it iteratively invokes $m := \mathsf{Decode}(\mathsf{sum}, i)$ to compute the ballots for each candidate i. Let Π be the dictionary holding the voting result of all candidates. The smart contract finally puts Π on the blockchain.

4.6 Ballot Verifying

After tallying ballots and before the voting administrator decrypts the tallying result, the public can verify ballots on the blockchain to make sure the validity of the voting process. We define two roles for people who can verify the voting. The first one is those who have the right to access the data on the blockchain while they do not have the right to vote. The second one is those who have both rights to vote and access the data on the blockchain. The public verifiability to those who have first role is as follows (1) Checking the number of ballots that were counted and the number of people registered for this voting. (2) Checking the correctness of the tallying result by downloading and adding all the encrypted ballots to match with the tallying result published on the blockchain. Compared to those who have the first role, people assigned to the second role can also verify that his/her ballot is recorded on the blockchain by checking whether there exists one ballot that is signed with his/her signature; This ensure his/her vote is recorded and counted.

In practice, we suggest enhancing the trustworthiness of the blockchain by allowing different political parties/stakeholders host their own validation nodes. The data on the blockchain is verified by different entities/stakeholders, and it is unlikely that these entities/stakeholders collude with each other to publish a false voting result.

5 Correctness and Security Analysis

5.1 Correctness Analysis

The correctness of our voting system is guaranteed by the public verifiability provided by the smart contract and the proof of knowledge provided by the cryptographic schemes. More than that, the smart contract ensures the consistency of a transaction execution. Any inconsistencies generate an error which results in the rejection of the transaction. This means the voting participants can be assured that every transaction on the blockchain is verified and accepted by all participating nodes. This prevents compromised nodes from putting an invalid

data on the blockchain unless the adversary can take control of a proportion of the nodes in the whole blockchain network[5].

5.2 Security Features of Our Voting System

- **Privacy:** The ballots on the public ledger are encrypted and only the voting administrator who initiates the voting can decrypt the ballots. This ensures that the tallying center can count the ballots without knowing the content of the ballots.
- **Anonymity:** The voters, candidates, or smart contract cannot tell the public key of the signer with a probability larger than 1/b, where b is the number of the voters. This can be guaranteed by the anonymity property of the linkable ring signature (LRS) scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Double-voting-avoided:** In our voting system, we take the advantage of linkability provided by the short ring signature scheme. This means it is infeasible for a voter to generate two signatures such that they are determined to be unlinked. Our system can detect whether the signatures are from the same voter. Hence, a voter can only sign on one ballot and cast his/her ballot only once. This can be guaranteed by the linkable property of the LRS scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Slanderability-avoided:** A voter cannot generate a signature which is determined to be linked with another signature not generated by him/her. In other words, an adversary cannot fake other voters' signature. This can be guaranteed by the non-slanderability property of the LRS scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Receipt-freeness:** Even if an adversary obtains a voter's secret key, the adversary cannot prove that this voter voted for a specific candidate. This is guaranteed by the addition of encryption of zero which provides additional randomness to the ciphertext which is unknown to the voter. Thus, even if the voter's secret key is disclosed, no one can prove his casted ballot. For our voting system, the security level of the receipt-freeness is affected by the size of the encryption of zero pool, as the voters can collude with each other to reconstruct the encryption of zero pool. One solution is increase the size of zero pool thus more voters is required to reconstruct the pool. Another solution is applying coin flipping protocol on all validation node to work out a consistent randomness encryption of zero for each valid ballot. We have taken the first approach in this paper with 4096 encryptions of zeros.
- **Public verifiability:** Anyone who has the relevant rights to access the blockchain can verify that all the ballots are counted correctly; moreover, voters can also verify whether their ballots have been recorded.
- **Correctness:** Proof-of-knowledge ensures the correctness of the voting process. Voting participants need to prove the correctness of the interactions

[5] The number of nodes to be compromised depends on the underlying consensus protocol.

with the blockchain. Even if some blockchain nodes are compromised, others can still verify whether the proof is correct.
– **Vote-and-Go:** Compared with the voting system proposed in [31], our voting system does not need the voter to trigger the tallying phase. Moreover, in our system, voters can also cast their vote and quit before the voting ends, unlike [28] which needs all voters to finish voting before tallying the ballots.

5.3 Security and Coercion-Resistance Analysis

To address the security and coercion-resistance, we make the following assumptions:

– The trustworthiness of the blockchain platform is achieved by allowing different stakeholders/entities to host the blockchain validation nodes.
– Voters cast their ballots in a secure terminal, which means it is assumed that no one stand behind a voter or uses digital devices to record the voting process. We do not take the physical voting environment security into our consideration.
– The possibility of an attacker to create a blockchain and apply a social engineering technique to launch a phishing attack is beyond our research scope.
– The administrator should not reveal the Paillier system secret key and the encryption of zeros to anyone.
– Voters should cast their ballots by themselves. No one else can cast the ballot with a voter's identification except the voter himself.

We demonstrate the robustness of our system under two typical attacks below.

Man-in-the-Middle Attacks: Our voting system has strong resistance to this attack. First, as the voters and the smart contract both sign their messages and the voting data is encrypted, it is impossible for an adversary to forge the signature or alter the data on any parties involved in the transactions. Second, the public keys used for signature verification are all published on the blockchain, preventing the adversary from cheating any parties by replacing the original public key with the adversary's public key. The encryption of the ballot also eliminates the possibility of the ballot leakage.

Denial-of-Service (DoS) Attacks: DoS attack is feasible to launch since the network service is provided in a relatively centralised way. In addition, the servers have relatively limited processing ability for a large number of requests. Distributing the service on different nodes is one of the solutions to DoS attack as it is almost impossible for the adversary to compromise all the servers.

Coercion-Resistance Analysis: Coercion-Resistance means it is infeasible for the adversary to determine whether a coerced voter complies with the demand. Our voting system security features discussed in Sect. 5.2 make it impossible to launch the Ballots-buyer attack and Double voting attack. Additionally, our voting system is free from randomization attack.

For the Ballots-buyer attack, an attacker coerces a voter by requiring that he submits a randomly composed ballot. Under such circumstances, 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. For our system, it is impossible to launch this attack as the voter should prove that the ciphertext is one out of ρ encrypted candidates by calling $\{v_j, e_j, u_j\}_{j\in\mathcal{P}} := \mathsf{PoK}_{\mathsf{mem}}(\mathsf{Enc}_{\mathsf{Paillier}}(\zeta, \mathsf{pk}), \Upsilon)$. Since Υ is held by the smart contract, any ballot that is not the encryption of any element in Υ is rejected and the voter is notified that this transaction is failed.

6 System Deployment and Experiments

6.1 System Deployment

Our voting system can be deployed in any blockchain platforms with smart contract capability and achieve the same level of security. There might be some other reasons to choose a particular platform such as voting latency and flexibility requirements. Different consensus protocols have significant impact on the blockchain network latency and node scalability [38]. If the ballots' confirmation latency is not a major issue for the voting system, the PoW-based blockchain system could be a good option to achieve maximum node scalability. Otherwise, a BFT-based blockchain platform is a better solution. In our scenario, we employ the BFT-based blockchain platform Hyperledger Fabric and deploy our voting system in a practical scenario.

6.2 Experiments and Performance Evaluation

We deploy our system in docker containers running on a desktop with 4 cores i5-6500 CPU and 8 GB DDR3 memory. We conduct 1 million voters voting process on the blockchain that consists of 4 validation nodes and 1 PBFT leader node. Each of the validation nodes runs in one dedicated container; thus, we run five docker containers to build our testing blockchain system. We set a voter's public key as 1024 and 2048 bits respectively and the Paillier key pairs as 1024 bits. The deployment pattern is shown in Fig. 1(b). We summarize the time spent on our employed cryptographic processes for 1 million voters' voting in Table 1.

Voting Parameters Setting Up Time (Administrator Side): To initialise the voting, the administrator is responsible for uploading the voting parameters as discussed in Sect. 3. Let t_{cal} be the time taken to generate T, and t_{upload} be the time spent on uploading T to the blockchain. With 1024 bits key length, the pool size is 1 MB. According to our test, t_{upload} is $<$1 s and T_{cal} is about 14 s. In conclusion, under 100 MB bandwidth network, on the smart contract side, the majority of the time is spent on bottom half key accumulation, and on the administrator side, the most time-consuming phase is generating and uploading T (the pool of encryption of zeros).

Table 1. Time consumed on each step.

Step	Time
Generate T	13,560 ms
Bottom half key accumulation	<34 s
Top half key accumulation	<23 ms
Download parameters	4 ms
Upload ballots	≈776 ms
Tally	3,850 ms
Decode and publish	<2,000 ms

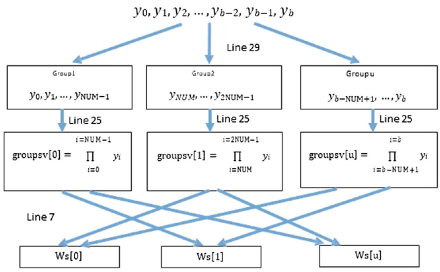
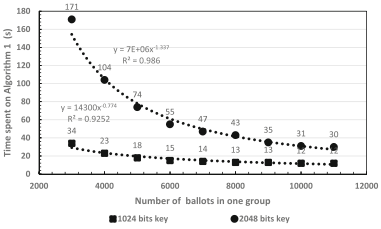


Fig. 2. The diagram of Algorithm 1.

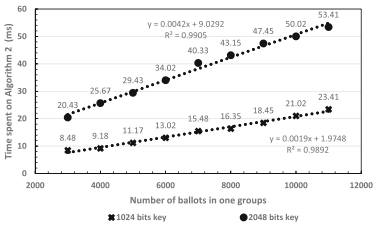
SLRS Parameter Setting Up Time: Compared with LRS, SLRS enables the size of the signature constant no matter how many signers are involved in this signature. This feature is critical important for a large scale voting (i.e. the number of voters > 100,000 1024-bits keys) as the signature should be constant to be suitable for storing on the blockchain. Compared with LRS, SLRS needs an extra step that accumulates all the signers' public keys. Let y_i be the public key of i^{th} voter and ψ be the SLRS public parameter for all voters. We define the key accumulation operation for all the voters' SLRS public keys for the i^{th} voter as $w_i = \psi^{y_1 y_2 \cdots y_{i-1} y_{i+1} y_{i+2} \cdots y_b}$. In order to make the time spent on key accumulation acceptable, we divide the key accumulation into two halves. The bottom half is run on smart contract and the top half is run by each voter.

Bottom Half Time Consumption (Smart Contract Side): For the bottom half (the bottom half algorithm is shown in our full paper [41]), on the smart contract, we divide the voter SLRS public keys into m groups and pre-calculate the accumulation of all the public keys except the keys in the given group i and denote this key accumulation as ws_i. A diagram that shows how bottom half algorithm works is also given in Fig. 2. We only discuss a case in which the number of the voters is larger than 500; otherwise, the voters can generate the key accumulation themselves within a reasonable computation time. We denote G as the group that contains the voters' SLRS public key pk and f the public key accumulation function. We invoke an array operation function *append* to add an element into the array. We distribute the voters' SLRS public keys into u groups and each group has Num of keys (except the last group). We denote the array WS to store all ws, and gkeys to store the voters' SLRS public keys groups. The most time-consuming part is the multiplication of public keys for each group. In our implementation, we calculate the WS using four threads to save time.

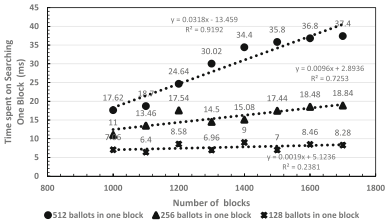
We evaluate the performance for 1 million voters' voting and the result is shown in Fig. 3(a). We find that the time spent on calculating WS decreases with the growth of the voter numbers in one group. For example, for 1024 bits length and 2048 bit length key accumulation, it decreases from 34 s and 171 s for the group that contains 3000 voters to 13 s and 35 s for the group that contains 8,000



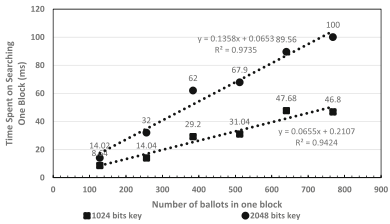
(a) Time spent in Algorithm 1.



(b) Time spent in Algorithm 2.



(c) Time versus growth of ballots in a block.



(d) Time versus growth of voter numbers.

Fig. 3. SLRS public key accumulation and searching a block on a given blockchain in 1 million voters' voting

voters, respectively. This is due to (1) the time spent on running the exponential computation loop in bottom half algorithm that dominates the time spent for the whole algorithm (2). For the 1 million voters' public key accumulation, we decrease the number of groups by increasing the number of the voters in a group to decrease the time spent on the loop in bottom half execution.

Top Half Time Consumption (Voter Side Key Accumulation): For the top half execution (The top half algorithm is shown in our full paper [41]), the voter downloads the array WS and the key group that his key belongs to in gkeys from the blockchain. The time spent on downloading these parameters is acceptable because of two reasons (1) the key size and the element size in WS are constant. (2) The number of groups is relatively small. For instance, if we have 1 million voters and we group 5000 voters in one group, and set the public key size as 1024 bits and N as 1024 bits, then the size of all the public keys in this group, denoted as $\mathcal{Y}'$, is approximately 624 KB. The size of an element in WS is restricted by SLRS parameter N; thus with the parameters above, WS is 256 KB. Therefore, the total size of $\mathcal{Y}'$ and WS is about 880 KB. The voter only needs to do one exponential operation, regardless of the voting scale. From Fig. 3(b), it can be seen that with the key size of 1024 bits, it increase from 8.48 ms for the group that contains 3000 voters to 16.35 ms for the group that contains 8000 voters. The increase of time spent for the key accumulation on the voter's side can be explained as the increase of time spent in top half execution, as it dominates the total time spent for the voter side key accumulation.

For 1 million voter's voting, we could set the number of voters in one group smaller to reduce the time spent on the voter side for key accumulation. For the key length of 1024 bits, we recommend to set each group contains about 7000 voters so that it takes 14 s and 15.48 ms on the smart contract side and the voter side key accumulation, respectively.

Based on the above experiments, it is clear that both the increases of the block size and chain length increase the time spent on searching one given block in the blockchain. For 1 million voters' voting system, the blockchain that consists of smaller blocks has better search performance. However, the drawback of the smaller block size is that it increases the number of searching operations (e.g., if we put all ballots in one block, users only searches once to get them; whereas if we allocate all of them in 10 blocks, users have to search 10 times to get them). Based on the experiment results shown in Fig. 3(c) and (d), in practice, we recommend to set each block contains 640 ballots to achieve both a reasonable search time latency and the average number of search operations.

7 Conclusion

To solve the problems that the current blockchain voting system cannot provide the comprehensive security features, and most of them are platform dependent, we have proposed a blockchain-based voting system that the voters' privacy and voting correctness are guaranteed by homomorphic encryption, linkable ring signature, and PoKs between the voter and blockchain. We analyse the correctness and the security of our voting system. The experimental results show that our voting system achieves a reasonable performance even in a large scale voting.

References

1. Adida, B.: Helios: web-based open-audit voting. In: USENIX Security Symposium, vol. 17, pp. 335–348 (2008)
2. Au, M.H., Chow, S.S.M., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006). https://doi.org/10.1007/11774716_9
3. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Constant-size ID-based linkable and revocable-iff-linked ring signature. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 364–378. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_26
4. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Certificate based (linkable) ring signature. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 79–92. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72163-5_8
5. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. Theor. Comput. Sci. **469**, 1–14 (2013)
6. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283. ACM (2001)

7. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
8. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006). https://doi.org/10.1007/11958239_12
9. Chow, S.S., Liu, J.K., Wong, D.S.: Robust receipt-free election system with ballot secrecy and verifiability. In: NDSS, vol. 8, pp. 81–94 (2008)
10. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Trans. Emerg. Telecommun. Technol. **8**(5), 481–490 (1997)
11. follow my vote. https://followmyvote.com/. Accessed 24 June 2017
12. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_66
13. Gentry, C.: A Fully Homomorphic Encryption Scheme. Stanford University (2009)
14. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_38
15. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. Technical report, 2016–1.10. Zerocoin Electric Coin Company (2016)
16. Joaquim, R., Zúquete, A., Ferreira, P.: Revs–a robust electronic voting system. IADIS Int. J. WWW/Internet **1**(2), 47–63 (2003)
17. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70. ACM (2005)
18. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_6
19. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45664-3_10
20. Laskowski, S.J., Autry, M., Cugini, J., Killam, W., Yen, J.: Improving the usability and accessibility of voting systems and products. NIST Special Publication, 256,500 (2004)
21. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36552-4_27
22. Li, C.T., Hwang, M.S., Lai, Y.C.: A verifiable electronic voting scheme over the internet. In: Sixth International Conference on Information Technology: New Generations, ITNG 2009, pp. 449–454. IEEE (2009)
23. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Trans. Knowl. Data Eng. **26**(1), 157–165 (2014)
24. Liu, J.K., Susilo, W., Wong, D.S.: Ring signature with designated linkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 104–119. Springer, Heidelberg (2006). https://doi.org/10.1007/11908739_8
25. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for Ad Hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_28

26. Liu, J.K., Wong, D.S.: Linkable ring signatures: security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganà, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005). https://doi.org/10.1007/11424826_65
27. Liu, J.K., Wong, D.S.: Enhanced security models and a generic construction approach for linkable ring signature. Int. J. Found. Comput. Sci. **17**(6), 1403–1422 (2006)
28. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. IACR Cryptology ePrint Archive 2017, 110 (2017)
29. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 116–125. ACM (2001)
30. Nsw election result could be challenged over ivote security flaw (2015). https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw
31. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028157
32. Ryan, P.Y.: Prêt à voter with paillier encryption. Math. Comput. Model. **48**(9), 1646–1662 (2008)
33. Tarasov, P., Tewari, H.: Internet voting using zcash. Cryptology ePrint Archive, Report 2017/585 (2017). http://eprint.iacr.org/2017/585
34. Theguardian: Why machines are bad at counting votes (2009). https://www.theguardian.com/technology/2009/apr/30/e-voting-electronic-polling-systems
35. Tivi voting. https://tivi.io/. Accessed 24 June 2017
36. Tsang, P.P., Au, M.H., Liu, J.K., Susilo, W., Wong, D.S.: A suite of non-pairing ID-based threshold ring signature schemes with different levels of anonymity (extended abstract). In: Heng, S.-H., Kurosawa, K. (eds.) ProvSec 2010. LNCS, vol. 6402, pp. 166–183. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16280-0_11
37. Volkhausen, T.: Paillier cryptosystem: a mathematical introduction. In: Seminar Public-Key Kryptographie (WS 05/06) bei Prof. Dr. J. Blömer (2006)
38. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_9
39. Weber, S.: A coercion-resistant cryptographic voting protocol-evaluation and prototype implementation. Darmstadt University of Technology (2006). http://www.cdc.informatik.tudarmstadt.de/reports/reports/StefanWeber.diplom.pdf
40. Xia, Z., Schneider, S.A., Heather, J., Traoré, J.: Analysis, improvement, and simplification of prêt à voter with paillier encryption. In: EVT 2008 Proceedings of the Conference on Electronic Voting Technology (2008)
41. Yu, B., et al.: Platform-independent secure blockchain-based voting system. Cryptology ePrint Archive, change me (2018). http://eprint.iacr.org/2018/
42. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. Comput. J. **56**(4), 407–421 (2013)
43. Zhao, Z., Chan, T.-H.H.: How to vote privately using Bitcoin. In: Qing, S., Okamoto, E., Kim, K., Liu, D. (eds.) ICICS 2015. LNCS, vol. 9543, pp. 82–96. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29814-6_8