Future Generation Computer Systems 112 (2020) 859–874

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities

Xuechao Yang [a,*], Xun Yi [a], Surya Nepal [b], Andrei Kelarev [a], Fengling Han [a]

[a] School of Science, RMIT University, Melbourne, VIC, 3000, Australia
[b] Data61: CSIRO, Armidale, NSW 2350, Australia

ARTICLE INFO

Article history:
Received 30 November 2017
Received in revised form 19 March 2020
Accepted 25 June 2020
Available online 29 June 2020

Keywords:
Blockchain voting
End-to-end voter verification
Self-tallying
Homomorphic addition

ABSTRACT

In organizing elections, a difficult problem is to achieve trust of all voters in the tallying process. Practical elections often lead to recounting of the submitted votes and raise questions about the validity of many submitted votes. There are even situations, when the opposition raises concerns about the validity of the whole election process due to insufficient transparency in the verification of the votes and in tallying. To solve this problem, the present paper proposes a new voting protocol based on the blockchain technology. There are several main advantages of the protocol. It does not rely on a trusted tallying authority. All votes are submitted with complete proofs of validity and are available for public access in an encrypted form. We propose a new encryption mechanism to guarantee that nobody can decrypt the votes, but everyone can verify the validity of the votes as well as the outcome of the tallying process by using the homomorphic property of the encryption. This makes the results of the election publicly verifiable. Our protocol enables all voters to store, verify, and tally all submitted votes which are added to a blockchain database. They allow every voter to rank each candidate by assigning different scores to them, rather than voting for only one candidate. Each vote is encrypted using a new encryption mechanism before submission. For each encrypted score in the vote, proofs are generated and stored. Everyone can use these proofs to verify the correctness and the eligibility of each submission without decrypting the content of the vote. This ensures the validity of the submitted votes in the counting process and at the same time maintains confidentiality. The security and performance analyses included in this paper demonstrate the feasibility of the proposed protocol for implementation in real elections.

1. Introduction

The first cryptocurrency, Bitcoin, was rated as the top performing currency in 2015 [1] and the best-performing commodity in 2016 [2]. It had more than 300k confirmed transactions daily in 2017 [3]. Since the debut of Bitcoin in 2009, its underlying technique, blockchain, has shown promising application prospects and has attracted a lot of attention from academia and industry [4–9].

Blockchain is an undeniably public and append-only ledger, which is maintained by a decentralized peer-to-peer network. The purpose of blockchain was to remove the centralized role (such as the server of a bank) for maintaining the database (the ledger). Today, the blockchain technique has been applied to many fields, including agricultural supply chain systems [10], artificial neural networks [11], blockchain-based IoT systems [12–15], business processes [16], Byzantine consensus [17], client authentication [18], clock synchronization [19], cloud data management [20], crowdsourcing and crowdsensing systems [21,22], data structures [23], digital rights management [24], distributed databases [25], healthcare data [26,27], imported product traceability [28], IoT information sharing [29], large scale networks [30], logistics supply chain of iron and steel enterprises [31], multi-agent symbiotic systems [32], payment systems [33,34], personnel big data management [35], proof of assets and transaction [36], smart cities [37], search systems [38] and trust management [39].

In this paper, we focus on decentralized online voting using the blockchain. Online voting systems that support verifiability usually assumes the existence of a public bulletin board that provides a consistent view to all voters. In practice, an example of implementing the public bulletin board can be seen in the yearly elections of the International Association of Cryptologic Research (IACR) [40]. It uses the Helios voting system [41] whose bulletin board is implemented as a single web server. This server is trusted

* Corresponding author.
E-mail addresses: xuechao.yang@rmit.edu.au (X. Yang), xun.yi@rmit.edu.au (X. Yi), surya.nepal@data61.csiro.au (S. Nepal), andrei.kelarev@gmail.com (A. Kelarev), fengling.han@rmit.edu.au (F. Han).

https://doi.org/10.1016/j.future.2020.06.051
0167-739X/

to provide a consistent view to all voters. Instead of such a trust assumption, [42] explored the feasibility of using blockchain as a public bulletin board, in which the voters are responsible for coordinating communications among themselves.

There are already proposals to use a blockchain for e-voting. The Abu Dhabi Stock Exchange is launching a blockchain voting service [43] and a recent report [44] by the Scientific Foresight Unit of the European Parliamentary Research Service discusses whether blockchain-enabled e-voting will be a transformative or incremental development. In practice, companies such as The Blockchain Voting Machine [45], FollowMyVote [46] and TIVI [47] propose solutions that use the blockchain as a ballot box to store voting data. These solutions achieve voter privacy with the involvement of trusted third parties (e.g. a central server or tallying authorities).

In this paper, we propose a new voting protocol based on the blockchain technology. They do not rely on a trusted tallying authority. All votes are submitted with complete proofs of validity and are available for public access in an encrypted form. This makes the results of the election publicly verifiable. We propose a new encryption mechanism to guarantee that nobody can decrypt the votes, but everyone can verify the validity of the votes as well as the outcome of the tallying process. In this protocol, the voters' privacy does not rely on the trust in a tallying authority, and all submitted votes can be counted without involving third parties in tallying. Our protocol allows each voter to rank all candidates by assigning different scores to them, rather than voting for just one candidate of their choice. Our new protocol employing a new encryption algorithm combines a number of advanced cryptographic techniques adapted and merged in an appropriate fashion.

Our protocol ensures that the following security requirements [48] are met.

Eligibility of voters: Only authorized voters can submit ballots.

Uniqueness: Each voter can only vote once, and multiple voting by a particular voter is detected and identified: this is also known as double voting detection.

Privacy of voters: All votes must be stored secretly and should not reveal voting preferences of the voters.

Integrity of submitted votes: No one can modify or duplicate any submitted ballot without being detected.

Correctness of the tallied result: Only verified votes are counted and added to the final result.

End-to-End Voter Verifiability: Any voter is able to verify the computation in the steps of the protocol from the following aspects: cast-as-intended, recorded-as-cast and counted-as-recorded.

Receipt-free: The voters cannot prove to others how they voted after submission (see Fig. 1).

In our proposed protocol, the authorized voters are able to register via Internet by submitting their identification documents (e.g. passport or driver's licence). Therefore, no one can register more than once, and double voting can also be prevented, because the identities of voters are public values. Voters are able to cast their ballots by assigning different scores to different candidates (ranking all candidates), rather than only voting for one candidate. The content of each individual vote is encrypted and never revealed after submission. Furthermore, voter verifiability is achieved. Any voter can act as a miner to verify the eligibilities of all submitted votes, because all corresponding proofs are generated based on the proof of zero (or partial) knowledge, which can be verified without collaboration with anyone else. Furthermore, the final result can also be computed and revealed by any individual voter.

The contributions of this paper are as follows.

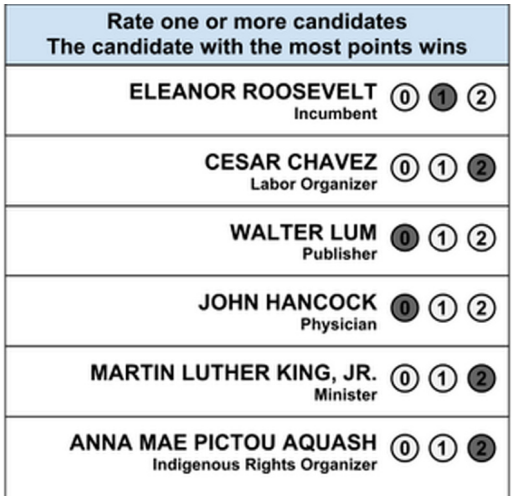

Fig. 1. Each voter can assign different scores to different candidates. However, the range of each score is defined by the election. In this case, the voter can only assign 0, or 1, or 2 to each candidate (cf. https://en.wikipedia.org/wiki/Range_voting).

1. We propose a new encryption mechanism, which merges two previously known approaches and combines the use of the secret keys of all voters and the secret keys of all candidates of the election.
2. We propose a new protocol, Protocol 1, to ensure security, privacy and public verifiability of the whole election including the tallying and vote verification processes. The protocol does not rely on a trusted tallying authority.
3. We propose a new implicit verification algorithm to verify the validity of all votes submitted to the election.
4. No adaptive issue. In our protocol, all candidates will reveal their secret keys after the voting deadline. Thus, even the individual who votes last cannot tally the result earlier than other voters since no one knows any candidate's secret key before the deadline.

Our proposed protocol, Protocol 1, is based on a new encryption mechanism, which ensures that our system satisfied all these requirements.

The organization of this paper as follow: Section 2 provides a literature review of recent work devoted to e-voting based on blockchain technology. Section 3 introduces necessary cryptographic building blocks for our protocol. Sections 4 and 5 describe our new implicit verification procedure and new encryption mechanism. Section 6 presents our proposed protocol. Security analysis and performance analysis can be found in Sections 7 and 8, respectively. Section 9 concludes this paper.

2. Related works

FollowMyVote [46] is generally regarded as the first organization to provide a remote voting (e-voting) solution using blockchain technology, which exists entirely online. Firstly, voters have to establish their identification by uploading their relevant documents (e.g. driver's licence) via a downloaded application. The general idea is to use a blind signature scheme, meaning a trusted signing/verifying party would sign a blind token from a voter. Once a voter's identification is verified, he/she is able to request an online ballot and submit his/her ballot with an unblind token to the blockchain. Furthermore, FollowMyVote allows multiple submissions from the same voter since only the most recent

ballot of the voter will be counted. Moreover, FollowMyVote provides anonymous voting because the voters cannot be identified in the blockchain. However, the voters are still clearly known to the central authority that enfranchises them. also achieved in FollowMyVote based on elliptic curve cryptography (ECC). Each voter has 2 key pairs, one for identity verification, the other one for voting, which allows voters to verify their votes without sacrificing their right to vote anonymously.

BitCongress [49] is another decentralized voting scheme that was developed based on blockchain technology. It introduced an application called AXIOMITY as the BitCongress wallet that allows voters to participate in every aspect of the democratic process. In order to vote for any candidate, each voter should create a custom vote token as a vote, and send it to the address of that candidate. A decentralized proof-of-tally is also maintained for each voter and is updated by an election upon registering the voter's vote, which gives a profile for each voter along with his/her voting history and is used in voter verification. However, the BitCongress specification does not provide details as to exactly how this is implemented. For instance, how is this information maintained throughout the blockchain so that it is easily accessible for verification purposes? Moreover, anyone can register to become a voter through BitCongress allowing them to participate in democratic processes. Each address becomes associated with a blockchain ID, allowing a person to be mapped to only one address. This ultimately requires a central authority with identity information to verify a voter and give an address in order for that voter to interact with BitCongress.

In 2017, McCorry et al. presented a smart contract implementation for the Open Vote Network that runs on Ethereum [42], which is claimed to be the first implementation of a decentralized and self-tallying voting protocol using blockchain. The procedure of casting a vote is similar to the procedure of generating a ring signature: each vote is generated using the voter's secret key and all other voters' public keys, and the final result can only be computed by using all submitted votes. An individual vote can never be revealed unless the secret key is revealed. The final result can be computed by anyone via a public accessible function. The implementation results show that the proposed protocol used with minimal setup for election, and McCorry et al. also plan to investigate the feasibility of running a larger-scale election over the blockchain in future.

3. Background and notation

In this section, we introduce the underlying cryptographic building blocks for our proposed e-voting protocol. Let us begin with Table 1 summarizing the notation used in this paper.

3.1. ElGamal cryptosystem

For preliminaries on the ElGamal cryptosystem, we refer to the monograph [50]. An example of recent application of the ElGamal cryptosystem was presented in [51]. The ElGamal cryptosystem is composed of the following algorithms.

Key generation: A key pair will be generated using the ElGamal key generation algorithm, which consists of a public key and a secret key. The key generation works as follows:

- Randomly choose a cyclic group G of a prime order q with a generator g;
- Randomly choose a secret key sk from $\mathbb{Z}_q^*$; and
- Compute the public key $\mathsf{pk} = g^{\mathsf{sk}}$.

The secret key is sk. It should always be kept secret. The public key is pk along with (G, q, g).

Encryption: Given a message $m \in G$, this can be encrypted using the public key pk as follows:

Table 1
Notation used in the paper.

v_i:	ith voter, $i \in [1, n_v]$.
n_v:	The total number of voters.
c_j:	jth candidate, $j \in [1, n_c]$.
n_c:	The total number of candidates.
x_{v_i}:	Voting secret key of ith voter, $i \in [1, n_v]$.
$g^{x_{v_i}}$:	Voting public key of ith voter, $i \in [1, n_v]$.
y_{v_i}:	Pre-computed (public) value of ith voter, $i \in [1, n_v]$.
sk_{c_j}:	Secret key for retrieving all votes for jth candidate, $j \in [1, n_c]$.
pk_{c_j}:	Public key of jth candidate, $j \in [1, n_c]$.
p:	The highest score that can be assigned to any candidate.
$p^i_{c_j}$:	The score that was assigned by voter v_i to candidate c_j.
p_{c_j}:	The final tallied score of candidate c_j.
Vote_{v_i}:	The vote that is submitted by voter v_i.
Sig_{v_i}:	Digital signature (signed by v_i's secret key).
$\mathsf{ZKP}(x_{v_i})$:	Zero knowledge proof of x_{v_i}. We use it to prove that a computed value contains x_{v_i} without revealing x_{v_i}.
$\mathsf{PKP}(p^i_{c_j})$:	Partial knowledge proof of $p^i_{c_j}$. It is used to prove the range of $p^i_{c_j}$ without revealing $p^i_{c_j}$.

- Randomly choose an integer r from $\mathbb{Z}_q^*$;
- Compute $c_1 = g^r$;
- Compute $c_2 = g^m \cdot \mathsf{pk}^r$.

The ciphertext of m is (c_1, c_2). obtained by applying ElGamal Encryption Algorithm $E(m, \mathsf{pk}) = (c_1, c_2)$.

Decryption: Given ciphertext $E(m, \mathsf{pk}) = (c_1, c_2)$, the plaintext m can only be revealed by using the secret key sk as follows:

$$D(E(M), \mathsf{sk}) = \frac{c_2}{{c_1}^{\mathsf{sk}}} = \frac{g^m \cdot \mathsf{pk}^r}{(g^r)^{\mathsf{sk}}} = \frac{g^m \cdot (g^{\mathsf{sk}})^r}{(g^r)^{\mathsf{sk}}} = g^m,$$

where D denotes ElGamal decryption algorithm. The m can be revealed by comparing the value g^m with $g^0, g^1, g^2, \ldots$ until a match is found.

Homomorphic addition: ElGamal encryption has an inherited homomorphic property [50], which allows multiplication to be performed on a set of ciphertexts, so that the decrypted result is equal to the sum of the plaintexts:

$$\begin{aligned} E(m_1, \mathsf{pk}) \times E(m_2, \mathsf{pk}) &= (g^{r_1}, g^{m_1} \cdot \mathsf{pk}^{r_1}) \times (g^{r_2}, g^{m_2} \cdot \mathsf{pk}^{r_2}) \\ &= (g^{r_1+r_2}, g^{m_1+m_2} \cdot \mathsf{pk}^{r_1+r_2}) \\ &= E(m_1 + m_2, \mathsf{pk}) \end{aligned}$$

3.2. Group-based encryption

This section contains formal preliminaries on a secret sharing scheme applied for the aggregation of encrypted data and known as a group-based encryption. It was considered, for example, in [52] and [53]. The group-based encryption with appropriate adjustments has also been applied in the voting systems designed in [54–57]. Section 5 provides more explanations of the main idea behind this scheme and how it is incorporated in our encryption mechanism.

Let G denote a finite cyclic group of prime order q in which the decision Diffie–Hellman problem is intractable. Let g be a generator in G. There are n_v users, all of whom agree on (G, g).

We assume that there are n_v different voters $v_1, v_2, \ldots, v_{n_v}$. Each voter v_i chooses a secret value $x_{v_i} \in \mathbb{Z}_q$, and computes a public value $g^{x_{v_i}}$, where $1 \le i \le n_v$. Each v_i computes a y_{v_i} as below:

$$y_{v_i} = \frac{\prod_{k=1}^{i-1} g^{x_{v_k}}}{\prod_{k=i+1}^{n_v} g^{x_{v_k}}} \tag{1}$$

which is publicly computable since the computation uses all public values $g^{x_{v_i}}$.

3.2.1. Encryption

We assume that the message for each v_i is m_i, and the encrypted message is

$$E(m_i, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} \cdot g^{m_i}$$

where $E(m_i, y_{v_i}, x_{v_i})$ denotes the message m_i encrypted using y_{v_i} and x_{v_i}.

3.2.2. Homomorphic addition

Anyone can compute the sum of all ciphertexts by multiplying all encrypted messages:

$$\prod_{i=1}^{n} E(m_i, y_{v_i}, x_{v_i}) = \prod_{i=1}^{n} (y_{v_i})^{x_{v_i}} g^{m_i} = \prod_{i=1}^{n} g^{m_i} = g^{\sum_{i=1}^{n} m_i},$$

where $\prod_{i=1}^{n} (y_{v_i})^{x_{v_i}} = 1$, according to [52].

3.3. Proof of zero knowledge: two numbers contain the same exponent

Given a group $G = \langle g \rangle = \langle h \rangle$ and public knowledge $A = g^x$ and $B = h^x$. The prover must convince the verifier that it knows x, and A and B contain the same exponent x [58,59].

Prover

- generates a random number $k \in \mathbb{Z}_q$,
- computes $K_1 = g^k$,
- computes $K_2 = h^k$,
- computes $c = \mathsf{Hash}(K_1 \parallel K_2)$,
- computes $Z = xc + k$,
- sends K_1, K_2, Z to **Verifier**.

Verifier

- computes $c = \mathsf{Hash}(K_1 \parallel K_2)$,
- verifies that $g^Z = A^c K_1$,
- verifies that $h^Z = B^c K_2$.

If the test is passed, this proves that A and B contain the same exponent, because $A^c K_1 = g^{xc} g^k = g^Z$ and $B^c K_2 = h^{xc} h^k = h^Z$.

4. An improved implicit verification procedure

In our implicit verification procedure, each submitted vote is protected by our new encryption mechanism, which invokes the public key of the voter and the public keys of all the candidates. When the secret keys of the candidates are released at the end of the vote, the value of each submitted vote remains protected by the public key of the corresponding voter. This is why all submitted votes remain confidential even after the verification and tallying process.

In this case, the ElGamal encryption algorithm (cf. Section 3.1) is used. Given a cyclic group G of a prime order q with a generator g, the secret key is sk, and the public key is $\mathsf{pk} = g^{\mathsf{sk}}$.

We assume that there are n different messages $m_1, \ldots, m_n$. The prover computes a ciphertext $E(m_i, \mathsf{pk})$, and generates proofs for a statement "$E(m_i, \mathsf{pk})$ is one of $E(m_1, \mathsf{pk}), \ldots, E(m_n, \mathsf{pk})$, where $1 \le i \le n$" is true. The verifiers can verify the statement without decrypting $E(m_i, \mathsf{pk})$, meaning they will never know which one is the truth [60].

In the following example, we assume that $m_i = m_1$ (all computation included below were carried out mod p. For easy reading we have omitted mod p in them).

Prover

- generates a random number $r \in \mathbb{Z}_q$,
- computes $E(m_1, \mathsf{pk}) = (c_1, c_2) = \{g^r, g^{m_1} \cdot \mathsf{pk}^r\}$,
- generates random numbers $t_j \in \mathbb{Z}_q$, where $j \in [1, n]$,
- generates random numbers $V_j \in \mathbb{Z}_q$, where $j \in [2, n]$,
- computes $S_j = r \cdot V_j + t_j$, where $j \in [2, n]$,
- computes $T_{0j} = g^{t_j}$, where $j \in [1, n]$ (e.g. $T_{01} = g^{t_1}$),
- computes $T_j = (g^{m_j \cdot V_j} \cdot \mathsf{pk}^{S_j}) / {c_2}^{V_j}$, where $j \in [1, n]$,
- $T_1 = \mathsf{pk}^{t_1}$ in this case, because we assume that $m_i = m_1$ (if $m_i = m_3$, $T_3 = \mathsf{pk}^{t_3}$),
- computes

$$V = \mathsf{Hash}(c_1 \parallel c_2 \parallel T_{01} \parallel T_{02} \parallel \ldots \parallel T_{0n} \parallel T_1 \parallel T_2 \parallel \ldots \parallel T_n)$$

- computes $V_1 = V \oplus V_2 \oplus V_3 \oplus \cdots \oplus V_n$, $\oplus$ denotes XOR,
- computes $S_1 = r \cdot V_1 + t_1$,
- sends $c_1, c_2, T_{01}, \ldots, T_{0n}, V_1, \ldots, V_n, S_1, \ldots, S_n$ to **Verifier**.

Verifier

- computes $T_j = (g^{m_j \cdot V_j} \cdot \mathsf{pk}^{S_j}) / {c_2}^{V_j}$ since the verifier knows $g, m_j, V_j, pk, S_j, c_2$ and V_j, where $j \in [1, n]$,
- verifies if

$$V_1 \oplus \cdots \oplus V_n = \mathsf{Hash}(c_1 \parallel c_2 \parallel T_{01} \parallel \ldots \parallel T_{0n} \parallel T_0 \parallel \ldots \parallel T_n),$$

- verifies if $g^{S_j} = T_{0j} \cdot {c_1}^{V_j}$, where $j \in [1, n]$.

There are 4 types of proofs available: T_{0j}, T_j, V_j and S_j. The prover only sends T_{0j}, V_j and S_j.

Every verifier can (1) compute T_j using V_j, S_j and V_j; (2) verify V_j using T_{0j} and T_j; (3) verify S_j using T_{0j} and V_j. The verification processes proceed iteratively in a cycle so that, if all verification steps succeed, then this means all values in the submitted vote are correct.

The following reasons explain why the verifier cannot recover the secret information from the proofs.

1. Each T_{0j} is computed by different random number t_j, which did not send/transfer at all.

2. The equality $T_{0j} = g^{t_j}$ is satisfied. It does not reveal the value of t_j to the verifier, because the secret key is x, the public key is g^x, and x is never revealed.

3. The equality $S_j = r \cdot V_j + t_j$ holds true. The verifier knows S_j and V_j, but they do not know r and t_j. Each value t_j is used only once to compute the particular S_j.

4. Only in the "real proof" the equality $T_j = pk^{t_j}$ holds. However, for verifier, all $T_j = (g^{m_j \cdot V_j} \cdot \mathsf{pk}^{S_j}) / {c_2}^{V_j}$, which did not use t_j at all.

Thus, the verifier cannot reveal any values r, t_j and m_i from the proofs.

If all verification tests return true, it can be concluded that the statement is true. This means that then the verifiers can trust that the ciphertext is one of the form $E(m_1, \mathsf{pk}), \ldots, E(m_n, \mathsf{pk})$. However, they can never know which one of the messages has been encrypted in the ciphertext.

Since all the values used to compute each T_j are public values, any verifier can compute $T_j = (g^{m_j \cdot V_j} \cdot \mathsf{pk}^{S_j}) / {c_2}^{V_j}$ by using $g, m_j, V_j, \mathsf{pk}, S_j$.

However, for the prover, only the "real proof" is pk^{t_j}, all other T_j are "fake proofs". For the verifier, all proofs are "fake proofs", even the "real" one pk^{t_j} is also equal to $T_j = (g^{m_j \cdot V_j} \cdot \mathsf{pk}^{S_j}) / {c_2}^{V_j}$. Thus, there is no way to know which one is real. In other words, the prover can prove that a ciphertext $E(m_i, \mathsf{pk})$ is either $E(1, \mathsf{pk})$, $E(2, \mathsf{pk})$ or $E(3, \mathsf{pk})$ when $n = 3$, but the verifier can never figure out whether m_i is equal to 1 or 2 or 3.

5. Our proposed encryption mechanism

Here we propose a new encryption mechanism for our election protocol. Note that it is not enough to use the classical ElGamal encryption to protect the privacy of the voters. Indeed, if only ElGamal encryption is used, and if the votes are encrypted by

voters public key, then only the voter can reveal the final result. If the content is encrypted by a voter's secret key, then everyone can view the content as the user's public key is in the public domain. Finally, if the content of the votes is encrypted by a third party's public key, then, the third party can reveal the content of all users' transactions, which is not suitable for our decentralized protocol.

This is why, to achieve appropriate protection of the voter's privacy, here we propose a new encryption mechanism, which merges ElGamal encryption and group-based encryption into one scheme. The content of votes is encrypted using each voter's secret key and is masked by applying public keys of the candidates, as explained in this section.

Our new protocol employing a new encryption mechanism combines a number of cryptographic techniques adapted and merged in an appropriate fashion. Several of these techniques have been considered previously by other authors.

The first essential idea behind our encryption mechanism is that each candidate publishes their own public key to the blockchain. Each voter encrypts the vote for each candidate using that candidate's public key. The voter then uploads the ciphertext to the blockchain. Everyone can use the homomorphic property to aggregate the votes of the candidates. At the end of the election, each candidate publishes their own private key to the blockchain, so that anyone can decrypt the final result. This idea was also used, for example, in [61], in a different setting incorporating a tallying authority.

The second essential idea behind our encryption mechanism deals with the prevention of anyone from decrypting the individual votes after the publication of the private keys of all candidates. To this end, our scheme merges the homomorphic encryption using the candidates' keys with another well-known data-aggregation technique. Since all public keys of the voters are known, a private secret share of each voter is computed in such a way that the product of all secret shares is the identity element of a group. This data aggregation method has been applied, for example in [53]. The homomorphic encryptions of the votes are further masked by these secret shares of the voters. At the end of the election, the total sum of all votes can be decrypted, since the secret shares multiply to 1, but all individual votes remain protected.

Let us now discuss the formulae used in the steps of our encryption mechanism. Denote by $p^i_{c_j}$ the score assigned by voter v_i to candidate c_j. Applying ElGamal encryption and using the public key pk_{c_j} of candidate c_j, we get the ciphertext

$$E(p^i_{c_j}, \mathsf{pk}_{c_j}) = g^r,\ g^{(p^i_{c_j})}(\mathsf{pk}_{c_j})^r.$$

After that, we apply group-based encryption (cf. Section 3.2) to mask the first part of the ElGamal ciphertext. Namely, the first part g^r of the encrypted value is "encrypted" again by using the pre-computed value y_{v_i} explained in Section 3.2 and the secret voting key x_{v_i} of the voter v_i, according to Section 3.2.1

$$E(g^r, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} \cdot g^r$$

This means that each score $p^i_{c_j}$ is encrypted by applying the following combined formula (2).

$$E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} g^r,\ g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_j})^r. \tag{2}$$

where the score $p^i_{c_j}$ is encrypted by using y_{v_i} (voting public keys of all voters), pk_{c_j} (public key of the score's recipient) and x_{v_i} (the voting secret key of the voter). Finally, the whole cast vote Vote_{v_i} can be presented as

$$\mathsf{Vote}_{v_i} = \begin{bmatrix} E(p^i_{c_1}, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i}) \\ \vdots \\ E(p^i_{c_{n_c}}, \mathsf{pk}_{c_{n_c}}, y_{v_i}, x_{v_i}) \end{bmatrix} \tag{3}$$

where a Vote_{v_i} can be treated as a container of different encrypted scores $(p^i_{c_1}, \ldots, p^i_{c_{n_c}})$ that are assigned to different candidates $(c_1, \ldots, c_{n_c})$ by the voter v_i.

Here we include additional explanation of the security of this construction as a whole. The group-based encryption requires each user to have a secret key (x) and a public key (g^x). When all voters have been registered, suppose that they have the following pairs of secret and public keys:

$$(x_1, g^{x_1}), (x_2, g^{x_2}), \ldots, (x_{n_v}, g^{x_{n_v}}). \tag{4}$$

Here $g^{x_1}, g^{x_2}, \ldots, g^{x_{n_v}}$ are public values. Then each user computes another public key y (which actually can also be computed by other users) using all public values g^x of all other voters. User 1, has x_1 and g^{x_1}, he/she computes y_1 by using all values of the other voters $g^{x_2}, g^{x_3}, \ldots, g^{x_{n_v}}$ as indicated in formula (1) in the paper. Symbolically, this can be represented in the form of equality

$$y_1 = \text{computeY}(g^{x_2}, g^{x_3}, \ldots, g^{x_{n_v}})$$

Likewise, user 2 computes y_2 by using the values $g^{x_1}, g^{x_3}, g^{x_4}, \ldots, g^{x_{n_v}}$ according to formula (1). Symbolically, this can be recorded as

$$y_2 = \text{computeY}(g^{x_1}, g^{x_3}, g^{x_4}, \ldots, g^{x_{n_v}}).$$

This can be recorded in the same way for all other voters. Finally, user n_v computes

$$y_{n_v} = \text{computeY}(g^{x_1}, g^{x_2}, \ldots, g^{x_{n_v-1}}, g^{x_{n_v-1}})$$

Then all users submit their messages: user 1 computes and submits $y_1^{x_1} * \text{message}_1$, user 2 submits $y_2^{x_2} * \text{message}_2, \ldots$, user n_v submits $y_{n_v}^{x_{n_v}} * \text{message}_{n_v}$. Here $x_1, x_2, \ldots, x_{n_v}$ are secret numbers. Everyone knows the public values $y_1, y_2, \ldots, y_{n_v}$. The whole message still can be treated as encrypted or blinded. For example, without x_1, no one can reveal $y_1^{x_1} * \text{message}_1$. Likewise, without x_{n_v}, no one can reveal $y_{n_v}^{x_{n_v}} * \text{message}_{n_v}$. According to formula (2), in our proposed voting system, each score p is first encrypted by using ElGamal cryptosystem as $g^r, g^p \mathsf{pk}^r$. Then the first part of the ciphertext, g^r, is blinded by using the voter's y^x as $y^x g^r$. The score p can be presented as

$$y^x g^r, g^p pk^r.$$

Since we assume that the voter never reveals the secret key x, it means that no one can determine g^r, and no one can decrypt the value even when the private key of the candidate is known. (Note that pk is the public key of the candidate.)

This new encryption mechanism achieves the following advantages. First, the privacy of all votes is ensured. No one can reveal any vote without the secret key of the user and the private key of the corresponding candidate. We assume that all users and all candidates are going to keep their keys confidential. Second, even the person voting last cannot determine the final outcome before submitting their vote. The final outcome can be computed without revealing contents of votes by using the homomorphic property. We assume that the candidates will apply their private keys after the deadline for the submission of votes, when all votes have completed their submissions and the final outcome is already computed in such a way that everyone can verify the validity of all calculations in the blockchain database.

6. Our proposed protocol for blockchain-based voting

We propose a new protocol for an online score voting with proofs of the validity of votes and tallying process. Protocol 1 requires all registered voters to make sure that they submit a valid vote. It can be applied only in situations, where it is possible to ensure this condition. For example, this can be used in a

situation, where all voters can be requested to submit a valid vote soon after their registration. Even during paper-based election, all voters are requested to submit their forms immediately after collecting them at the voting station. The possibilities of fines for all voters who register and do not submit a valid vote can also be considered.

Protocol 1:(Decentralized blockchain voting protocol requiring all registered voters to vote.)

1 Initialization described in Section 6.1.
2 Voter registration described in Section 6.2.
3 Vote casting described in Algorithm 1 and Section 6.2.1.
4 Verification described in Algorithm 2 and Section 6.3.
5 Tallying described in Algorithm 3 and Section 6.4.

Protocol 1 uses the encryption mechanism presented in Section 5. The main stages of the protocol are illustrated in Fig. 2. These stages are the initialization presented in Section 6.1, voter registration presented in Section 6.2, vote casting presented in Section 6.2.1, proof generation presented in Section 6.2.2, verification presented in Section 6.3, and tallying presented in Section 6.4.

6.1. Initialization

In the initialization stage, all candidates generate their key pairs (private key and public key), and broadcast them to the blockchain database which can be accessed by anyone.

All voters and candidates agree on (G, g), where G denotes a finite cyclic group of prime order q in which the Decisional Diffie–Hellman (DDH) problem is intractable, and g is a generator in G.

We assume that there are n_c candidates. Each candidate c_i chooses a private key $\mathsf{sk}_{c_i} \in \mathbb{Z}_q$, and computes the public key $\mathsf{pk}_{c_i} = g^{\mathsf{sk}_{c_i}}$, where all pk_{c_i} will be broadcast to the blockchain database, which is publicly readable but becomes immutable once the content is added to that database. Thus, the first block should contain all candidates' public keys, such as $\mathsf{pk}_{c_1}, \ldots, \mathsf{pk}_{c_{n_c}}$, which are generated by the election organizers.

Further, the highest accepted score p is defined in this stage. For example, if $p = 3$, the score for each candidate can only be 0 or 1 or 2 or 3. Any score which is greater than p or less than 0 will result in the whole vote being treated as invalid.

6.2. Voter registration

In the registration stage, all voters register to the voting system by providing verified photo identification document(s). Successfully registered voters must upload their public keys and keep their private key secret at all times.

Each voter v_i chooses a voting private key $x_{v_i} \in \mathbb{Z}_q$, and computes a voting public key $g^{x_{v_i}}$, where $i \in [1, n_v]$.

There are two ways to register to vote in the election:

(1) voters submit their identification documents (e.g. passport or driver's licence) and their voting public key for the system; and

(2) voters present their identification document at one of the voting stations in person. Both ways will require the successfully registered voters to broadcast their public voting keys ($g^{x_{v_i}}$) to the blockchain database (publicly readable but immutable), where the second block of the database should contain all voters' voting public keys, such as $g^{x_{v_1}}, \ldots, g^{x_{v_{n_v}}}$.

6.2.1. Vote casting

In the vote casting stage, each voter casts his/her vote by assigning different scores to different candidates, and protects the contents by using his/her own private key and all candidates' public keys.

We assume that all voters are registered and all public voting keys $g^{x_{v_i}}$ of all successfully registered voters are added to the blockchain database. Thus, the pre-computing value y_{v_i} of any voter can be computed by using all other $g^{x_{v_i}}$ via Eq. (1).

Our proposed e-voting protocol allows voters to assign different scores to different candidates according to their personal preferences, and the winner is the candidate who receives the highest score. There are three phases in the voting stage: pre-computing, vote casting and proof generation.

Our proposed voting protocol does not remove the connection between the identity of voters and their votes, meaning everyone can see if or when a voter submitted his/her vote. However, the content of their votes are encrypted. This implies that no one is able to reveal the content of any individual vote.

Each voter is able to assign any score between 0 and p to candidates. Because each vote consists of multiple scores, those scores are treated as private and confidential to the voters. Thus, the scores must be encrypted before submission. In our case, we use $p^i_{c_j}$ to denote a score that is assigned by voter v_i to candidate c_j, which is encrypted using our novel encryption mechanism presented in Section 5 and using formulae (2) and (3).

6.2.2. Proof generation

In order to allow anyone (not just miners and voters, but anyone who can access the blockchain database) to verify each encrypted score $E(p^i_{c_1}, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i})$ without decrypting the ciphertext and revealing the content, the voters are required to generate two kinds of proofs for each encrypted score before submission: they are $\mathsf{PKP}(p^i_{c_j})$ and $\mathsf{ZKP}(x_{v_i})$.

- $\mathsf{PKP}(p^i_{c_j})$: partial knowledge proof of $p^i_{c_i}$, which is used to prove the value of score $p^i_{c_j}$ is in the range between 0 and p (where p is pre-defined at the Initialization stage).
- $\mathsf{ZKP}(x_{v_i})$: zero knowledge proof of x_{v_i}, which is used to prove the encrypted score is computed correctly using voting private key x_{v_i} of the voter v_i.

The voter v_i has to generate both of $\mathsf{ZKP}(x_{v_i})$ and $\mathsf{PKP}(p^i_{c_j})$ for each encrypted score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$. The summarized processing procedure of the voting stage is shown as Algorithm 1.

$\mathsf{PKP}(p^i_{c_j})$ and $\mathsf{ZKP}(x_{v_i})$ are based on the proof of zero knowledge (cf. Section 3.3) and the proof of partial knowledge (cf. Section 4). The digital signatures employ the ElGamal signature scheme (refer to [62,63]).

In order to protect the privacy of the voters, each score is encrypted (cf. Eq. (2)) before submission. However, voters have to generate corresponding proofs to prove their votes are cast correctly by observing the following:

1. the score is between 1 and p ($p = 3$ in this case);
2. the encrypted score is computed correctly using the voter's private key x_{v_i}.

Next, we present the proofs generation for an encrypted score in vote Vote_{v_i}, where we assume that the encrypted score is $E(1, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} \cdot g^{r_1}, g^1 \cdot (\mathsf{pk}_{c_1})^{r_1}$:

Generating proofs for $E(\ell, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i})$, **where** $1 \leq \ell \leq n_c$:

Next, we present the proofs generation for an encrypted score in vote Vote_{v_i}, where we assume that the encrypted score is $E(\ell, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} \cdot g^r, g^\ell \cdot (\mathsf{pk}_{c_1})^r$.

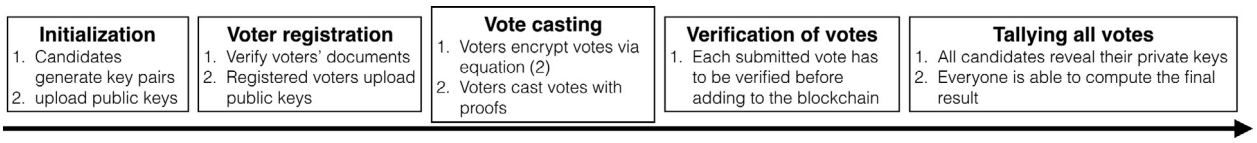

Fig. 2. Main steps of Protocol 1.

Algorithm 1: Voting (pre-computing, vote casting and proof generation).

Input : Voter v_i, private key of v_i: x_{v_i},
the highest score for each candidate: p,
all voting public keys of voters $g^{x_{v_1}}, \ldots, g^{x_{v_n}}$,
all public keys of candidates $\mathsf{pk}_{c_1}, \ldots, \mathsf{pk}_{c_{n_c}}$.
Output: Vote_{v_i} and Sig_{v_i}

. Compute y_{v_i}. ▷ cf. Eq. (1)
// Verify if v_i submitted a Vote_{v_i} previously.
// Verify if the voter is eligible (all public keys of the eligible voters are added to the blockchain).
if *search*(Sig_{v_i}) == *true* **then**
  throw; ▷ if there is a signature of v_i in the Blockchain
end
Set $\mathsf{Vote}_{v_i} = []$.
for $j \leftarrow 1$ **to** n_c **do**
  $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i}) = \left((y_{v_i})^{x_{v_i}} \cdot g^r, g^{p^i_{c_j}} \cdot \mathsf{pk}^r_{c_j}\right)$. ▷ cf. Eq. (2)
  $\mathsf{PKP}(p^i_{c_j})$: $\{T_{01}, \ldots, T_{0p}, V_1, \ldots, V_p, S_1, \ldots, S_p\}$. ▷ cf. Section 4
  $\mathsf{ZKP}(x_{v_i})$: $\{K_1, K_2, Z_1, Z_2\}$. ▷ cf. Section 3.3
  $\mathsf{Vote}_{v_i} = \mathsf{Vote}_{v_i} \cup [E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i}), \mathsf{PKP}(p^i_{c_j}), \mathsf{ZKP}(x_{v_i})]$.
end
Generate the signature $\mathsf{Sig}_{v_i} = \mathsf{Sign}(\mathsf{Vote}_{v_i})$. ▷ Sign the vote using v_i's private key.
return $\mathsf{Vote}_{v_i} = \begin{bmatrix} E(p^i_{c_1}, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i}), \mathsf{PKP}(p^i_{c_1}), \mathsf{ZKP}(x_{v_i}) \\ \vdots \\ E(p^i_{c_{n_c}}, \mathsf{pk}_{c_{n_c}}, y_{v_i}, x_{v_i}), \mathsf{PKP}(p^i_{c_{n_c}}), \mathsf{ZKP}(x_{v_i}) \end{bmatrix}, \mathsf{Sig}_{v_i}$.

(1) The score ℓ is between 1 and n_c (cf. Section 4):
- Generate random numbers $t_1, t_2, \ldots, t_{n_c} \in \mathbb{Z}_q$, and $V_1, \ldots, V_{\ell-1}, V_{\ell+1}, \ldots, V_{n_c} \in \mathbb{Z}_q$.
- For $j = 1, \ldots, \ell-1, \ell+1, \ldots, n_c$, compute

$$S_j = t_j + r \cdot V_j$$

- For $j = 1, 2, \ldots, n_c$, compute

$$T_{0j} = g^{t_j}/(y_{v_i})^{x_{v_i}}$$

$$T_j = \begin{cases} g^{\ell \cdot V_j} \cdot (\mathsf{pk}_{c_j})^{S_j}/{c_2}^{V_j} & \text{if } j \neq \ell, \\ pk^{t_\ell} & \text{if } j = \ell. \end{cases}$$

- Compute $v = \mathsf{Hash}(c_1 \parallel c_2 \parallel T_{01} \parallel \cdots \parallel T_{n_c} \parallel T_1 \parallel \cdots \parallel T_{n_c})$.
- Compute $V_\ell = v \oplus V_1 \oplus \cdots \oplus V_{\ell-1} \oplus V_{\ell+1} \oplus V_{n_c}$, where $\oplus$ denotes XOR.
- Compute $S_\ell = t_\ell + r \cdot V_\ell$.
- $\mathsf{PKP}(\ell)$: $\{T_{01}, \ldots, T_{0n_c}, V_1, \ldots, V_{n_c}, S_1, \ldots, S_{n_c}\}$.

(2) The encrypted score $E(\ell, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i})$ is computed correctly via Eq. (2) using private key x_{v_i} (cf. Section 3.3):
- Generate random numbers $k_1, k_2 \in \mathbb{Z}_q$.
- Compute $K_1 = (y_{v_1})^{k_1} \cdot g^{k_2}$.
- Compute $K_2 = g^{k_1}$.
- Compute $c = \mathsf{Hash}(K_1 \parallel K_2)$.
- Compute $Z_1 = x_{v_i} c + k_1$.
- compute $Z_2 = rc + k_2$.
- $\mathsf{ZKP}(x_{v_i})$: $\{K_1, K_2, Z_1, Z_2\}$.

6.3. Verification

In the verification stage, the eligibility of each submission has to be verified before adding it to the blockchain database, which actions are processed by miners (voters can also apply to act as miners).

The verification algorithm employed in our protocol is publicly accessible. This means that anyone with access to the blockchain database can verify any submitted vote. Each Vote_{v_i} consists of multiple encrypted scores $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$, and each encrypted score has two proofs: $\mathsf{PKP}(p^i_{c_j})$, $\mathsf{ZKP}(x_{v_i})$. Both $\mathsf{PKP}(p^i_{c_j})$ and $\mathsf{ZKP}(x_{v_i})$ can be verified without revealing the content of the encrypted score. The processing procedure of verification is shown in Algorithm 2.

Algorithm 2: Verify a submitted Vote_{v_i} (can be performed by miners or voters).

Input : a vote Vote_{v_i} and the public voting key of v_i: $g^{x_{v_i}}$.
Output: Accept or Reject.

for $j \leftarrow 1$ **to** n_c **do**
  $E(p^i_{c_j}, \mathsf{pk}_{c_j}) = (c_1, c_2) = ((y_{v_i})^{x_{v_i}} \cdot g^r,\ g^{p^i_{c_j}} \cdot \mathsf{pk}^r_{c_j})$ ▷ cf. Algorithm 1
  $\mathsf{PKP}(p^i_{c_j}) = \{T_{01}, \ldots, T_{0p}, V_1, \ldots, V_p, S_1, \ldots, S_p\}$ ▷ cf. Algorithm 1
  Verify $\mathsf{PKP}(p^i_{c_j})$: ▷ cf. Section 4
  for $k \leftarrow 1$ **to** p **do**
    Verify if $g^{S_k} = T_{0k} \cdot {c_1}^{V_k}$.
    //By using publicly available values in the blockchain
    compute $T_k = (g^{k \cdot V_k} \cdot \mathsf{pk}^{S_k})/{c_2}^{V_k}$
  end
  Verify if
  $V_1 \oplus V_2 \oplus \cdots \oplus V_p = \mathsf{Hash}(c_1 \| c_2 \| T_{01} \| \ldots \| T_{0p} \| T_1 \| \ldots \| T_p)$.
  $\mathsf{ZKP}(x_{v_i}) = \{K_1, K_2, Z_1, Z_2\}$ ▷ cf. Algorithm 1
  Verify $\mathsf{ZKP}(x_{v_i})$: ▷ cf. Section 3.3
  Verify if $(y_{v_i})^{Z_1} g^{Z_2} = (c_1)^{\mathsf{Hash}(K_1 \| K_2)} K_1$.
  Verify if $g^{Z_1} = (g^{x_{v_i}})^{\mathsf{Hash}(K_1 \| K_2)} K_2$.
end
if *all the above verifications are passed* **then**
  return Accept
else
  return Reject

Only if both $\mathsf{PKP}(p^i_{c_j})$ and $\mathsf{ZKP}(x_{v_i})$ are verified, the score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ can be considered as valid. And only when all encrypted scores in a Vote_{v_i} are verified as valid, can the Vote_{v_i} be treated as a verified submission, and added to the blockchain database.

Next, we give the computation details of verifying $\mathsf{PKP}(\ell)$ and $\mathsf{ZKP}(x_{v_i})$. In the verification procedure of the proofs, we treat $E(\ell, \mathsf{pk}_{c_1}, y_{v_i}, x_{v_i}) = (c_1, c_2)$.

1. Verification of $\mathsf{PKP}(\ell)$.
   - For $j = 1, \ldots, \ell-1, \ell+1, \ldots, n_c$, compute $T_j = (g^{j \cdot V_j} \cdot {\mathsf{pk}_{c_1}}^{S_j})/{c_2}^{V_j}$.
   - Verify if $V_1 \oplus V_2 \oplus V_3 = \mathsf{Hash}(c_1 \| c_2 \| T_{01} \parallel \cdots \parallel T_{03} \parallel T_1 \parallel \cdots \parallel T_3)$.
   - For $j = 1, \ldots, \ell-1, \ell+1, \ldots, n_c$, compute $g^{S_j} = T_{0j} \cdot {c_1}^{V_j}$.

2. Verification of $\mathsf{ZKP}(x_{v_i})$:

   - verify if $(y_{v_i})^{Z_1} g^{Z_2} = K_1 \times (c_1)^{\mathsf{Hash}(K_1 \| K_2)}$,
   - verify if $g^{Z_1} = K_2 \times (g^{x_{v_i}})^{\mathsf{Hash}(K_1 \| K_2)}$,

where y_{v_i} and $g^{x_{v_i}}$ are public values.

6.4. Tallying

In the tallying stage, all candidates reveal their private keys, and all users who can access the blockchain database are able to compute the final result.

Once the deadline of the election has expired, the organizers will check if all voters have submitted their votes and if they submitted valid votes.

In our protocol, we assume that all voters have submitted their votes, all candidates must broadcast their private keys

$$\mathsf{sk}_{c_1}, \mathsf{sk}_{c_2}, \ldots, \mathsf{sk}_{c_{n_c}} \tag{5}$$

to the blockchain.

Under our proposed algorithm, each score is encrypted using our developed ElGamal encryption (cf. Eq. (2)), where the ciphertexts can be computed according to homomorphic addition (cf. Section 3.1). In this case, we can simply multiply all valid Vote_{v_i} in the blockchain database, such as below.

$$\begin{aligned}\prod_{i=1}^{n_v} \mathsf{Vote}_{v_i} &= \begin{bmatrix} \prod_{i=1}^{n_v} E(p_{c_1}^i, \ldots) \\ \vdots \\ \prod_{i=1}^{n_v} E(p_{c_2}^i, \ldots) \end{bmatrix} \\ &= \begin{bmatrix} \prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} g^{r_1}, & \prod_{i=1}^{n_v} g^{(p_{c_1}^i)} (\mathsf{pk}_{c_1})^{r_1} \\ \vdots & \\ \prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} g^{r_{n_c}}, & \prod_{i=1}^{n_v} g^{(p_{c_{n_c}}^i)} (\mathsf{pk}_{c_2})^{r_{n_c}} \end{bmatrix},\end{aligned}$$

where we assume that there are n_v voters and n_c candidates.

Since $\prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} = 1$ (as explained in Section 3.2),

$$\begin{aligned}\prod_{i=1}^{n_v} \mathsf{Vote}_{v_i} &= \begin{bmatrix} \prod_{i=1}^{n_v} g^{r_1}, & \prod_{i=1}^{n_v} g^{(p_{c_1}^i)} (\mathsf{pk}_{c_1})^{r_1} \\ \vdots & \\ \prod_{i=1}^{n_v} g^{r_{n_c}}, & \prod_{i=1}^{n_v} g^{(p_{c_{n_c}}^i)} (\mathsf{pk}_{c_2})^{r_{n_c}} \end{bmatrix} \\ &= \begin{bmatrix} (g^r, & g^{\sum_{i=1}^{n_v} p_{c_1}^i} (\mathsf{pk}_{c_1})^r) \\ \vdots & \\ (g^r, & g^{\sum_{i=1}^{n_v} p_{c_{n_c}}^i} (\mathsf{pk}_{c_1})^r) \end{bmatrix}.\end{aligned}$$

Finally, the total score of each candidate c_j can be revealed by using the published private key sk_{c_j} of all candidates,

$$\begin{aligned}\begin{bmatrix} p_{c_1} \\ \vdots \\ p_{c_{n_c}} \end{bmatrix} &= \begin{bmatrix} D\big(g^r, & g^{\sum_{i=1}^{n_v} p_{c_1}^i} (\mathsf{pk}_{c_1})^r, & \mathsf{sk}_{c_1}\big) \\ & \vdots & \\ D\big(g^r, & g^{\sum_{i=1}^{n_v} p_{c_{n_c}}^i} (\mathsf{pk}_{c_{n_c}})^r, & \mathsf{sk}_{c_{n_c}}\big) \end{bmatrix} \\ &= \begin{bmatrix} g^{\sum_{i=1}^{n_v} p_{c_1}^i} \\ \vdots \\ g^{\sum_{i=1}^{n_v} p_{c_{n_c}}^i} \end{bmatrix},\end{aligned}$$

where we use p_{c_j} to denote the total tallied score of the candidate c_j, and $D(\ldots)$ denotes decryption algorithm of ElGamal decryption (cf. Section 3.1). The winner of the election can be determined by comparing $p_{c_1}, \ldots, p_{c_{n_c}}$. The candidate who receives the highest score is the winner.

The self-tallying algorithm is shown as Algorithm 3, which can be used by anyone.

Algorithm 3: Self-tally all votes

Input : all votes $\mathsf{Vote}_{v_1}, \ldots, \mathsf{Vote}_{v_{n_v}}$
all privates keys of candidates $\mathsf{sk}_{c_1}, \ldots, \mathsf{sk}_{c_{n_c}}$
Output: $p_{c_1}, \ldots, p_{c_{n_c}}$

```
sets p_{c_1}, ..., p_{c_{n_c}} = 1
for i ← 1 to n_v do
    for j ← 1 to n_c do
        p_{c_j} = p_{c_j} × E(p^i_{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})        ▷ cf. Section 3.1
    end
end
for j ← 1 to n_c do
    p_{c_j} = D(p_{c_j}, sk_{c_j}) = D(∏_{i=1}^{n_v} E(p^i_{c_j}, pk_{c_j}), sk_{c_j}) = g^{∑_{i=1}^{n_v} p^i_{c_j}}
end
return p_{c_1}, p_{c_2}, ..., p_{c_{n_c}}
```

7. Security analysis

Formal models and rigorous formal security proofs are very important. Our protocol represents a special case, since it is sophisticated and relies on the underlying blockchain mechanism and several cryptographic primitives (adapted and merged in a nontrivial fashion). These primitives have been published in peer-reviewed journals, and subsequently have been used in academic publications by other authors. It follows that a formal proof of security of our whole protocol would have to be very long. This is why here we include only brief sketches of proofs. It remains an open problem to provide complete formal proofs. The authors believe that this is an excellent option for future work, and that in the future this can be accomplished in several separate papers by various authors. We believe that a number of formal models and the corresponding formal proofs would need to be published in several articles in order to address the user requirements, which may often vary in practice.

In our brief model considered in this paper, the adversaries are only those legitimate participants (voters or candidates) who may wish to change the outcomes of the election by carrying out some steps of the protocol incorrectly or submitting incorrect data, or may wish to compromise the privacy of other users by eavesdropping on communications or collecting publicly available information and decrypting it. Here is a proposed brief summary of security assumptions, which may serve as a concise outline guiding the future development of complete formal security models and formal proofs. We assume that the security model will include all standard assumptions required for the successful operation of the blockchain mechanism and the blockchain database is correctly organized, including the proof-of-work and consensus mechanism (cf. [64]) so that the blockchain is secure and unbreakable. In particular, we assume that all communication between the users and the blockchain ledger is safe and secure. This ensures that there are no adversaries capable of isolating any user from the blockchain ledger or replacing the user's submissions to the blockchain by counterfeit entries. In the future complete and rigorous security proofs the best possible option is probably to incorporate the conditions discussed in a formal model proposed in [64], since the blockchain mechanism introduced there is more efficient than other versions suitable for the current application. Let us refer to [65] for detailed recommendations on creating a comprehensive list of adversaries and clear definitions of the adversary goals, assumptions and capabilities in a complete formal security model.

The voters have to be registered, and we assume that the registration authority is honest. Otherwise, it can forfeit the election outcomes by introducing a large cohort of fake voters or by denying the registration of legitimate voters. The requirement to

have safe and secure communication is also necessary to enable all users to see the list of all voting public keys submitted by the users. After the registration, the list of all ids/names is a public knowledge, and we assume that the communication facilities are safe and secure so that everyone can view this list correctly, as well as all other public data. We assume that all information added to the blockchain database can be accessed by everyone but is immutable.

The security analysis presented in the following subsections aims to provide an outline of brief proofs establishing that our protocol satisfies all security requirements explained in Section 1. We also assume that the candidates and voters keep their private keys secret. A voter revealing his/her private key will compromise the secrecy of their cast vote. The analysis assumes that all voters never share their private keys with anyone else, and that candidates never share their private keys until the voting has closed.

7.1. Eligibility of voters

The identification of voters has to be verified in order for them to participate in the election.

Theorem 1. *Only successfully registered voters are able to submit their votes.*

As mentioned in the first paragraph of Section 7, here we include only brief sketches of formal models and proofs.

Proof. In our protocol, all voters have to register before the election starts. The brief model considered in this paper assumes that the registration authority is honest. Indeed, a dishonest registration authority can influence the election outcomes by adding a large number of spurious voters or by preventing legitimate voters from registering. All successfully registered voters are required to upload their voting public keys. Each successfully registered voter v_i has to upload his/her voting public key $g^{x_{v_i}}$, where x_{v_i} is his/her voting secret key of v_i. The public voting keys will be added to the blockchain database. Due to the feature of the blockchain, everyone can see the list of all voting public keys but no one is able to modify it.

Once the election starts, each voter should generate a digital signature Sig_{v_i} by using his/her voting private key x_{v_i}, and submit it along with his/her vote (e.g. Vote_{v_i} and Sig_{v_i} from the voter v_i). After submission, anyone can verify the eligibility of each vote by verifying the digital signature (refer to [62,63]) using the corresponding voting public key from the blockchain database. (Since each Sig_{v_i} is generated by using x_{v_i} of v_i, it follows that the Sig_{v_i} can be verified by using $g^{x_{v_i}}$ of v_i, because x_{v_i} and $g^{x_{v_i}}$ form a key pair).

If no corresponding key is found in the database for this voter v_i, then the submission will be treated as invalid and will be discarded. If the digital signature Sig_{v_i} of a submission is found to be invalid by using the corresponding (previously uploaded) voting public key $g^{x_{v_i}}$ of v_i, then the submission will be treated as invalid and will be discarded. □

7.2. Uniqueness

In order to ensure the fairness of the election, each verified voter is only allowed to contribute one vote to the election.

Theorem 2. *Each voter can only submit once, which means that multiple submissions from the same voter can be detected and discarded.*

Proof. In our protocol, the id/name of each submission is public information for everyone: only the content of the submission is encoded. Suppose that a voter v_i submitted a valid vote with his/her digital signature. Then both Vote_{v_i} and Sig_{v_i} are stored in the blockchain database. Everyone is able to see that the voter v_i has already submitted a vote, because the name/id of v_i is not hidden in the blockchain ledger. Only the content of Vote_{v_i} is encrypted. The content of the whole database should look like this:

$$\begin{aligned} v_1 &\Rightarrow \{\mathsf{Vote}_{v_1}, \mathsf{Sig}_{v_1}\},\\ v_2 &\Rightarrow \{\mathsf{Vote}_{v_2}, \mathsf{Sig}_{v_2}\},\\ &\vdots\\ v_{n_v} &\Rightarrow \{\mathsf{Vote}_{v_{n_v}}, \mathsf{Sig}_{v_{n_v}}\}. \end{aligned}$$

Thus, multiple submissions can be detected easily by searching the database for the id/name of each voter v_i.

If a submission has been added to the database with the same id/name, the latest submission will be discarded, which means that our system will only keep the first submission for each voter and ignore the other submissions from the same voter. This matches the standard practice adopted in all regular elections, where each voter is allowed to submit only one vote, and it not allowed to vote again after his/her first vote had been submitted. □

7.3. Voter privacy

Each submitted vote contains the voting preference of the voter, which can be treated as private to the voter and must be kept secret at all times. The vote is protected by using the voter's private key and the candidates' public keys.

Theorem 3. *The content of any individual vote will never be revealed.*

Proof. In our protocol, each Vote_{v_i} consists of as many encrypted scores as the number of candidates. Each score is given by an expression $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ (cf. Algorithm 1). Each encrypted score is generated by using the following parameters: the score $p^i_{c_j}$ for candidate c_j, public key pk_{c_j} of the candidate c_j, precomputed y_{v_i} of v_i and the voting secret key x_{v_i} of v_i. All these parameters are public values with the exception of the x_{v_i} of v_i), which is encoded by using not only the public key pk_{c_j} of the candidate, but also the voting private key x_{v_i} of the voter. In our proposed algorithm, each score $p^i_{c_j}$ is encrypted twice. In the first step, the $p^i_{c_j}$ is encrypted by using pk_{c_j} and ElGamal encryption algorithm, and the encrypted value is treated as

$$E(p^i_{c_j}, \mathsf{pk}_{c_j}) = \{g^r, g^{p^i_{c_j}} \cdot (\mathsf{pk}_{c_j})^r\}$$

At this moment, the $E(p^i_{c_j}, \mathsf{pk}_{c_j})$ can only be decrypted by using the secret key sk_{c_j}. In other words, only candidate c_j can decrypt $E(p^i_{c_j}, \mathsf{pk}_{c_j})$. However, the encrypted value $E(p^i_{c_j}, \mathsf{pk}_{c_j})$ will be encrypted again. In the second step of our proposed algorithm, the first part of $E(p^i_{c_j}, \mathsf{pk}_{c_j})$ (the encrypted value of ElGamal encryption has two separate parts, such as g^r and $g^{p^i_{c_j}} \cdot (\mathsf{pk}_{c_j})^r$) is further encrypted by using y_{v_i} and x_{v_i} of the voter v_i as $(y_{v_i})^{x_{v_i}} \cdot g^r$. The final encrypted score can be represented as

$$(y_{v_i})^{x_{v_i}} \cdot g^r, g^{p^i_{c_j}} \cdot (\mathsf{pk}_{c_j})^r.$$

At this stage, the encrypted score cannot be decrypted by using either sk_{c_j} or x_{v_i}.

Indeed, suppose that the adversary trying to decrypt it has only the sk_{c_j}. The only thing the adversary could then do is to

apply the ElGamal decryption algorithm. However, the adversary cannot remove $(y_{v_i})^{x_{v_i}}$ from $(y_{v_i})^{x_{v_i}} \cdot g^r$ and get the g^r, and cannot get the plaintext $p^i_{c_j}$ without g^r.

On the other hand, if the adversary has only x_{v_i}, then the only thing the adversary can do is to try removing $(y_{v_i})^{x_{v_i}}$ from $(y_{v_i})^{x_{v_i}} \cdot g^r$ in order to get g^r. However, then it remains impossible to decrypt the ElGamal ciphertext, because the sk_{c_j} is not available.

Thus, each encrypted score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ can only be decrypted by using both the private key (sk_{c_j}) of the candidate and the voting private key x_{v_i} of the voter.

Although all candidates will reveal their sk_{c_j}'s after the voting is closed, the $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ cannot be decrypted without the x_{v_i}.

Since it is assumed that the voters will never disclose their voting private keys, it follows that the content of any encrypted vote will never be revealed. Without the secret keys of the voters there is no way to decode the encrypted scores. □

Remark 1. It is essential to clarify that a complete formal security proof would require reducing the problem of decrypting submitted votes to one of the well-known hard problems. We have only indicated why several possible concrete attacks fail. The authors could not come up with any other examples of attacks and would like to acknowledge that in a rigorous proof the claim of the theorem should be confirmed by a formal security reduction. This paper contains only brief outlines of security proofs, because the proposed protocols are very complex, include many steps, and complete formal security proofs would be long.

7.4. Integrity of all submissions

Our protocol uses a blockchain database, which is maintained by all users (miners). All votes will be verified before being added to the database, whereupon the content cannot be further modified. In the brief model considered in the present paper, we assume that the blockchain database is secure and unbreakable. We assume that a complete the security model will include all standard assumptions required for the secure operation of the blockchain mechanism and the blockchain database is correctly organized. In particular, we assume that all communication between the users and the blockchain ledger is safe and secure. This ensures that there are no adversaries capable of isolating any user from the blockchain ledger or replacing the user's submissions to the blockchain by counterfeit entries.

As mentioned above, we had to assume that the registration authority is honest, since otherwise the whole election simply cannot be trustworthy. Since the registration authority also has access to the blockchain database, as all other users do, in the future complete security models it can be involved in those cases of verification and maintenance of the blockchain entries, where there may be arguments between some of the individual users contributing to the database ledger.

Theorem 4. *No one can modify any data in a submission once it is confirmed and added to the blockchain database.*

Proof. A blockchain database is a decentralized database, which means that there is no centralized server managing the database, but everyone has the latest version of the whole database. In this situation, no one can modify any data in the database because the modification will not be accepted by most users (and miners) who are (at least generally) assumed to be honest.

Furthermore, there is a smart contract in the blockchain for recording all the rules and logics for the particular election. This also helps to prevent anyone from any harmful action to the database and to keep the election results secure.

To sum up, without a general notification no one is able to change anything in the blockchain database. Either most users (miners) or the registration authority will reject any unauthorized modification, given the assumption that either most users are honest, or at least the registration authority can be made involved in case of any disputes. This means that the integrity of all or any submissions cannot be compromised once they have been confirmed and added to the blockchain database. □

7.5. Correctness of final result

It is very easy to verify the submitted secret key of each candidate used in the tallying algorithm, because the public key of each candidate is a public knowledge. Everyone knows the public key, and so everyone can use it to verify the submitted secret key of the candidate. Therefore, it is impossible for the candidate to influence his/her outcome by submitting an incorrect secret key. This will be discovered immediately, and as a result the candidate will be assigned the score equal to 0 as if nobody voted for him/her. Thus, a candidate cannot gain any benefit by providing a fake private key.

We assume that everything added to the blockchain database can be accessed by everyone but is immutable, and the correctness of the final result can be analysed from two aspects.

Theorem 5. *Any individual submission can be verified by anyone without accessing the content of the submission.*

Proof. A submission can only be included in the final result if it has been verified as valid using Algorithm 2. The corresponding proofs of each submitted vote are generated by the voter according to the proof of partial knowledge (cf. Section 4) and the proof of zero knowledge (cf. Section 3.3).

We use proof of partial knowledge to allow each voter to prove that each encrypted score is valid score, for example, the value of each score must be between 0 and the maximum number equal to the number of candidates. Each voter must prove that he/she did not assign any invalid score, such as -100, or 200 to any candidate. The proof $\mathsf{PKP}(p^i_{c_j})$ consists of V_j, S_j and T_{0j}, where V_j are random generated values, $S_j = t_j + r \cdot V_j$, which is computed by using random values, and r comes from the ElGamal encryption. Further, T_{0j} is computed by using y_{v_i}, x_{v_i} and random values t_j (more details can be found in Section 4). None of these values will reveal the encrypted score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$.

We also used proof of zero knowledge to allow each voter v_i to prove he/she uses correct voting secret key x_{v_i} to generate the submission Vote_{v_i}. The proof $\mathsf{ZKP}(x_{v_i})$ consists of K_1, K_2, Z_1 and Z_2 (more details can be found in Section 3.3). Here K_1 is computed by using y_{v_i} and two random values, K_2 is computed by using g and a random value, Z_1 is computed by using the voting secret key x_{v_i} of the voter v_i, and Z_2 is computed by using r (from the ElGamal encryption) and a random value. As in the case of $\mathsf{PKP}(p^i_{c_j})$, none of these values will reveal the encrypted score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$.

Both of the proof of partial knowledge and proof of zero knowledge have been used and verified many times by various researchers. By combining the features of these protocols, each submission can be verified by anyone without revealing any knowledge of the confidential information in the content of the vote. □

Theorem 6. *The final tallied result can be computed by anyone in the end of election using all candidates' private keys.*

Proof. Once all submissions are recorded in the blockchain, anyone is able to add up all votes cast for each candidate and compute the sum of the votes in an encrypted form, while keeping the

contents of the votes secret and encrypted by the private keys of the voters, according to the additive homomorphic property (cf. Section 3.2.2).

In our protocol, all candidates c_j have to reveal their private keys sk_{c_j} once all Vote_{v_i} are received and verified as valid. At that point, everyone or anyone can simply decrypt the sum (ciphertext) for any candidate via Algorithm 3 because the blind part of each submission will be taken out according to the group-based encryption (cf. Section 3.2), which means anyone can use the revealed private key of the particular candidate to decrypt the sum (ciphertext) and get the final result of the candidate.

As we described in Theorem 3, revealing all secret keys sk_{c_j} of candidates will never reveal any Vote_{v_i} because each encrypted score $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ in Vote_{v_i} is encrypted by using both public key pk_{c_j} and voting secret key x_{v_i} of v_i. According to our proposed algorithm 2, there is no way to reveal the score $p^i_{c_j}$ from $E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$ without x_{v_i}. After sk_{c_j} is revealed, only the voter v_i is able to reveal Vote_{v_i} as only v_i has the voting secret key x_{v_i} of v_i, we assume that all voters will keep their private keys secret, because it does not make any sense for v_i to reveal his/her own submission Vote_{v_i}. □

7.6. End-to-end voter verification

End-to-end voter verification is achieved by our proposed protocol. The corresponding analysis has the following three aspects.

Theorem 7. *Each vote is cast-as-intended, which means that the voter is able to verify that the vote has been generated and recorded in the ledger correctly as intended.*

Proof. In our protocol, each vote is cast based on ElGamal encryption (cf. Section 3.1) and group-based encryption (cf. Section 3.2) using all candidates' public keys pk_{c_j}, all voters' voting public keys y_{v_i} and the voter's voting private key x_{v_i} (cf. Section 1) as below

$$E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i}) = \left((y_{v_i})^{x_{v_i}} \cdot g^r, g^{p^i_{c_j}} \cdot \mathsf{pk}^r_{c_j}\right)$$

As all the necessary additional elements and components $(y_{v_i}, x_{v_i}, g, r, p^i_{c_j}, \mathsf{pk}_{c_j})$ for this particular vote Vote_{v_i} are public for the voter v_i, he/she can re-generate the vote in order to verify that the Vote_{v_i} has been generated and recorded in the ledger correctly. □

Theorem 8. *Each submission is recorded-as-cast, meaning any modification to the submission can be discovered without difficulty.*

Proof. ElGamal encryption is a probabilistic encryption. This means that the encrypted result will always be different even if one encrypts the same plaintext multiple times. For example, the encryption algorithm of ElGamal uses a random value r every time (cf. Section 3.1)

$$E(m) = g^r, g^m \cdot \mathsf{pk}^r$$

where r is a random number. It is different every time, which means that

$$E(m) \neq E(m).$$

However, the decryption algorithm of ElGamal eliminates the random number r by applying

$$g^m \cdot \mathsf{pk}^r / (g^r)^{\mathsf{sk}},$$

so that

$$D(E(m)) = D(E(m)) = m.$$

Therefore, each voter v_i is able to save his/her cast vote Vote_{v_i} as an original receipt before submission, because each ciphertext of ElGamal encryption is unique and different from others even if the plaintexts are the same. Once the submission is added to the blockchain, the voter can easily verify if the recorded submission is the one he/she submitted by comparing the content with the content of the original receipt, in order to prevent the modification of the vote by any middle man during the submission, or any change of the record as a result of tampering with the database after submission. □

Theorem 9. *Each vote is counted-as-recorded, meaning that the voter is able to verify if his/her vote has contributed correctly to the final result.*

Proof. After the official result is published, the voters are able to verify the result by re-computing it if they have any doubts and do not completely trust the officially published result. In our protocol, a self-tally algorithm (cf. Algorithm 3) can be used by any individual voter.

According to the proposed Algorithm 3, anyone can tally any candidate's result by multiplying all submitted votes in the database. For example, to calculate the tally result of c_j, any user can compute

$$\prod_{i=1}^{n_v} E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i})$$

and then, simply decrypt the multiplication result by using revealed secret key (sk_{c_j}) of c_j as follows

$$D\left(\prod_{i=1}^{n_v} E(p^i_{c_j}, \mathsf{pk}_{c_j}, y_{v_i}, x_{v_i}), \mathsf{sk}_{c_j}\right).$$

This means that everyone is able to tally all submissions and find the winner without any collaboration with the others. This enables everyone to verify that the published result is correct. At the same time, the content of each individual Vote_{v_i} will never be revealed. □

Theorem 10. *Each submitted vote remains secret after all candidates' private keys have been revealed.*

Proof. According to the proposed Algorithm 2, every score within a submitted vote is encrypted by using both the voter's secret voting key and the particular candidate's public key as follows:

$$(y_{v_i})^{x_{v_i}} g^r,\ g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_j})^r. \tag{6}$$

The voter's secret voting key x_{v_i} will never be revealed, which means that the first part of the encrypted score $(y_{v_i})^{x_{v_i}} g^r$ always remains secure. Based on this, the following detailed analysis given below demonstrates that it remains impossible to recover the plaintext score $p^i_{c_j}$ from $g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_j})^r$, even after the candidates' private keys have been revealed.

First, note that r is a random number, and it is impossible to remove r from Eq. (6) directly.

Second, when sk_{c_j} is revealed, everyone can get

$$g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_j})^r = g^{(p^i_{c_j})} \cdot (g^{\mathsf{sk}_{c_j}})^r, \tag{7}$$

but it still remains impossible to determine r, as the following explanation shows.

It is impossible to determine sk_{c_1} from $\mathsf{pk}_{c_1} = g^{\mathsf{sk}_{c_1}}$ directly. Furthermore, in our protocol, even if the values g, sk_{c_j} and $g^{\mathsf{sk}_{c_j}}$ are known to everyone, it still remains impossible to reveal r from $(g^{\mathsf{sk}_{c_j}})^r$, since this is the well-known discrete logarithm problem.

Therefore, it is impossible for an adversary to determine $p^i_{c_j}$ from $g^{(p^i_{c_j})}(g^{\mathsf{sk}_{c_j}})^r$ without prior knowledge of r.

Third, the adversary cannot reveal the $p^i_{c_j}$ by dividing two different scores from the same vote. To illustrate, let us consider two different encrypted scores from the same vote given by the formulae

$$(y_{v_1})^{x_{v_1}} g^{r_1},\ g^{(p^1_{c_1})} \cdot (\mathsf{pk}_{c_1})^{r_1}, \tag{8}$$

$$(y_{v_1})^{x_{v_1}} g^{r_2},\ g^{(p^1_{c_2})} \cdot (\mathsf{pk}_{c_2})^{r_2}. \tag{9}$$

The values above can be treated as voter v_1 assigning score $p^1_{c_1}$ to candidate c_1 and assigning score $p^1_{c_2}$ to candidate c_2. If an adversary is going to try revealing the first part of the ciphertexts, she/he may try to apply division as in the formulae

$$\frac{(y_{v_1})^{x_{v_1}} g^{r_1}}{(y_{v_1})^{x_{v_1}} g^{r_2}},\ \frac{g^{(p^1_{c_1})} \cdot (\mathsf{pk}_{c_1})^{r_1}}{g^{(p^1_{c_2})} \cdot (\mathsf{pk}_{c_2})^{r_2}}. \tag{10}$$

Then the adversary obtains the plaintext of the first part of the ciphertext as $g^{r_1-r_2}$. However, the second part of the ciphertext still remains hidden, because the key pk_{c_1} is different from pk_{c_2}. Thus, the adversary can only get

$$g^{(r_1-r_2)},\ g^{(p^1_{c_1}-p^1_{c_2})} \cdot \frac{(\mathsf{pk}_{c_1})^{r_1}}{(\mathsf{pk}_{c_2})^{r_2}}, \tag{11}$$

but he/she cannot obtain any of the values $p^1_{c_1}$ or $p^1_{c_2}$ even when he/she knows sk_{c_1} and sk_{c_2}.

Fourth, we are going to prove that the adversary cannot reveal $p^i_{c_j}$ from all parameters of the ZKP and PKP protocols. After the sk_{c_j} has been revealed, the random number r still remains secret, and this is the reason why the adversary cannot determine the plaintext $p^i_{c_j}$.

It remains to verify that r cannot be determined from the parameters of ZKP and PKP protocols and is kept secret all the time after sk_{c_j} has been revealed. To this end, next we consider the lists (12) and (13) of all parameters of ZKP and PKP protocols, respectively, for an encrypted score

$$(y_{v_i})^{x_{v_i}} g^{r},\ g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_i})^{r}$$

and discuss formal verification procedures of these protocols.

The list of all parameters of $\mathsf{ZKP}(x_{v_i})$ is

$$\{K_1, K_2, Z_1, Z_2\} \tag{12}$$

where

- $K_1 = (y_{v_1})^{k_1} \cdot g^{k_2}$,
- $K_2 = g^{k_1}$,
- $c = \mathsf{Hash}(K_1 \parallel K_2)$,
- $Z_1 = x_{v_i} c + k_1$,
- $Z_2 = rc + k_2$.

The relation that these parameters of ZKP protocol must satisfy can be described as the following verification procedure, which includes verifying that the voter generated the encrypted score by using his/her own secret key x_{v_i} and the recorded values are legitimate. To explain the verification procedure, let us denote the encrypted score $(y_{v_i})^{x_{v_i}} g^{r},\ g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_i})^{r}$ stored in the blockchain ledger as (c_1, c_2). In order to verify that the submitted vote is legitimate, the users proceed as follows:

- compute $c = \mathsf{Hash}(K_1 \parallel K_2)$;
- verify that $(y_{v_i})^{Z_1} g^{Z_2} = (c_1)^c + K_1$;
- verify that $g^{Z_1} = (g^{x_{v_i}})^c + K_2$, where $g^{x_{v_i}}$ is a public value of voter v_i.

The encrypted score is treated as a legitimate one only if the above two equalities are satisfied. This means that the score was generated correctly using the voter's x_{v_i}.

For $\mathsf{ZKP}(x_{v_i})$, we did not use the secret key sk_{c_j} of the candidate c_j. This means that nothing changes after sk_{c_j} has been revealed. The values k_1, k_2 and random numbers being kept secret. The numbers r and x_{v_i} are also secret values that only the voter v_i knows. Therefore, they will never be revealed. Thus, all the elements from $\mathsf{ZKP}(x_{v_i}) = \{K_1, K_2, Z_1, Z_2\}$ do not reveal any useful information after the candidate's sk_{c_j} has become known.

The list of all parameters of $\mathsf{PKP}(p^i_{c_j})$ is

$$\{T_{01}, \ldots, T_{0p}, V_1, \ldots, V_p, S_1, \ldots, S_p\} \tag{13}$$

where

- $V_1, \ldots, V_{\ell-1}, V_{\ell+1}, \ldots, V_{n_c} \in \mathbb{Z}_q$ are random numbers;
- $S_j = t_j + r \cdot V_j$, where S_j and V_j are random numbers and public values, but t_j and r are secret numbers, and t_j cannot be used to calculate r as this is the original equation of the original version of this protocol, which has not been modified;
- $T_{0j} = g^{t_j}/(y_{v_i})^{x_{v_i}}$, where $j = 1, 2, \ldots, n_c$. Here the adversary cannot determine t_j or x_{v_i} by using T_{0j} and y_{v_i}.

The relation that these parameters of PKP protocol must satisfy can be described as the following verification procedure. In order to explain the verification procedure, we denote the encrypted score $(y_{v_i})^{x_{v_i}} g^{r},\ g^{(p^i_{c_j})} \cdot (\mathsf{pk}_{c_i})^{r}$ recorded in the blockchain ledger by (c_1, c_2), and assume that anyone is able to verify the PKP of the encrypted score. This includes verifying that the voter assigned a valid score to the candidate $p^i_{c_j} \in [0, p]$, and that a negative score or a score greater than p has not been assigned. In order to verify that the submitted vote is legitimate, the users proceed as follows:

- verify that $g^{S_k} = T_{0k} \cdot c_1^{V_k}$ for each $k \in [0, p]$;
- compute $T_k = g^{k \cdot V_k} \cdot \mathsf{pk}^{S_k}/c_2^{V_k}$, for each $k \in [0, p]$;
- verify that $V_1 \oplus V_2 \oplus \cdots \oplus V_p = \mathsf{Hash}(c_1 \parallel c_2 \parallel T_{01} \parallel \cdots \parallel T_{0p} \parallel T_1 \parallel \cdots \parallel T_p)$.

The encrypted score is treated as valid only if the above two verifications are satisfied. This means that the assigned score $p^i_{c_j}$ lies between 0 and p.

The above arguments demonstrate that r remains secret after sk_{c_j} has been revealed. To summarize, each assigned score $p^i_{c_j}$ remains secret after sk_{c_j} has been revealed, because an adversary can never recover the random value r. Without prior knowledge of r, it remains impossible to get the $p^i_{c_j}$. This is equivalent to the problem of computing sk_{c_j} from $\mathsf{pk}_{c_j} = g^{\mathsf{sk}_{c_j}}$, which is intractable since r has same bit-length as sk_{c_j}. This completes the proof. □

Remark 2. As indicated in Remark 1, this paper contains only brief sketches of security proofs. In particular, it is essential to clarify that in the proof of Theorem 10 we have only indicated why several particular attacks are impossible. While the authors did not invent other types of attacks, in the complete formal security proofs it would be necessary to establish that there is absolutely no way to design any feasible procedure cancelling out the blinding factors of our protocol.

8. Performance analysis

The analysis of performance has two subsections: client (voters) and the server (miners). All tests were performed on a laptop with the following specifications: CPU: 2.2 GHz Intel Core i7, Memory: 16 GB 1600 MHz DDR3.

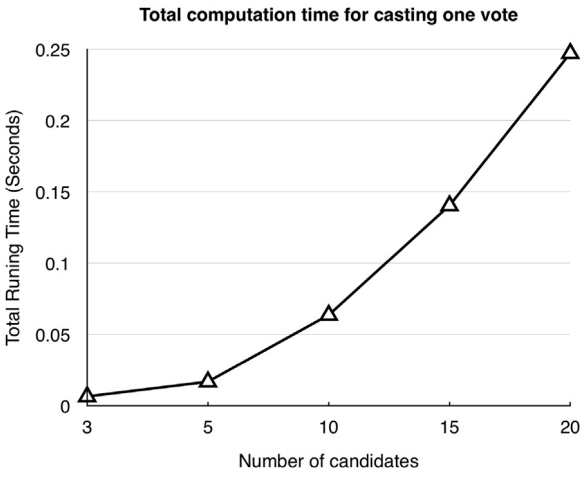

Fig. 3. Estimate of total time spent casting one vote when the number of candidates n_c are 3, 5, 10, 15 and 20, and the corresponding p is also 3, 5, 10, 15 and 20.

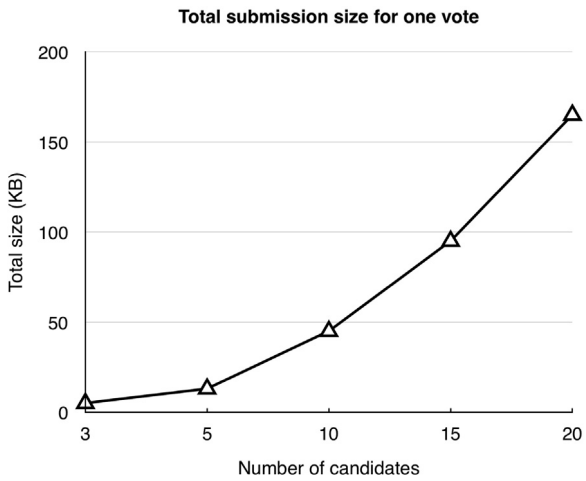

Fig. 4. Estimate of total size of one vote when the number of candidates n_c are 3, 5, 10, 15 and 20, and the corresponding p is also 3, 5, 10, 15 and 20.

All tests used high performance code from libgmp via the gmpy2 python module (cf. gmpy2.readthedocs.io/en/latest/) with 1024-bit length key (p and q are 1024-bit). We use t to denote the computation time of one exponentiation (such as a g^a mod b), in this case, $t = 0.00012$ seconds on the laptop.

ElGamal encryption requires two exponentiations, and ElGamal decryption requires one exponentiation (cf. Section 3.1). The division can be avoided by calculating ${c_1}^{q-\mathrm{sk}}$ instead of ${c_1}^{\mathrm{sk}}$, where ${c_1}^{q-\mathrm{sk}}$ is the inverse of ${c_1}^{\mathrm{sk}}$. In this case, we use t_E and t_D to denote the computation time of encryption and decryption, respectively, where $t_E = 2t$ and $t_D = t$, approximately.

8.1. Performance on voters' side

On the voters' side, each voter casts and submits a vote to the blockchain database. To prevent the voting preferences of each vote being revealed, and also to protect the voters' privacy, we require voters to encrypt their votes before broadcasting them to the blockchain database.

Thus, the performance on the voters' side can be summarized as being the total computation time for casting a vote and the total submission size of the vote.

8.1.1. Total computation time for casting one vote

In this case, we use T_{voter} to denote the total computation time on the voters' side, where we only focus on the computation cost of all exponentiations. According to Algorithm 1, the computation time for each candidate costs an ElGamal encryption computation ($2t$, cf. Section 3.1), a PKP computation ($5pt - 2t$, cf. Section 4) and a ZKP computation ($3t$, cf. Section 3.3). Therefore, the total computation time on the voters' side can be presented as

$$T_{\mathrm{voter}} = (5pt + 3t) * n_c$$

where p and n_c denotes the number of possible scores for each candidate and the number of candidates, respectively.

In this experiment, we tested the T_{voter} over five rounds on the laptop, according to different numbers of candidates ($n_c = 3$, 5, 10, 15 and 20) and different total available points ($p = n_c$). The result is shown in Fig. 3.

From the results in Fig. 3, we can see that the time taken for casting one vote is less than 0.25 s even if there are 20 candidates in the election.

8.1.2. Total submission size for one vote

In this case, we use $S_{\mathrm{one\_vote}}$ to denote the total submission size for one vote, which contains n_c encrypted value, n_c proofs of PKP and n_c proofs of ZKP. Thus, the total submission size $S_{\mathrm{one\_vote}}$ can be presented as

$$S_{\mathrm{one\_vote}} = 1024 * (3p + 6) * n_c$$

where an encrypted score for each candidate contains an ElGamal encrypted value which is $1024 * 2$ bits (cf. Section 3.1), a proof of PKP is $1024 * (3p)$ bits (cf. Section 4) and a proof of ZKP is $1024 * 4$ bits (cf. Section 3.3).

In this experiment, we also tested the $S_{\mathrm{one\_vote}}$ over five rounds on the laptop, where the numbers of candidates (n_c) and the total available points (p) are the same as the previous experiment, which is $n_c = p = 3$, 5, 10, 15 and 20. The result is shown in Fig. 4.

From the result of Fig. 4, we found the size of one submission is less than 200 kB even for a 20-candidate vote. Based on the publicly-available assessment of the speed test ranks Internet access speed in more than 100 countries [66], the slowest Internet speed is 3.03 Mbps and the fastest is 62.59 Mbps. It shows that the submission size for a vote (including all encrypted values and all proofs) of our protocol should not be problematic.

8.2. Performance on miners' side

Our protocol uses a blockchain database. Therefore, the performance on the miners' side can be summarized as having the following attributes: (1) the computation time for verifying all pending submissions and (2) the computation of self-tallying all verified votes.

Please note, here we do not discuss the computation time for solving the mathematical puzzle of the blockchain database in order to confirm a new block: the difficulty level can be defined by the organizers.

8.2.1. The computation time for verifying all pending submissions

In this case, we use T_{verify} to denote the total computation time for verifying all pending submissions, which can be treated as the sum of verification time of PKP proofs and verification time of ZKP proofs. According to Algorithm 2, the computation time for verifying a PPK and a ZPK are $5pt$ (cf. Section 4) and $5t$ (cf. Section 3.3) respectively. Thus the total computation time for verifying all pending submissions is

$$T_{\mathrm{verify}} = (5pt + 5t) * n_c * n_v$$

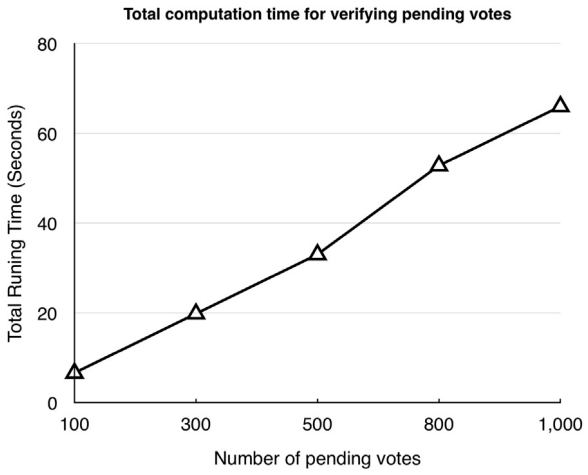

Fig. 5. Estimate total computation time (by using a laptop) for verifying 100, 300, 500, 800 and 1000 pending votes, for 10 candidates in the election.

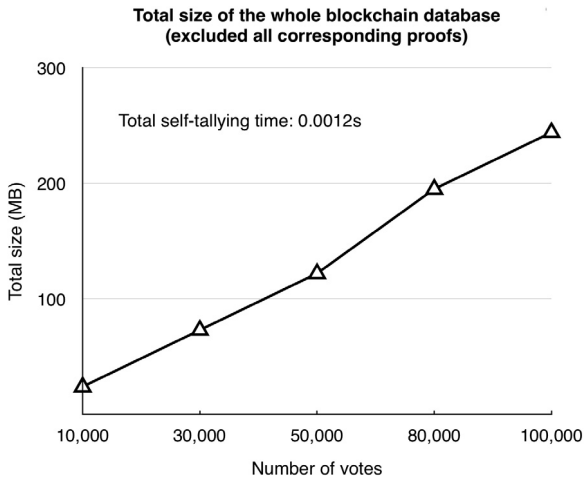

Fig. 6. Estimate total size of the blockchain database (excluding all corresponding proofs) when the total number of pending votes are 10,000, 30,000, 50,000, 80,000 and 100,000 pending votes, for 10 candidates in the election.

where n_v denotes the number of the pending submissions.

In this experiment, we assume there are 10 candidates ($n_c = p = 10$) in the election, and test the T_{verify} over five rounds on the laptop, where the numbers of pending submissions are 3, 5, 10, 15 and 20. The results are shown in Fig. 5.

From the results in Fig. 5, we found the time spent in verifying 1000 votes takes approximately one minute using a standard laptop (the specifications of which were provided at the start of this section), where the miners usually use at least a server-power computer, which means that the running time will be further reduced.

8.2.2. The cost of self-tallying all verified votes

In our protocol, all voters are able to download all votes from the blockchain database. We assume all votes must be verified before being added to this database. This means normal users do not need to download all corresponding proofs unless they want to independently verify them. Self-tallying the votes is done using Algorithm 3. In this case, we use $S_{\text{all\_votes}}$ and T_{tally} to denote the total size of the whole blockchain database (excluding all corresponding proofs) and the computation time for tallying all verified votes, which can be presented as

$$S_{\text{all\_votes}} = 1024 * 2 * n_c * n_v \text{ and } T_{\text{tally}} = t * n_c,$$

respectively, according to Algorithm 3.

In this experiment, we again assume there are 10 candidates ($n_c = p = 10$) in the election, which means $T_{\text{tally}} = t * n_c = 10t$, and we test $S_{\text{all\_votes}}$ over five rounds on the same laptop, based on five different numbers of votes (10,000, 30,000, 50,000, 80,000 and 100,000). The results are shown in Fig. 6.

From the results in Fig. 6, we found the total size of the whole blockchain database (excluding all corresponding proofs) is less than 250 MB when there are 100,000 votes. The computation time for self-tallying all votes only regards the number of candidates: in this case, it only took 0.0012 s to tally them by using a standard laptop (the specifications of which were provided at the start of this section).

9. Conclusion and future work

In this paper, we propose a new decentralized publicly verifiable online voting protocol based on blockchain technology. Our new protocol employs a new encryption mechanism and combines a number of cryptographic techniques adapted and merged in an appropriate fashion. The protocol stores all submitted votes in a blockchain database, which can be accessed by all users but is immutable. Our proposed protocol also allows voters to cast their ballots by assigning different points to different candidates. Each vote is encrypted before submission and remains encrypted at all times. The additive homomorphic property of the exponential ElGamal cryptosystem enables effective processing of the ciphertexts during these procedures. Moreover, the eligibility of voters and their submissions can be verified by anyone without revealing the contents of the votes, and our proposed verification and self-tallying algorithms allow any voter to verify the correctness of the final result. The whole blockchain database is maintained by all users (voters and miners) without a need to involve a third party in verification and tallying. This paper also provides a concise security and performance analysis and confirms the feasibility of our proposed blockchain online voting protocol for real-life elections.

In this paper, we do not hide the identification of each vote (because laws in several countries require everyone to vote). In our protocol, everyone can see the sender of each vote via the blockchain database. In future, we plan to develop an alternative algorithm, which can hide the voter's identification of each submitted vote, while at the same time allowing to keep a separate record of the voters who submitted their votes, if necessary.

Our protocol does not achieve receipt-freeness and coercion resistance considered, for example, in [61]. The authors believe that incorporating these requirements represent significant challenges and can be best addressed in separate future publications. This is why here we record these conditions as open problems to be solved in subsequent future articles.

This paper included only brief sketches of a simple model and security proofs. As explained in Section 7, we believe that a number of formal models and the corresponding rigorous formal proofs would need to be published in several separate articles in order to address various possible user requirements and cover all security assumptions of the whole system and the underlying blockchain mechanism and cryptographic primitives.

CRediT authorship contribution statement

Xuechao Yang: Conceptualization, Methodology, Software. **Xun Yi:** Supervision, Writing - original draft. **Surya Nepal:** Supervision. **Andrei Kelarev:** Writing - review & editing, Investigation. **Fengling Han:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by Discovery grant DP160100913 from Australian Research Council and Data61 Research Collaborative Project (Enhancing Security and Privacy in IoT). The authors are grateful to the anonymous reviewers for thorough reports with detailed explanations that have helped to improve this paper.

References

[1] J. Desjardins, It's official: Bitcoin was the top performing currency of 2015, 2016, http://money.visualcapitalist.com/its-official-bitcoin-was-the-top-performing-currency-of-2015/.
[2] J. Adinolfi, And 2016's best-performing commodity is ... Bitcoin?, 2016, http://www.marketwatch.com/story/and-2016s-best-performing-commodity-is-bitcoin-2016-12-22.
[3] BLOCKCHAIN, Confirmed transactions per day, 2017, 2017, https://blockchain.info/charts/n-transactions?timespan=1year.
[4] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. 107 (2020) 841–853.
[5] C. Lin, D. He, X. Huang, K.-K.R. Choo, A.V. Vasilakos, BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0, J. Netw. Comput. Appl. 116 (2018) 42–52.
[6] W. Meng, E.W. Tischhauser, Q. Wang, Y. Wang, J. Han, When intrusion detection meets blockchain technology: A review, IEEE Access 6 (2018) 10179–10188.
[7] S. Tang, S.S.M. Chow, Systematic market control of cryptocurrency inflations: Work-in-progress, in: Proc. 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC 2018, 2018, pp. 61–63.
[8] T. Volety, S. Saini, T. McGhin, C.Z. Liu, K.-K.R. Choo, Cracking Bitcoin wallets: I want what you have in the wallets, Future Gener. Comput. Syst. 91 (2019) 136–143.
[9] J.H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, K. Wehrle, Secure and anonymous decentralized Bitcoin mixing, Future Gener. Comput. Syst. 80 (2018) 448–466.
[10] K. Leng, Y. Bi, L. Jing, H.-C. Fu, I. Van Nieuwenhuyse, Research on agricultural supply chain system with double chain architecture based on blockchain technology, 86 (2018) 641–649.
[11] R.-Y. Chen, A traceability chain algorithm for artificial neural networks using T-S fuzzy cognitive maps in blockchain, Future Gener. Comput. Syst. 80 (2018) 198–210.
[12] M.U. Hassan, M.H. Rehmani, J. Chen, Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions, Future Gener. Comput. Syst. 97 (2019) 512–529.
[13] S. Higgins, IBM Invests $200 million in blockchain-powered IoT, 2016, https://www.coindesk.com/ibm-blockchain-iot-office/.
[14] M.A. Khan, K. Salah, IoT security: Review, blockchain solutions, and open challenges, Future Gener. Comput. Syst. 82 (2018) 395–411.
[15] A. Reyna, C. Martin, J. Chen, E. Soler, M. Diaz, On blockchain and its integration with IoT. challenges and opportunities, Future Gener. Comput. Syst. 88 (2018) 173–190.
[16] C. Prybila, S. Schulte, C. Hochreiner, I. Weber, Runtime verification for business processes utilizing the Bitcoin blockchain, Future Gener. Comput. Syst. 107 (2020) 816–831.
[17] V. Gramoli, From blockchain consensus back to Byzantine consensus, Future Gener. Comput. Syst. (2017) http://dx.doi.org/10.1016/j.future.2017.09.023.
[18] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, X. Lin, PTAS: Privacy-preserving thin-client authentication scheme in blockchain-based PKI, Future Gener. Comput. Syst. 96 (2019) 185–195.
[19] K. Fan, S. Sun, S. Yan, Q. Pan, H. Li, Y. Yang, A blockchain-based clock synchronization scheme in IoT, Future Gener. Comput. Syst. 101 (2019) 524–533.
[20] L. Zhu, Y. Wu, K. Gai, K.-K.R. Choo, Controllable and trustworthy blockchain-based cloud data management, Future Gener. Comput. Syst. 91 (2019) 527–535.
[21] W. Feng, Z. Yan, MCS-Chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain, Future Gener. Comput. Syst. 95 (2019) 649–666.
[22] M. Yang, T. Zhu, K. Liang, W. Zhou, R.H. Deng, A blockchain-based location privacy-preserving crowdsensing system, Future Gener. Comput. Syst. 94 (2019) 408–418.
[23] P. Otte, M. de Vos, J. Pouwelse, Trustchain: A sybil-resistant scalable blockchain, Future Gener. Comput. Syst. 107 (2020) 770–780.
[24] Z. Ma, M. Jiang, H. Gao, Z. Wang, Blockchain for digital rights management, Future Gener. Comput. Syst. 89 (2018) 746–764.
[25] M. Muzammal, Q. Qu, B.V. Nasrulin, Renovating blockchain with distributed databases: An open source system, Future Gener. Comput. Syst. 90 (2019) 105–117.
[26] L. Chen, W.-K. Lee, C.-C. Chang, K.-K.R. Choo, N. Zhang, Blockchain based searchable encryption for electronic health record sharing, Future Gener. Comput. Syst. 95 (2019) 420–429.
[27] A.A. Omar, M.Z.A. Bhuiyan, A. Basu, S. Kiyomoto, M.S. Rahman, Privacy-friendly platform for healthcare data in cloud based on blockchain environment, Future Gener. Comput. Syst. 95 (2019) 511–521.
[28] X. Xu, Q. Lu, Y. Liu, L. Zhu, A.V. Vasilakos, Designing blockchain-based applications a case study for imported product traceability, Future Gener. Comput. Syst. 92 (2019) 399–406.
[29] H. Si, C. Sun, Y. Li, H. Qiao, L. Shi, IoT information sharing security mechanism based on blockchain technology, Future Gener. Comput. Syst. 101 (2019) 1028–1040.
[30] A. Dorri, S.S. Kanhere, R. Jurdak, MOF-BC: A memory optimized and flexible blockchain for large scale networks, Future Gener. Comput. Syst. 92 (2019) 357–373.
[31] A. Yang, Y. Li, C. Liu, J. Li, Y. Zhang, J. Wang, Research on logistics supply chain of iron and steel enterprises based on block chain technology, Future Gener. Comput. Syst. 101 (2019) 635–645.
[32] R. Skowroński, The open blockchain-aided multi-agent symbiotic cyber-physical systems, Future Gener. Comput. Syst. 94 (2019) 430–443.
[33] H. Huang, X. Chen, Q. Wu, X. Huang, J. Shen, Bitcoin-based fair payments for outsourcing computations of fog devices, Future Gener. Comput. Syst. 78 (2018) 850–858.
[34] L. Zhong, Q. Wu, J. Xie, J. Li, B. Qin, A secure versatile light payment system based on blockchain, Future Gener. Comput. Syst. 93 (2019) 327–337.
[35] J. Chen, Z. Lv, H. Song, Design of personnel big data management system based on blockchain, Future Gener. Comput. Syst. 101 (2019) 1122–1129.
[36] H. Wang, D. He, Y. Ji, Designated-verifier proof of assets for Bitcoin exchange using elliptic curve cryptography, Future Gener. Comput. Syst. 107 (2020) 854–862.
[37] P.K. Sharma, J.H. Park, Blockchain based hybrid network architecture for the smart city, Future Gener. Comput. Syst. 86 (2018) 650–655.
[38] P. Jiang, F. Guo, K. Liang, J. Lai, Q. Wen, Searchain: Blockchain-based private keyword search in decentralized storage, Future Gener. Comput. Syst. 107 (2020) 781–792.
[39] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, P.D. Drobintsev, Trust management in a blockchain based fog computing platform with trustless smart oracles, Future Gener. Comput. Syst. 101 (2019) 747–759.
[40] Helios, International association for cryptologic research. about the helios system, 2016, https://www.iacr.org/elections/eVoting/about-helios.html.
[41] B. Adida, Helios: Web-based open-audit voting, in: USENIX Security Symposium, Vol. 17, 2008, pp. 335–348.
[42] P. McCorry, S.F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in: IACR Cryptology ePrint Archive, Vol. 2017, 2017, p. 110.
[43] S. Higgins, Abu dhabi stock exchange launches blockchain voting service, 2016, https://www.coindesk.com/abu-dhabi-exchange-blockchain-voting/.
[44] P. Boucher, What if blockchain technology revolutionized voting, Eur. Parliam. (2016) Unpublished manuscript.
[45] A. Hertig, The first Bitcoin voting machine is on its way, 2015, https://motherboard.vice.com/en_us/article/3da8e5/the-first-bitcoin-voting-machine-is-on-its-way.
[46] Followmyvote, Blockchain voting: The end-to-end process, 2016, https://followmyvote.com/blockchain-voting-the-end-to-end-process/.
[47] S. Saba, Now you can vote online with a selfie, 2016, http://www.businesswire.com/news/home/20161017005354/en/VoteOnline-Selfie.
[48] X. Yang, X. Yi, C. Ryan, R. van Schyndel, F. Han, S. Nepal, A. Song, A verifiable ranked choice internet voting system, in: International Conference on Web Information Systems Engineering, Springer, 2017, pp. 490–501.
[49] M. Rockwell, BitCongress - process for blockchain voting & law, 2016, http://www.bitcongress.org/.
[50] X. Yi, R. Paulet, E. Bertino, Homomorphic Encryption and Applications, Vol. 3, Springer, 2014.

[51] A. Kelarev, X. Yi, S. Badsha, X. Yang, L. Rylands, J. Seberry, A multistage protocol for aggregated queries in distributed cloud databases with privacy protection, Future Gener. Comput. Syst. 90 (2019) 368–380.
[52] F. Hao, P.Y. Ryan, P. Zieliński, Anonymous voting by two-round public discussion, IET Inf. Secur. 4 (2) (2010) 62–67.
[53] M. Chase, S.S.M. Chow, Improving privacy and security in multi-authority attribute-based encryption, in: Proc. 16th ACM Conference on Computer and Communications Security, CCS 2009, 2009, pp. 121–130.
[54] F. Hao, P. Zieliński, A 2-round anonymous veto protocol, in: International Workshop on Security Protocols, Springer, 2006, pp. 202–211.
[55] D. Khader, B. Smyth, P. Ryan, F. Hao, A fair and robust voting system by broadcast, in: Lecture Notes in Informatics (LNI), Proceedings-Series of the Gesellschaft fur Informatik (GI), Gesellschaft fur Informatik (GI), 2012, pp. 285–299.
[56] M. Arnaud, V. Cortier, C. Wiedling, Analysis of an electronic boardroom voting system, in: International Conference on E-Voting and Identity, Springer, 2013, pp. 109–126.
[57] S.T. Ali, J. Murray, An overview of end-to-end verifiable voting systems, in: Real-World Electronic Voting: Design, Analysis and Deployment, CRC Press, 2016, pp. 171–218.
[58] D. Chaum, T.P. Pedersen, Wallet databases with observers, in: Crypto, Vol. 92, Springer, 1992, pp. 89–105.
[59] E. Fujisaki, T. Okamoto, Statistical zero knowledge protocols to prove modular polynomial relations, in: Advances in Cryptology, Proc. 17th Annual Internat. Cryptology Conf., CRYPTO'97, Springer, 1997, p. 16.
[60] R. Cramer, I. Damgård, B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, in: Advances in Cryptology, CRYPTO 94, Springer, 1994, pp. 174–187.
[61] S.S.M. Chow, J. Liu, D. Wong, Robust receipt-free election system with ballot secrecy and verifiability, in: Proc. 16th Annual Network & Distributed System Security Symposium, NDSS 2008, 2008, available at http://www.isoc.org/isoc/conferences/ndss/08/papers/05_robust_receipt--free.pdf.
[62] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inform. Theory 31 (1985) 469–472.
[63] ElGamal signature scheme, 2019, https://en.wikipedia.org/wiki/ElGamal_signature_scheme.
[64] Z. Liu, S. Tang, S.S.M. Chow, Z. Liu, Y. Long, Fork-free hybrid consensus with flexible proof-of-activity, Future Gener. Comput. Syst. 96 (2019) 515–524.
[65] Q. Do, B. Martini, K.-K.R. Choo, The role of the adversary model in applied security research, Comput. Secur. 81 (2019) 156–181.
[66] K. Murnane, Speedtest ranks internet access speed in more than 100 countries, 2017, https://www.forbes.com/sites/kevinmurnane/2017/08/14/speedtest-ranks-internet-access-speed-in-more-than-100-countries/.

Xuechao Yang is a Research Fellow in the School of Science, RMIT University, Australia. He received the Bachelor degree in Information Technology from RMIT University (2013), and received Bachelor of Computer Science Honours in 2014. In 2018, He completed the Ph.D. degree in School of Science RMIT, with data61 CSIRO. His research interests include cryptosystems, homomorphism and blockchain technology.

Xun Yi is a Full Professor with the School of Science, RMIT University, Australia. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and privacy-preserving data mining. He has published more than 160 research papers in international journals, such as IEEE Trans. Computers, IEEE Trans. Parallel and Distributed Systems, IEEE Trans. Knowledge and Data Engineering, IEEE Trans. Wireless Communication, IEEE Trans. Dependable and Secure Computing, IEEE Trans. Information Forensics and Security, IEEE Trans. Circuit and Systems, IEEE Trans. Vehicular Technologies, IEEE communication Letters, IEEE Electronic Letters, and conference proceedings. He was a member of the programme committees for more than 40 international conferences. Recently, he has led several Discovery Projects of Australian Research Council (ARC) and has been a member of the ARC College of Experts. Since 2014, he has been an Associate Editor for IEEE Trans. Dependable and Secure Computing.

Surya Nepal received the B.E. degree from the National Institute of Technology, Surat, India, the M.E. degree from the Asian Institute of Technology, Bangkok, Thailand, and the Ph.D. degree from RMIT University, Australia. He is a Principal Research Scientist with CSIRO Data61. He has authored or co-authored over 150 publications to his credit. At CSIRO, he undertook research in the area of multimedia databases, web services and service oriented architectures, social networks, security, privacy and trust in collaborative environment and cloud systems and big data. His main research interest includes the development and implementation of technologies in the area of distributed systems and social networks, with a specific focus on security, privacy, and trust. Many of his works are published in top international journals and conferences, such as VLDB, ICDE, ICWS, SCC, CoopIS, ICSOC, the International Journal of Web Services Research, the IEEE Transactions on Service Computing, ACM Computing Survey, and ACM Transaction on Internet Technology.

Andrei Kelarev is a Research Fellow in the School of Science, RMIT University, Australia. He is an author of two books and 198 journal articles. He worked as an Associate Professor in the University of Wisconsin and University of Nebraska in USA, a Senior Lecturer in the University of Tasmania in Australia, and was a Chief Investigator of a large Discovery grant from Australian Research Council. He is working on cyber security applications of machine learning and data mining.

Fengling Han received the B.E. degree in the Department of Automatic Control, Harbin Engineering University, China, the M.Eng. degree in the Department of Control Engineering, Harbin Institute of Technology, China, and the Ph.D. degree in the School of Electrical and Computer Engineering, RMIT University, Australia. She is currently a Lecturer in the School of Science, RMIT University, Australia. Her research interests include observers, network security, and complex systems.