IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 4, 2018, accepted March 13, 2018, date of publication March 20, 2018, date of current version April 25, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2817518

A Secure Verifiable Ranked Choice Online Voting System Based on Homomorphic Encryption

XUECHAO YANG[1], XUN YI[1], SURYA NEPAL[2], ANDREI KELAREV[1], AND FENGLING HAN[1]
[1]School of Science, RMIT University, Melbourne, VIC 3000, Australia
[2]Data61, CSIRO, Armidale, NSW 2350, Australia

Corresponding author: Xuechao Yang (xuechao.yang@rmit.edu.au)

This work was supported by the Discovery Grant from the Australian Research Council and the Data61 Research Collaborative Project (Enhancing Security and Privacy in IoT) under Grant DP160100913.

ABSTRACT Advanced security methods are necessary to introduce effective online voting in the whole world. Elections conducted on paper consume a lot of resources and contribute to the destruction of forests, which leads to climate deterioration. Recent online voting experiences in countries, such as the United States, India, and Brazil, demonstrated that further research is needed to improve security guarantees for future elections, to ensure the confidentiality of votes and enable the verification of their integrity and validity. In this paper, we propose a ranked choice online voting system, which addresses these challenges. It eliminates all hardwired restrictions on the possible assignments of points to different candidates according to the voters' personal preferences. In order to protect the confidentiality of the votes, each cast ballot is encrypted using the exponential ElGamal cryptosystem before submission. Furthermore, during voting the system ensures that proofs are generated and stored for each element in the cast ballot. These proofs can then be used to verify the correctness and the eligibility of each ballot before counting without decrypting and accessing the content of the ballot. This validates the votes in the counting process and at the same time maintains confidentiality. The security and performance analyses included in this paper demonstrate that our method has achieved significant improvements in comparison with the previous systems. The outcomes of our experiments also show that our proposed protocols are feasible for practical implementations.

INDEX TERMS Online voting, privacy preservation, homomorphic encryption, homomorphism tally, end-to-end verification.

I. INTRODUCTION

Homomorphic encryption is a well-known powerful technique with many useful applications (cf. [1]–[5]). Recently, it has been applied to the design of online voting systems (see the next section for details). This is motivated by the need to develop advanced security systems to facilitate a broad introduction of online voting throughout the whole world. Elections conducted by paper votes are unsustainable, as they consume a lot of resources and lead to destruction of forests contributing to deterioration of climate. Recent experimental online voting in countries such as the United States, India and Brazil highlighted significant challenges that require further research to improve security guarantees in future elections.

Secure e-voting systems are required for casting votes using the Internet. Online voting systems not only increasing sustainability, but also reduce the overall cost of running elections and may increase voter participation because of the more convenient procedure, in particular, for the voters with disabilities. The study of electronic elections contributes to the more general area of privacy-preservation (cf. [6]–[10]) and relies on secure implementations of other aspects involved in e-voting (cf. [11]–[14]). Since online voting remains vulnerable to malicious activity and hacking attacks, the design of a secure, flexible and verifiable e-voting system is a very important problem (cf. [15], [16]).

In this paper, we propose an e-voting system inspired by the so-called approval voting also known as score voting, [17]. Approval voting has been used in various elections since 1987. Examples include elections conducted by some scientific and engineering societies, an econometric society and democratic state committees, [17]. However, to the best of our knowledge, score voting has not been applied in secure, flexible and verifiable e-voting systems.

Our e-voting system enables voters to score all candidates and assign points to different candidates directly without any restrictions apart from the total number of available points

specified by the organizers of the election. This is illustrated in Fig. 1 where the total number of available points is equal to 6.

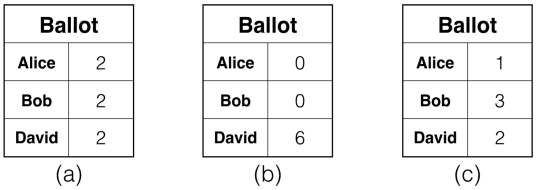

FIGURE 1. Here (a), (b) and (c) are the voting mechanism of our e-voting system when the total number of available points is equal to 6. A voter can treat all candidate equally as in (a), or support only one candidate as in (b), or rank all candidates as in (c).

Our ranked choice e-voting system constructed in this paper uses the exponential ElGamal cryptosystem due to [18] (see also [2], [5]). Before submission, the contents of each cast ballot are encrypted using the exponential ElGamal encryption. The additive homomorphism property of this cryptosystem, [19], makes it possible to tally encrypted ballots directly without decrypting them. Our cryptosystem also includes cryptographic proofs incorporated to ensure the integrity of the voting process and to verify the validity of each vote before it is counted.

Maintaining the privacy and security of the voters is a priority for any online voting system. Our voting system ensures that the following security requirements are met. These requirements are essential for voting according to [20]–[22].

Eligibility of Voters: Only authorized voters can submit ballots.

Multiple-Voting Detection: Each voter can only vote once. Multiple voting by any one voter is detected and identified.

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

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

Correctness of Tallied Result: Only verified ballots are counted and added to the final result.

End-to-End Voter Verifiable: Every voter is able to verify whether their vote is posted and counted correctly.

In order to make the system End-to-End (E2E) voter verifiable, we require each voter to generate proofs for each encrypted element of the ballot. These proofs are sent along together with the encrypted ballots. After submission, everything is made available publicly to all users. The proofs are generated using proof of partial knowledge (cf. Section III-B) and zero knowledge proof (cf. Section III-C), which means that the eligibility of each encrypted ballot can be verified by anyone.

Contributions of this Paper: We propose a new e-voting system, which is more flexible than the previous systems. It uses encryption to achieve verification of the integrity of the voting process and the validity of the ballots, at the same time maintaining confidentiality of the users. Our e-voting system is an E2E voter verifiable voting system. Each ballot is encrypted by the exponential ElGamal encryption algorithm and contains proofs used in verification. The proposed protocols achieve the following.

(1) The information of each encrypted ballot can be added to other ballots without decrypting any votes.

(2) Each encrypted ballot can be verified without revealing voting preferences. Only verified ballots are tallied for the final result.

(3) Voters can verify that their ballots are submitted correctly to the pool.

(4) Everyone is able to verify the eligibility of any voter's ballot without revealing voter's privacy.

(5) Each voter can verify the correctness of the final tallied result.

Structure of This Paper: Section II provides a literature review of recent work devoted to e-voting systems based on encryptions with homomorphic properties. Section IV presents our e-voting system. Security and performance analysis can be found in Sections V and VI, respectively. Section VII concludes this paper. For the convenience of the readers, Sections III-A, III-B and III-C describe preliminaries on the cryptographic models used in our e-voting system.

II. RELATED WORK

Homomorphic encryption has been used in online voting systems, for example, in [23]–[25]. The homomorphic property makes it possible to tally all encrypted ballots without decrypting them and accessing the content of any individual ballot.

Helios [26] is the first web-based voting system. It used ElGamal encryption to achieve open-audit voting. Helios did do not claim any cryptographic novelty apart from that fact that, assuming that there were enough auditors, even if all the authorities fully colluded to corrupt the system, they would be unable to counterfeit the election result without a high chance of being caught. However, the security of the Helios relies on the trust of all participants in the Helios server. The security level of Helios depends on a mix-net shuffling mechanism implemented by the server. It follows that a corrupt Helios server can attempt to shuffle submitted votes incorrectly or decrypt shuffled votes incorrectly. Further, the performance results reported in [26] show that the computation time is quite long. The verification and auditing process took more than three hours on a server and a complete audit took more than four hours on voter, even though there were only 2 questions in each vote and 500 voters in total. Thus, [26] provided the time achieved in only one experiment with a fixed number of voters.

All previous voting systems including [26] imposed several security assumptions required on their systems. Adida [26] made an assumption that there were several honest authorities and a central honest server. If any authority is compromised, they can attempt to shuffle the votes incorrectly. Likewise, a corrupt Helios server knows the usernames and passwords of all users, and can easily authenticate and cast ballots on behalf of users.

Several improvements to Helios were made in Helios 2.0 (see [27]), which was used in a real election (UCL election). Helio 2.0 could handle 25,000 potential voters. In the UCL election, 5000 participants registered and nearly 4000 voted in each round of the election. Helio 2.0 updated the open-audit mechanism, so that it could provide more evidence of the counters' works to all voters. However, their proposed approach distributed the key-generation and decryption code among a few trusted members of the election commission. This required the trustees to be technically savvy and honest. There is no indication of the running time of any new experiments in [27].

The security assumptions in [27] were the same as in [26]. In particular, a compromised server in Helios 1.0 and 2.0 could subvert all data in the ballot box near the end of the election day.

A multi-authority e-voting system introduced in [28] applied the distributed ElGamal DSA with an inherited additive homomorphism property. In addition to the authorities and the voters in the election, a trusted third party was introduced and was used to distribute the shared secret key among the multiple authorities. The proposed system became receipt-free, because the encryption of each ballot now is done by the trusted third party. This means that the voters could not prove how they voted by using their encrypted vote, and the authorities of the election could not learn the content of each vote from its encryption. However, the drawback of the system was that the third party could collude with any of the authorities and together they could recover the contents of submitted votes, which would violate the privacy of voters. The paper [28] assumed that there was a trusted third party involved in the scheme to distribute the shared secret key among the authorities. There was no indication of the running time or performance analysis of any new experiments in [28].

Cobra [29] was the first coercion-resistant system. It was a proof-of-concept voting system offering concurrent ballot authorization. The paper focused on the problem of coercion and vote selling. However, the proposed registration process could not be applied in practical elections, because it required each voter to register multiple times. In the performance analysis section, the authors provided concrete numbers only for a hypothetical election with 5 candidates, 10,000 registered voters, 20,000 submitted ballots and 3 trustees. This example took almost two hours of computing time on a fully parallel 8-core machine. A careful analysis of the running times of various steps of the algorithm was presented in [29], but only in one experiment with a fixed number of voters. In [29], an assumption was made that all authorities (trustees) should be honest. The trustees of an election authority engaged in a secure, universally verifiable protocol implementing a ballot authorization function.

Zeus [30] was a system developed based on Helio [26]. Zeus used the same workflow as Helio, but it provided more types of voting. The paper [30] assumed that the server should be always trusted. However, a potential security vulnerability of the system was that Zeus was run in a black box on a remote virtual machine. The lack of control and access to this black box virtual machine implementation created a possibility that without any control or awareness of any participants in the election process of how the black box operates, it could be subverted and confidential information could leak, or even the whole computing procedure could be incorrectly implemented. Thus, in the proposed implementation Zeus did not provide any reliable guarantee of anonymity and security to the users. Furthermore, Zeus turned out to be a computationally expensive system. It took approximately 65 minutes of computing time to handle 10000 votes by using a 16-core 2.26 GHz machine. This is the only new experiment with the running time indicated in [30].

A flexible e-voting system for online discussion forums was proposed in [31], where it was assumed that there is a trusted third party (registration server). This paper presented two diagrams with line graphs comparing only the time of the encryption operations, since it considered the special case of online voting conducted as a part of continuing online debates.

The most recent online voting system based on homomorphic encryption was proposed in [20]. It enables voters to cast their ballots by ranking all candidates. The main idea behind the system proposed in this paper is to convert each cast ballot into a square matrix, where the size of the matrix depends on the number of candidates. After that, each element in the matrix is encrypted by a verifiable homomorphic encryption algorithm. This approach, makes it possible to verify the eligibility of each submitted ballot without accessing the content of the ballot. Besides, the final result can be computed by using the additive homomorphic property without decrypting the cast votes.

The paper [20] assumes that there is more than one authority, and at least one of them is honest. The major weakness of that system is in the high cost of the computation on the voter side. It is explained by the number of exponentiations required being equal to the square of the number of candidates. It follows that the computation time can be expressed as $t_E \times {n_c}^2$, where t_E and n_c denote the computation time of a single encryption and the number of candidates, respectively (see Table 3). Moreover, the number of the proofs that have to be generated before submission of the vote is equal to ${n_c}^2$, because the verification is performed based on these proofs. Creating all the required proofs significantly increases the computation time of the system proposed in [20]. Furthermore, it increases the size of each submission, which is made up of ${n_c}^2$ ciphertexts plus ${n_c}^2$ proofs.

In addition to all the details mentioned above, to facilitate the comparison of the security assumptions and results of previous publications and the present paper, we include Table 1 with a brief outline of the results of experiments in the previous papers, and Table 2 with a summary of security assumptions used in the previous papers.

The present paper is an improved and extended version of the brief conference publication [20]. The authors have managed not only to expand the text by adding details and

TABLE 1. Comparison of the outcomes of experiments in previous papers.

Previous papers	Presentation of experimental outcomes
[27], [28].	These papers did not include the running time of any experiments.
[26], [30], [29].	These papers indicated the time achieved in only one experiment with a fixed number of voters.
[26], [27], [28], [29], [30].	These papers did not include any line graph, bar chart or histogram with experimental outcomes.
[31].	This paper presented two diagrams with line graphs comparing only the time of encryption operations.
[20].	This paper included two diagrams with line graphs comparing the performance comparison for elections with various numbers of voters.
The present paper.	It contains three diagrams with line graphs representing the new experiments and comparing them with the results of [20] for elections with various numbers of voters.

TABLE 2. Security assumptions in previous papers.

Previous papers	Security assumptions
[26], [27], [29]	These papers assume that all authorities (or trustees) and all servers are honest.
[28], [31]	These papers assume that an honest third party exists.
[30].	This paper does not require any authorities, but assumes that the main server is honest.
The present paper and [20].	This paper assumes there is more than one authority, and at least one of them is honest all the time.

TABLE 3. Notations used in the rest of this paper.

n_c:	number of candidates
n_v:	number of voters
n_a:	number of authorities
C_i:	i-th Candidate; $i \in [1, n_c]$
V_i:	i-th Voter; $i \in [1, n_v]$
A_i:	i-th Authority; $i \in [1, n_a]$
B_i:	the ballot submitted by V_i; $i \in [1, n_v]$
P:	total available points for a ballot
L_P	the number of bits of P (binary)
$B_{j,k}^{(i)}$:	the value on position (j, k) of B_i; $j \in [1, n_c]$, $k \in [1, L_P]$
$C_{j,k}^{(i)}$:	the encrypted value of $B_{j,k}^{(i)}$; $j \in [1, n_c]$, $k \in [1, L_{P_B}]$
Sig_{V_i}:	digital signature of V_i; $i \in [1, n_v]$
pk_{V_i}:	public key of V_i; $i \in [1, n_v]$
sk_{V_i}:	secret key of V_i; $i \in [1, n_v]$
pk_{A_i}:	public key of A_i; $i \in [1, n_a]$
sk_{A_i}:	secret key of A_i; $i \in [1, n_a]$
PK:	common public key for encrypting ballots
PZK{...}:	proof of zero knowledge
PPK{...}:	proof of partial knowledge

explanations, but also to improve the system proposed in [20] as follows.

- The new data structure of a binary matrix is introduced in the present paper. It was never considered previously. The paper [20] also used matrices, but they were different and had larger dimension.
- This innovation has significantly improved the running time of the system. The running time of our new algorithm has improved to $O(n_v \log n_v)$, as compared to $O(n_v^2)$ in [20], where n_v denotes the number of voters (see Table 3).
- In the present paper, several aspects of the election system are adjusted to handle more general situation. The new system proposed in the present paper has become applicable in a broader class of settings and can be used for elections with less strict requirements.

All algorithms have been improved, so that the present paper contains new "ballot generation algorithm", "verification algorithm" and "tallying algorithm". Here are further details on improvements made in the present paper.

(1) Improved ballot generation. In [20], a voter ranks all candidates to different positions and the ballot is converted to a square matrix with dimensions $n_c \times n_c$. In the present paper, a voter assigns arbitrary points to different candidates, and then each point is converted to its binary representation. This representation is stored in a new binary matrix, which is not square.

(2) Improved verification algorithm. In [20], the verification relied on the fact that the voters were not allowed to rank candidates to the same position in the ranked list. The system proposed in the present paper allows the voters to assign the same rank to different candidates. This is why a new verification procedure has been included in this paper.

(3) Improved tallying algorithm. In [20], the tallying was based only on the additive homomorphic property. In the present paper, the tallying process uses both the additive and multiplicative homomorphic properties of the cryptosystem.

(4) Improved performance. In [20], the square matrix had dimensions $n_c \times n_c$. In the present paper, a new binary square matrix is introduced to encode ballots. The number of rows of this matrix is n_c, but the number of columns never exceeds $2\log(n_c)$. This is why the running time of the main algorithm in the present paper has improved to $O(n_c \log n_c)$.

III. PRELIMINARIES ON CRYPTOGRAPHY

Following [5], in this section we introduce prerequisites for the underlying cryptographic algorithms, which are used as building blocks in our system.

A. ELGAMAL CRYPTOSYSTEM

ElGamal cryptosystem is very well known (cf. [2] and [5]). We assume that the cyclic group (G, q, g) is defined and there are n users in the system. Each i-th user has its own public key y_i and secret key x_i. The distributed ElGamal cryptosystem consists of the following algorithms.

Key Generation: A common public key

$$PK = \prod_{i=1}^{n} y_i = g^{x_1 + \cdots + x_n}$$

is used in the distributed ElGamal cryptosystem.

Encryption: To encrypt a plaintext message $m \in G$:

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

The encrypted message is $E(m) = (c_1, c_2)$.

Decryption: A common decryption key is not computed. Each user computes and broadcasts a partially decrypted value, and the final plaintext is revealed by combining all partially decrypted values. For the ciphertext (c_1, c_2), decryption proceeds as follows:

- Each i-th user computes ${c_1}^{x_i}$;
- All users broadcast commitment of computed values $H({c_1}^{x_i})$;
- Each i-th user broadcasts ${c_1}^{x_i}$ and checks if each $c_1^{x_i}$ matches with $H(c_1^{x_i})$;
- Each user computes $\frac{c_2}{\prod_{i=1}^{n} {c_1}^{x_i}} = \frac{c_2}{c_1^{x_1+\cdots+x_n}} = g^m$.

Finally, m can be revealed by computing a discrete logarithm.

Homomorphism. ElGamal encryption has an inherited homomorphic property [5], which allows multiplication and exponentiation to be performed on a set of ciphertexts without decrypting them, such as

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

B. PROOF OF PARTIAL KNOWLEDGE

Given a cyclic group G of a prime order q with a generator g. The secret key is x, and public key is $y = g^x$. The verifier can confirm that the ciphertext is either $E(m_1)$ or $E(m_2)$, but the verifier will never know which one is the true one [32], [33]. The ElGamal encryption algorithm (cf. Section III-A) is used in this protocol.

Prover

- generates a random number $r \in \mathbb{Z}_q$
- computes $E(m_1) = (c_1, c_2) = \{g^r, g^{m_1} \cdot y^r\}$
- generates random numbers $t, v_2, s_2 \in \mathbb{Z}_q$
- computes $T_0 = g^t$
- computes $T_1 = y^t$
- computes $T_2 = (g^{m_2 \cdot v_2} \cdot y^{s_2})/{c_2}^{v_2}$
- computes $v = hash(c_1 \| c_2 \| T_0 \| T_1 \| T_2)$
- computes $v_1 = v \oplus v_2$, where $\oplus$ denotes XOR.
- computes $s_1 = r \cdot v_1 + t$
- sends $c_1, c_2, T_0, T_1, T_2, v_1, v_2, s_1, s_2$ to **Verifier**

Verifier

- verifies $v_1 \oplus v_2 = hash(c_1 \| c_2 \| T_0 \| T_1 \| T_2)$
- verifies $g^{s_1} = T_0 \cdot {c_1}^{v_1}$
- verifies $y^{s_1} = T_1 \cdot (c_2/g^{m_1})^{v_1}$
- verifies $y^{s_2} = T_2 \cdot (c_2/g^{m_2})^{v_2}$

If all verification tests return true, the ciphertext can be considered as encryption value of m_1 or m_2. Therefore the verifier can only confirm that the ciphertext is either $E(m_1)$ or $E(m_2)$, but cannot determine the exact value of the plaintext with certainty.

C. PROOF OF ZERO KNOWLEDGE

This subsection contains prerequisites on the zero knowledge proof algorithm of [34] and [35]. Suppose that plaintext message is m. In the zero knowledge proof algorithm of [34] and [35] the prover computes $E(m)$ and proofs. The verifier can use the proofs only to verify that the ciphertext is encrypted from m, but cannot decrypt $E(m)$.

Further, suppose that the ElGamal cryptosystem is used, where G is a cyclic group G of a prime order q with a generator g, the secret key is x and the public key is y. Then $E(m) = (c_1, c_2) = (g^r, g^m \cdot y^r)$ (cf. Section III-A). To verify that the ciphertext (c_1, c_2) is a correct encryption for m, the algorithm proposes verifying that c_1 and $\frac{c_2}{g^m}$ have the same exponentiation. This zero knowledge proof is correct, because

$$c_1 = g^r \tag{1}$$

and

$$\frac{c_2}{g^m} = \frac{g^m \cdot y^r}{g^m} = y^r \tag{2}$$

Indeed, if the ciphertext (c_1, c_2) is $E(m)$, then (1) and (2) have the same exponent r. Otherwise, if (c_1, c_2) is different from $E(m)$, then it is obvious that (1) and (2) have different exponents.

Prover

- generates random number $r \in \mathbb{Z}_q$
- computes $E(m) = (c_1, c_2) = (g^r, g^m \cdot y^r)$
- generates random number $t \in \mathbb{Z}_q$
- computes $T_1 = g^t$
- computes $T_2 = y^t$
- computes $v = Hash(E(m) \| T_1 \| T_2)$
- computes $s = r \cdot v + t$
- sends c_1, c_2, T_1, T_2, v, s to **Verifier**

Verifier

- verifies if $g^s = {c_1}^v \cdot T_1$
- verifies if $y^s = (c_2/g^m)^v \cdot T_2$

If both verifications are passed, the verifier believes the prover's statement.

IV. PROPOSED E-VOTING SYSTEM

A. OVERVIEW AND NOTATION

This section contains an overview of our ranked choice e-voting system with illustrations and examples. In the present paper, we consider a general security assumption that there is more than one authority, and that at least one of them is honest all the time. This assumption is the same as in [20]. It is less restrictive than the diverse security assumptions imposed in other related works, as detailed in Section II (see Table 2). In particular, our system does not require a centralized, trusted server as in [26] and [27], or a trusted third party as in [29] and [30].

The basic idea of our voting system is to encrypt each ballot using the common public key of the distributed ElGamal cryptosystem. Since the exponential ElGamal satisfies the

additive homomorphic property, the encrypted ballots can be directly tallied. This procedure is also known as homomorphic tallying [36]. Finally, the tallied result can only be decrypted by collaboration of all authorities.

Our e-voting system consists of the following stages: initialization, registration, ballot casting, verification of voters, verification of ballots, tallying and result revealing.

B. ENTITIES

Here we list all entities that are involved in our e-voting system.

Voters: Each authorized voter can cast a ballot ranking all the candidates by assigning different points to different candidates according to their own preferences.

Candidates: Each candidate can be treated as a contestant in the election, and can receive different points from different submitted ballots. The candidate who received the largest total number of points is the winner of the election.

Authorities: There are multiple authorities in the election, who take responsibility for auditing the voting process by computing the common encryption key, verifying the identification of voters for each submission, verifying the eligibility of each ballot before tallying, revealing the winner of the election.

Public Bulletin Board: An insert-only bulletin board, which displays all information about the election, such as public keys, all submitted ballots and final tallied result. The content of the board can be viewed by all entities. However, no one is able to modify or delete existing data on it.

C. INITIALIZATION OF ELECTION

At the beginning of an election, all authorities have to generate a common encryption key (PK) that can be used by voters in order to encrypt each cast ballot before submission. Each authority (A_i) owns a key pair (public key pk_{A_i} and secret key sk_{A_i}), and (PK) is computed using the public keys (pk_{A_i}) of all tallying authorities (cf. Section III-A). Finally, the PK is posted on the public bulletin board, which can be used by all voters.

During the common key generation, each A_i has to broadcast their pk_{A_i}. In order to prevent adversaries from replacing any public key of an authority, hash values of pk_{A_i} must be broadcast as a commitment before broadcasting the pk_{A_i}. In cryptography, the key commitment is designed such that any party cannot change the value or statement after all parties have committed to it. In our system, PK is computed using all pk_{A_i}, where a commitment means each A_i agreed to contribute their public key, which cannot be changed later. If any A_i modifies the value, the other authorities can identify this. The common key generation only commences when all the broadcast keys are authorized.

Note that the total available points for a ballot (P) must be confirmed before the election starts, the value is decided based on the number of candidates (n_c). For example, when there are 3 candidates, the value of P could be 5 or 8, as long as P is greater than n_c. Once the value of P is confirmed, all voters have to agree and use it when casting their ballots. More explanations are given in Section IV-E.

D. REGISTRATION OF THE VOTERS

In order to register with our e-voting system, each voter (V_i) must present their valid ID (e.g. driver licence). Once a voter's identity has been verified, he/she generates a signature key pair, which consists of a public key (pk_{V_i}) and a private key (sk_{V_i}). The pk_{V_i} is uploaded to the public bulletin board, and the sk_{V_i} is kept secret by the voter V_i.

Once a voter has completed the registration, their identity and the corresponding public key can be found on the public bulletin board. Our system requires each voter to sign their ballot using Digital Signature Algorithm (DSA), where the sk_{V_i} of voter is used to sign the voter's submission and published pk_{V_i} can be used to verify their signature Sig_{V_i}.

E. BALLOT CASTING

Our e-voting system allows voters to rank all candidates based on their personal preferences. Each voter can assign different numbers of points to candidates, and the winner is the candidate who receives the largest total number of points.

Voters are allowed to assign any points to any candidate. The only restriction is that the total number of assigned points must be equal to the total available points (P). To illustrate, here we look at an example where there are 3 candidates and $P = 6$. Then all options Fig. 1(a), Fig. 1(b) or Fig. 1(c) are acceptable, because $2+2+2 = 6$ for Fig. 1(a), $0+0+6 = 6$ for Fig. 1(b), and $1 + 3 + 2 = 6$ for Fig. 1(c).

After that, all assigned points are converted into binary, and the content of the cast ballot is treated as a matrix. The size of the matrix is $n_c \times L_P$, where we use n_c to denote the number of candidates, and we use L_P to denote the number of bits of P (binary). For example, the ballot illustrated in Fig. 1(c) is converted to its binary version in Fig. 2(b), where $P = 6$ and $L_P = 3$.

Once the ballot (B_i) is converted into its binary version, such as (b) of Fig. 2, the content must be encrypted (using PK) before submission. In our system, each binary bit of B_i is encrypted individually, as in Fig. 2(c), where $B^{(i)}_{j,k}$ denotes the binary bit (0 or 1) on position (j, k) of the ballot, and $C^{(i)}_{j,k}$ denotes the encrypted value of $B^{(i)}_{j,k}$, where $j \in [1, n_c]$ and $k \in [1, L_P]$.

Once the encrypted ballot ($E(B_i) = C^{(i)}_{1,1}, \cdots, C^{(i)}_{n_c,L_P}$) is ready, there are two types of proofs to be computed and sent with the encrypted ballot. The voter (prover) must convince everyone (authorities and other voters) that each $C^{(i)}_{j,k}$ is either $E(1)$ or $E(0)$ (cf. Algorithm 1) and that the number of total number of assigned points is equal to P (cf. Algorithm 2). To this end, our system uses proof of partial knowledge (cf. Section III-B) and proof of zero knowledge (cf. Section III-C) to generate proofs for each $C^{(i)}_{j,k}$ and the total number of assigned points, respectively, This allows authorities and other voters to verify $E(B_i)$ without revealing the content of B_i.

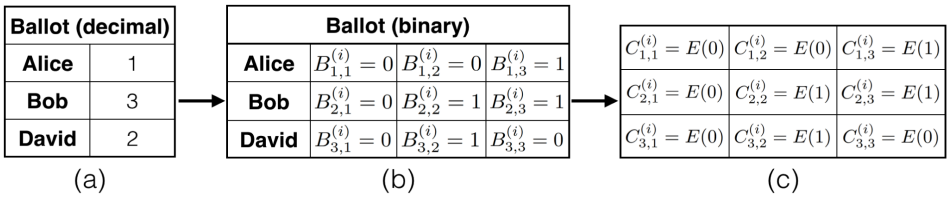

FIGURE 2. (a) is a ballot B_i cast by a voter V_i, (b) is a binary version of B_i, (c) is the encrypted version $E(B_i)$.

Algorithm 1 Generating Proofs for Each Element in an Encrypted Ballot

Input : $V_i, E(B_i), PK$
Output: $\text{PPKs}^{(i)}$
set $\text{PPKs} = \{\}$
for $j \leftarrow 1$ **to** n_c **do**
  for $k \leftarrow 1$ **to** L_P **do**
    PPK of $C_{j,k}^{(i)}$: prove $C_{j,k}^{(i)} = (c_1, c_2)$ is either $E(0)$ or $E(1)$ ▷ cf. Section III-B
      $\text{PPK}\{(C_{j,k}^{(i)}, T_0, T_1, T_2, v_1, v_2, s_1, s_2) :$
      $v_1 \oplus v_2 = H(C_{j,k}^{(i)} \| T_0 \| T_1 \| T_2), g^{s_1} = T_0 \cdot {c_1}^{v_1},$
      $PK^{s_1} = T_1 \cdot (c_2/g^1)^{v_1}, PK^{s_2} = T_2 \cdot (c_2/g^0)^{v_2}\}$
    $\text{PPKs}^{(i)} = \text{PPKs}^{(i)} \cup \text{PPK}$
  end
end
return $\text{PPKs}^{(i)}$

Algorithm 2 Generating Proofs for The Total Number of Assigned Points of an Encrypted Ballot

Input : $V_i, E(B_i), PK, P$
Output: $\text{PZK}^{(i)}$
set $\text{PZK} = \{\}$
set $P_{B_i} = E(0)$
for $j \leftarrow 1$ **to** n_c **do**
  set $t = 1$
  for $k \leftarrow L_P$ **to** 1 **do**
    $P_{B_i} = P_{B_i} \times (C_{j,k}^{(i)})^t$
    $t = t \times 2$
  end
end
generates proof of P_{B_i}: prove $P_{B_i} = (c_1, c_2)$ is $E(P)$ ▷ cf. Section III-C
$\text{PZK}^{(i)}\{(P_{B_i}, T_1, T_2, s) : v = hash(P_{B_i} \| T_1 \| T_2), g^s = {c_1}^v \cdot T_1, PK^s = (c_2/g^P)^v \cdot T_2\}$.
return $\text{PZK}^{(i)}$

Remark 1: The exponential ElGamal encryption is used, where $E(m) = (g^r, g^m \cdot y^r)$. For the convenience of readers, more details are given in Section III-A.

Remark 2: The proof of partial knowledge and the proof of zero knowledge in Algorithms 1 and 2 are denoted by PPK $\{a, b, \cdots : \alpha, \beta, \ldots\}$ and PZK$\{a, b, \cdots : \alpha, \beta, \ldots\}$, respectively. Here $a, b, \ldots$ are the proofs generated by the prover, and $\alpha, \beta, \ldots$ are the conditions satisfied by $a, b, \ldots$. The algorithm PPK verifies that a ciphertext $E(m)$ is the encryption of one of the multiple values $m_1, m_2, \ldots$ without decrypting it. Likewise, PZK verifies (without decryption) that the ciphertext $E(m)$ is the encryption of m. Both PPK and PZK never reveal the content of the ciphertext. The algorithms PPK and PZK are well-known. For the readers' convenience, more details on PPK and PZK are included in Sections III-B and III-C, respectively.

When the ballot has been cast, a digital signature (Sig_{V_i}) of the voter (V_i) is generated and sent with $E(B_i)$ and all proofs to the server. In this case, DSA is used, which means that Sig_{V_i} is generated by using sk_{V_i} and can be verified by using pk_{V_i}, as explained in the next subsection.

To summarize, each submission consists of the following: encrypted contents of a cast ballot ($E(B_i)$), proofs of each ciphertext ($\text{PPKs}^{(i)}$), proofs of total number of assigned points for the ballot ($\text{PZK}^{(i)}$) and a digital signature (Sig_{V_i}).

F. VERIFICATION OF EACH SUBMISSION

The contents of each submission are posted on the public bulletin board, including all encrypted values, all proofs and the digital signature. However, to prevent tallying any invalid ballot to the final result, the verification of each submission is a necessary and crucial step. It consists of the following three verifications: (1) verifying whether the sender of the submission is authorized, (2) verifying whether each encrypted element of the cast ballot ($E(B_i)$) is either $E(1)$ or $E(0)$, and (3) verifying whether the total number of assigned points of the $E(B_i)$ is equal to P.

In our system, each ballot is tallied to the final result only if its submission has been validated in the following three verification steps.

(1) Verify the Sender of Each Submission: In order to prevent unauthorized people impersonating authorized voters, we require each voter to sign their submission by using their private key (sk_{V_i}) based on the DSA algorithm. This means that the signature can be verified by using the voter's public key (pk_{V_i}).

Since the pk_{V_i} of every authorized voter is posted on the public bulletin board once he/she is successfully registered, it follows that anyone can verify whether a subsequent submission is sent by an authorized voter or not. To this end it suffices to verify its signature. For example, if $VerifySignature(Sig_{V_i}, pk_{V_i})$ is true, then the identification of the sender holds true and the sender is an authorized user; otherwise, the submission is discarded.

(2) Verify Each Encrypted Element of the Cast Ballot: Each ballot is treated as a binary matrix during ballot casting, as in

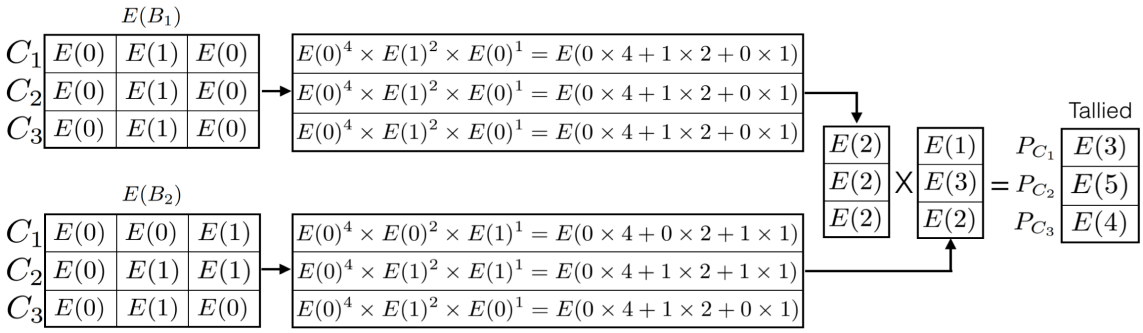

FIGURE 3. An illustration of tallying two encrypted ballots.

Fig. 2(b). It means that each element of a valid submitted ballot must be either $E(1)$ or $E(0)$, as in Fig.2(c).

Our system generates proofs for each voter submission based on the well-known proof of partial knowledge protocol (cf. Section III-B). This proof can be used to verify each encrypted element of the ballot without decryption. The verification of each element consists of two statements, both of which have to be confirmed as true. If either of these two statements is not true, then the whole ballot $E(B_i)$ cannot be counted in the final result. Only if all elements of the $E(B_i)$ are confirmed as valid, then $E(B_i)$ is considered as valid.

(3) Verify the Total Number of Assigned Points of the Cast Ballot: The total available points (P) of a ballot has been confirmed before the election. It is necessary to verify that the total number of assigned points for each ballot are equal to P in order to prevent a voter assigning more points to their ballot. In our system, we require each voter to generate proof for the total number of assigned points for their ballot based on zero knowledge proof procedure explained in Section III-C.

In this case, anyone is able to compute the total assigned points of any submitted ballot according to lines 1 to 8 of Algorithm 2. Then the self-computed value can be verified by using the voter-generated proof without decrypting anything. For convenience of the readers, more details on zero knowledge proof procedure are given in Section III-C.

G. TALLYING ALL VALID BALLOTS

In our system, each cast ballot is encrypted by using ElGamal encryption algorithm, which allows all encrypted ballots to be tallied directly without decrypting the contents of any ballot (cf. Section III-A).

The tallying is performed on each row of the binary ballot matrix, because each row denotes the received points of a particular candidate. In this case, we use P_{C_i} to denote the final tallied result for Candidate C_i, where $i \in [1, n_c]$. In order to tally P_{C_i} for C_i, the assigned points of each encrypted ballot are converted to decimal values by doing the exponentiation computation (cf. Section III-A). Once all encrypted ballots are converted as encrypted decimal ballots, the final result can be tallied based on different rows of all ballots. An illustration of tallying two encrypted ballots $E(B_1)$ and $E(B_2)$ is shown in Fig. 3 where there are 3 candidates C_1, C_2 and C_3.

The procedure of tallying all valid encrypted ballots is described in Algorithm 3.

Algorithm 3 Tally All Valid Encrypted Ballots.

Input : all encrypted ballots $E(B_1), \cdots, E(B_{n_v})$
Output: $P_{C_1}, \cdots, P_{C_{n_c}}$
1 **for** $j \leftarrow 1$ **to** n_c **do**
2 set $P_{C_j} = E(0)$
3 **end**
4 **for** $i \leftarrow 1$ **to** n_v **do**
5 **for** $j \leftarrow 1$ **to** n_c **do**
6 set $t = 1$
7 **for** $k \leftarrow L_P$ **to** 1 **do**
8 $P_{C_j} = P_{C_j} \times (C_{j,k}^{(i)})^t$ ▷ cf. Section III-A
9 $t = t \times 2$
10 **end**
11 **end**
12 **end**
13 **return** $P_{C_1}, \cdots, P_{C_{n_c}}$

H. RESULT REVEALING

Since the final tallied results $(P_{C_1}, \cdots, P_{C_{n_c}})$ for all candidates remain stored as ciphertexts, all of them have to be decrypted before publication. According to the distributed ElGamal cryptosystem (cf. Section III-A), the decryption procedure can only be done by collaboration of all authorities $(A_1, \cdots, A_{n_a})$, which requires each A_i to compute a partially decrypted value (cf. Section III-A), such as $c_1^{sk_{A_i}}$ and broadcast it to the others. In order to prevent a compromised A_i from any bad actions, such as somehow computing the value incorrectly, our system requires each A_i has to generate proof of zero knowledge (cf. Section III-C) in order to prove that the broadcast value is computed correctly, such as the exponent of the broadcast value (e.g. $c_1^{sk_{A_i}}$) must be the same as the exponent of its public key (e.g. $pk_{A_i} = g^{sk_{A_i}}$).

After decryption, the tallied result of each candidate must be revealed (cf. Section III-A) and published on the bulletin board.

V. SECURITY ANALYSIS

This section is devoted to a theoretical security analysis of our system. Note that none of the previous related papers provided a formal security model. They only included a description and an informal security discussion of their systems: see Section II. Our system relies on the ElGamal cryptosystem and several basic cryptographic protocols, which are presented in Section III and have reliable published proofs of their security. This is why here we include brief self-contained proofs of several theorems, which demonstrate that our proposed system fulfils all the security requirements.

A. ELIGIBILITY OF VOTER

Theorem 1: If the digital signature algorithm (DSA) is unforgettable, no one is able to submit a ballot by impersonating another voter.

Proof: In order to prevent adversaries from casting ballots by impersonating authenticated voters, we use a digital signature algorithm (DSA), which requires each voter V_i to have a key pair (public key pk_{V_i} and private key sk_{V_i}). The key pair is generated only if a voter is successfully verified during the registration. Then the pk_{V_i} of each verified voter is posted on the public bulletin board, and the voter is responsible for keeping their private key secret.

Once the election starts, each authorized voter signs their cast ballot by using their sk_{V_i}, and submits the encrypted ballot $E(B_{V_i}, PK)$ (which indicates the cast ballot B_{V_i} by voter V_i is encrypted using the common public key PK) along with their signature Sig_{V_i} to the public bulletin board, such as

$$\{E(B_{V_1}, PK), \text{corresponding proofs}, Sig_{V_1}\}$$

on the bulletin board. Others are able to verify the eligibility of each submission by verifying the Sig_{V_i} using the corresponding pk_{V_i} of V_i, where all pk_{V_i} of the successfully registered voters should be published on the public bulletin board.

In our protocols, no sk_{V_i} of voters is ever transferred. This means that only the voters themselves know their private keys. Therefore, no one is able to fake a voter's signature without the private key, and an adversary cannot submit a ballot by impersonating an authorized voter. □

B. MULTIPLE-VOTING DETECTION

Theorem 2: Only one submission from each voter can be stored on the server.

Proof: In our voting system, only the content of a submitted ballot is encrypted, the identification of the voter is in plaintext and can be viewed by everyone. For instance, in the extended previous example

$$V_1 \rightarrow \{E(B_{V_1}, PK), \text{corresponding proofs}, Sig_{V_1}\}$$

where the V_i is not encrypted.

Everyone is able to see the voter has submitted their ballot (e.g. voter's name or ID V_i is plaintext), but no one is able to discover how he/she voted.

Thus, multiple-voting detection is achieved by our system, because it is clear that it can always detect whether a voter has previously submitted a ballot: the voter id V_i for each submission is done on plaintext in the public bulletin board.

Furthermore, according to the requirements in the real-life case, our system can keep the first submission of each voter or replace the previous submission for each voter before the deadline. □

C. PRIVACY OF VOTERS

Since each ballot contains voting preferences of a voter, which can be treated as sensitive information, that must be protected all the time.

Theorem 3: If the ElGamal cryptosystem is semantically secure and at least one of authorities is honest, then the contents of ballots will not be revealed during ballot submission.

Proof: Every cast ballot is encrypted before submission. We use the distributed ElGamal cryptosystem (cf. Section III-A), which inherits the homomorphic property from the standard ElGamal system and is semantically secure as long as at least one of authorities is honest.

In our system, no one can reveal the contents of the ballots because of the following three reasons. First, all the submitted ballots remain in encrypted form as ciphertexts $E(B_{V_i}, PK)$ all the time. The homomorphic property makes it possible to add all $E(B_{V_i}, PK)$ without decrypting them. Second, there is no relationship between the ciphertexts $E(B_{V_i}, PK)$ and the corresponding plaintexts B_{V_i}, since the cryptosystem employed is probabilistic. It applies random numbers so that the ciphertext $E(B_{V_i}, PK)$ can take on different values even when the encryption $E(B_{V_i}, PK)$ is computed with the same B_{V_i} and PK. Third, the decryption must be done via collaboration of all authorities A_i as in the expression

$$D(E(B_{V_i}, PK), sk_{A_i}, \cdots, sk_{A_{na}}),$$

where B_{V_i} cannot be correctly computed if any of the sk_{A_i} is missing. As mentioned above, we assume that at least one of the authorities is honest and will not help others to do the decryption. Thus, the contents of each ballot remain secure. □

Theorem 4: If the partial knowledge proof protocol (cf. Section III-B) and the zero knowledge protocol (cf. Section III-C) o not reveal knowledge, contents of ballots will not be revealed during ballot verification.

Proof: In order to prevent counting invalid ballots to the final result, each encrypted ballot has to be verified before tallying. In our system, each encrypted ballot will be verified from 2 aspect: verify each encrypted element of the ballot, and verify the total number of assigned points of the ballot. Neither of the verification algorithms will reveal the voter's privacy, the analysis is shown as follows:

Verifying Each Element of a Ballot: For an encrypted ballot, each element in the ballot is encrypted individually. We also require the voter to generate proofs of each element, which is computed based on proofs of partial knowledge (cf. Section III-B).

The proof of each encrypted element $PPK\{...\}$ is generated based on the proof of partial knowledge, which consists of $T_0, T_1, T_2, v_1, v_2, s_1$ and s_2, where v_2 and s_2 are random values, and T_0, T_1, T_2, v_1, s_1 are computed by using random values t, v_2, s_2. Furthermore, the proof of partial knowledge is given in [26], which will not reveal the content of the original plaintext.

By using $PPK\{\cdots\}$, everyone can verify any element without decryption. The verifier(s) could only know if an element is either $E(0)$ or $E(1)$, but cannot determine the exactly value of the element is 0 or 1, and so the content of the ballot did not be revealed.

Only if all elements of a ballot are verified as valid, can the ballot be regarded as valid. In this case, we assume/assert the proof of zero knowledge is secure and will not reveal the information of the content. Thus, our protocol is secure and will not revealing voter privacy.

Verifying the Total Number of Assigned Points of a Ballot: The total number of assigned points of a ballot can be computed according to lines 1 through to 8 of Algorithm 2. We also require each voter to generate the proof for the total assigned points in order to convince the verifier(s) that the total number of assigned points is equal to the total available points.

According to the proof of knowledge $PZK\{...\}$ (cf. Section III-C), the self-computed total assigned points by verifier(s) can be verified by using voter-generated proofs without decrypting the self computed value.

The $PZK\{...\}$ consists of T_1, T_2, v and s, where all of these elements are generated by using another random number t, which is unrelated to $PZK\{...\}$ and $PPK\{...\}$. This means that nobody can derive any useful information from $PZK\{...\}$. The usability of $PZK\{...\}$ is proved in [20].

Moreover, even if the total number of assigned points of the ballot is decrypted, the voting preferences of the voter will not be disclosed and the voters' privacy will not be revealed because the total number of assigned points of each ballot should equal the total available points (public information), otherwise the ballot will be considered invalid and be discarded. □

None of the steps ''verify each element of the ballot'' and ''verify the total assigned points of the ballot'' require decryption. It follows that no one is able to reveal the content from the ciphertext without decryption because of the assumption that the ElGamal cryptosystem is secure. Furthermore, the decryption needs the collaboration of all authorities, and we assume that an honest authority will never commit bad actions. This means that decryption will never be executed in cases where it is not required. To sum up, the contents of the ballot and the privacy of voters remain secure in our verification protocols.

D. INTEGRITY OF BALLOTS

Theorem 5: Integrity of ballots is secured after submission.

Proof: We require voters to sign their ballots by using their private keys based on DSA, and we assume that all voters do not share their private keys with anyone else, to ensure that nobody can fake anyone else's signature.

The integrity of ballots can be protected in the following aspects: 1) Voters are able to save all contents (the ciphertexts) of the submissions as original receipts. Once the signature is verified, all contents are posted to the public bulletin board: voters can easily detect if their submissions are modified by comparing the original receipts and the published contents. 2) Once all contents of the submissions are posted on the public bulletin board, everyone can verify the integrity of any ballot by verifying the signature of the submission, because the signature is computed based on the content of the ballot (cf. Section IV-E). □

E. CORRECTNESS OF TALLIED RESULT

Theorem 6: Invalid ballots will be detected and discarded before tallying.

Proof: To prevent any invalid ballot from being tallied in the final result, each ballot has to undergo two verifications to check the total number of assigned points of a ballot and each element in the ballot.

Verifying Each Element in a Cast Ballot: In our protocol, the verifiers are able to verify that any element of a submitted ballot is either $E(0)$ or $E(1)$, which is used and proved in [26] and [27] . Once the ballot is verified as consisting only of 0s or 1s in plaintext, it follows that the voter did not assign negative points to any candidate because the value (decimal) of a binary that contains only occurrences of the digits 1 and/or 0 must be greater than or equal to 0. Once all elements are verified as valid, all candidates of the ballot receive non-negative points.

Verifying the Total Number of Assigned Points in a Cast Ballot: An honest voter can assign any point to any candidate according to their personal preferences, but the sum of all assigned points P_{B_i} should be equal to the total available points P for a ballot, which is confirmed before the election begins.

In our system, the total number of assigned points of a ballot can be computed according to lines 1 to 8 of Algorithm 2, and verified by using voter-generated proof (cf. Section IV-F). Once the total assigned points is verified as valid, it follows that the voter did not assign excess points to the ballot, which is used and proved in [20]. □

To sum up, a ballot can be considered as valid and contribute to the final result if and only if 1) the voter did not assign negative points to any candidate and 2) the voter did not assign excess points for the ballot. Otherwise, the ballot will be discarded.

F. END-TO-END VOTER VERIFIABLE

Theorem 7: Voters are able to verify the integrity and eligibility of their ballots and the correctness of final tallied result.

Proof: In our system, all contents (encrypted ballot, all proofs and signature) of each submission are be posted on the

public bulletin board, where they can be accessed by anyone. We assume that the bulletin board is secure, and it is "append-only". Voters can use the contents published on the public bulletin board to verify the following. First, they can verify the integrity of their own submission, because in the ElGamal cryptosystem each ciphertext $(g^r, g^m \cdot y^r)$ can be verified, since g and y are public values, m is the voting message and each r is known to the corresponding voter. Thus, each voter can verify that the ciphertext is a correct encryption of the ballot. Second, all participants can verify the eligibility of submissions of other voters, because the "verification of each submission uses only encrypted submissions available to all users as part of our system (see Section IV-F).

Furthermore, the final tallied result (ciphertext) can also be computed by anyone, because all encrypted ballots have to be tallied based on additive and multiplicative homomorphic property of ElGamal cryptosystem (cf. Section III-A), which is also a publicly accessible algorithm. The final decryption is performed by the collaboration of all authorities, where each authority computes the partial decryption value by using their secret keys. In our system, we require that each authority A_i be able to convince other users that the computed value $(g^r)^{sk_{A_i}}$ is the partial decryption of ciphertext $(g^r, g^m \cdot y^r)$ using proof of zero knowledge, where the value has the same exponentiation as $g^{sk_{A_i}}$, the public key of the authority. □

VI. PERFORMANCE ANALYSIS

The analysis of performance of each processing step is presented in a separate subsection below. All tests were performed using a 1024-bit key (p and q are 1024-bit). All tests were performed on a laptop with the following specifications: 2.8GHz quad-core Intel Core i7 (Turbo Boost up to 4.0GHz) with 6MB shared L3 cache and with 16GB of 1600MHz DDR3L onboard memory. We used a high performing implementation from libgmp via the gmpy2 python module (https://gmpy2.readthedocs.io/en/latest/). We use t to denote The computation time of one exponentiation is denoted by t. For the computer used in our experiments, we have $t = 0.00012$ seconds.

ElGamal encryption requires two exponentiations, and ElGamal decryption requires one exponentiation (cf. Section III-A), where the division can be avoided by using an alternative method [18]. 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.

A. PERFORMANCE OF THE VOTER SIDE

On the client-side, each voter should cast a ballot and submit it to the server. In order to prevent the voting preferences of each ballot being revealed after submission, we require each voter to encrypt their cast ballots, where every element in the ballot must be encrypted.

In this case, we set $P = 2n_c$, so that the total available points P is equal to twice the number of candidates n_c. For example, if there are 10 candidates ($n_c = 10$) in the election, each voter has 20 available points ($P = 2n_c = 20$) for their cast ballot. Since all elements in the cast ballot have to be encrypted, the larger the number of candidates participating, the more encryption processing time is required and the larger the submission grows. Therefore, the performance of the voter side can be summarized in two aspects: total computation time and total submission size.

Total Computation Time: We use the well-known Digital Signature Algorithm (DSA) to sign each ballot before submission. The processing time of signing is approximately equal to the time of one exponentiation, that is t. Thus, according to Section IV-E, the total computation time for a voter can be presented as the total computation time of $E(B_i)$, $\text{PPKs}^{(i)}$, $\text{PZK}^{(i)}$ and Sig_{V_i}, which can be expressed as

$$2t \times n_c \times L_P + 5t \times n_c \times L_P + 2t + t.$$

In this experiment, we test the total time spent for encrypting one ballot in five rounds on the laptop, according to different numbers of candidates (n_c =3, 5, 10, 15, 20) and different total available points ($P = 2n_c = 6, 10, 20, 30, 40$). Hence the numbers of bits (L_P) for each available points are $L_P = 3, 4, 5, 6, 6$, respectively. The result is shown in Fig. 4.

The computation time presented in Fig. 4 does not include the thinking time of the voters. It represents only the computation time of the algorithm converting all plaintext ballots cast by the voters to ciphertext ballots and tallying them.

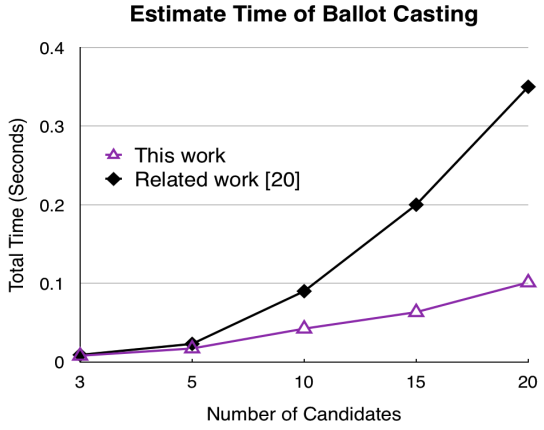

FIGURE 4. Estimate total time spent casting a ballot when the number of candidates (n_c) is 3, 5, 10, 15, 20, and the total available points (P) is 6, 10, 20, 30, 40. The performance of the system considered in [20] is also shown.

From the results in Fig. 4, we can see that the time cost for casting a ballot is approximately 0.1 seconds even if there are 20 candidates in the election.

Total Submission Size: In our system, the size of a ciphertext that is encrypted by ElGamal is 2048-bit (cf. Section III-A), and the size of proofs for each ciphertext is 7168-bit (cf. Section III-B). Furthermore, the size of proof of total number of assigned points is 5120-bit (cf. Section III-C) and the size of digital signature is 2048-bit. Thus, according to Section IV-E, the total submission size for a voter can be summarized as the total size of $E(B_i)$, $\text{PPKs}^{(i)}$, $\text{PZK}^{(i)}$ and Sig_{V_i}:

$$2048 \times n_c \times L_P + 7168 \times n_c \times L_P + 5120 + 2048.$$

In this experiment, we test the total submission size (including all encrypted elements, all proofs and a digital signature) for one cast ballot in five rounds on the laptop, according to different numbers of candidates ($n_c = 3, 5, 10, 15, 20$) and different total available points ($L_P = 3, 4, 5, 6, 6$). The result is shown in Fig. 5.

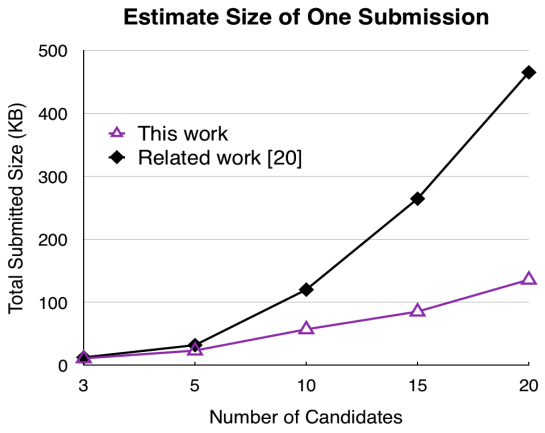

FIGURE 5. Estimate total size of one submission (including all encrypted values and all proofs) when n_c is 3, 5, 10, 15, 20, and P is 6, 10, 20, 30, 40. The performance of the system considered in [20] is also shown.

From these results, we found that the size of one submission is less than 150KB even for a 20-candidate ballot including all encrypted values and all proofs. It may be hard for the voters to compare too many candidates, and so much larger numbers of candidates seldom occur in elections. According to the Speedtest Global Index [36], in December 2017 the global average internet speed was equal to 21.25 Mbps download with 8.88 Mbps upload for mobile internet connections, and 40.71 Mbps download with 20.22 Mbps upload for fixed broadband connections, respectively. The Speedtest Global Index [36] evaluates and ranks the mobile and fixed internet connection speeds from around the world on a monthly basis. This demonstrates that the submission size of a cast ballot in our system is small enough to be submitted over the internet without delays for the voters. Note that the internet connection speed is rapidly increasing in the whole world.

Besides, in implementing practical applications the running time of our system can be improved further fine-tuning the system, for example, if the secure method for handling the encryption proposed in [37] is applied. Other efficient applications of encryption have been developed, for example, in [38]–[41].

B. PERFORMANCE OF THE SERVER SIDE

The performance of the server can be summarized from two aspects: verification of senders and verification of ballots. The verification of each sender is equivalent to verifying the digital signature of each submission, which does not cost much computation time. Therefore, we concentrate on the performance of ballot verification, which has two parts: the time of verifying each element of each ballot and time of verifying the total assigned points of each ballot. The total computation time on the server side is equal to

$$6t \times n_c \times L_P \times n_v + 4t \times n_v$$

Our experiments determined the total time spent on verifying all submitted ballots in five rounds according to different numbers of voters ($n_v = 1000, 2000, 4000, 7000, 10000$). In this experiment, we assume that the number of candidates is 10 ($n_c = 10$), and the total available points for a ballot is 20 ($P = 20$). Thus, the result is shown in Fig. 6.

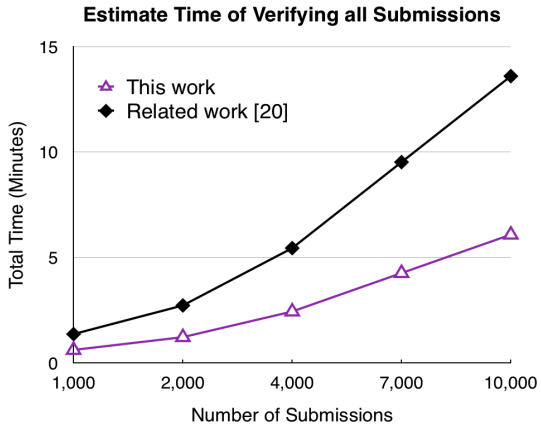

FIGURE 6. Estimate of the total time required for the verification of all ballots for 1000, 2000, 4000, 7000, 10000 voters, in the case of 10 candidates in the election. The performance of the system considered in [20] is also shown.

The results presented in Figures 4, 5, and 6 clearly demonstrate that our system has achieved a substantial improvement in the running time in comparison to [20]. It is easy to see this, because all the experimental results of [20] are also included as line graphs in Figures 4, 5, and 6.

The results of other previous papers are discussed and presented in Section II. First of all, note that a few of the previous papers did not include any experimental results, while the others included only the outcome of just one experiment with a fixed number of voters. These papers used different computers, some of which are no longer available for researchers. Keeping these circumstances in mind and comparing previous outcomes presented in Section II with the new results in Fig. 6, it is easily seen that our system has achieved substantial improvement in running time. For example, the experiment with 500 voters in [26] took more than 3 hours of computing time as compared to less than 2 minutes for our system with 1000 voters in Fig. 6. Likewise, the example in [29] took 2 hours for 10,000 voters, and the experiment in [30] with 10,000 voters took approximately 65 minutes of computing time, as compared to approximately 6 minutes for our system with 10,000 voters in Fig. 6.

From the results in Fig. 6, we found that the time spent of verifying 10, 000 ballots took approximately 6 minutes using the same laptop (the specifications of which are provided in the beginning of this section). Furthermore, in our protocols, all verifications can be done by an individual without collaboration with others. Thus, all submitted ballots can be

divided into multiple groups for simultaneous verification by different authorities. This can significantly reduce the time spent. For example, if there are 10 authorities in the election, then in practice the total verification time is approximately 10 times faster than the corresponding result presented in our diagrams, because in practice all authorities can work in parallel at the same time. Moreover, in the case of actual elections, it is natural to expect that cloud computing services will be available or multicore super computers can be used to execute the algorithms in parallel. This means that the running time will be further reduced.

VII. CONCLUSIONS

In this paper, we propose a secure voter verifiable e-voting system, which allows the voters to cast their ballots by assigning arbitrary numbers of points to different candidates. This means that the voters can assign equal points to different candidates, and they are also allowed to assign different points to different candidates. Our system incorporates the distributed ElGamal cryptosystem. Each cast ballot 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. Furthermore, the eligibility of voters and their submissions can be verified by anyone without the contents of the ballots being revealed. The security and performance analysis not only confirm the feasibility of our online voting system for practical elections, but also demonstrate that it has achieved significant improvements over other systems considered previously.

It is a limitation of our system is that we have to assume that at least one authority is honest, since otherwise the system is not secure. In future work, we plan to address this issue and potentially could consider further generalizations.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for comments and suggestions of improvements that have helped to revise this article.

REFERENCES

[1] W. Lu, A. L. Varna, and M. Wu, "Confidentiality-preserving image search: A comparative study between homomorphic encryption and distance-preserving randomization," *IEEE Access*, vol. 2, pp. 125–141, 2014.

[2] A. Ara, M. Al-Rodhaan, Y. Tian, and A. Al-Dhelaan, "A secure privacy-preserving data aggregation scheme based on bilinear ElGamal cryptosystem for remote health monitoring systems," *IEEE Access*, vol. 5, pp. 12601–12617, 2017.

[3] L. Chen, M. Lim, and Z. Fan, "A public key compression scheme for fully homomorphic encryption based on quadratic parameters with correction," *IEEE Access*, vol. 5, pp. 17692–17700, 2017.

[4] Z. Li, C. Ma, and D. Wang, "Towards multi-hop homomorphic identity-based proxy re-encryption via branching program," *IEEE Access*, vol. 5, pp. 16214–16228, 2017.

[5] X. Yi, R. Paulet, and E. Bertino, *Homomorphic Encryption and Applications*. New York, NY, USA: Springer, 2014.

[6] C. Esposito, A. Castiglione, B. Martini, and K.-K. R. Choo, "Cloud manufacturing: Security, privacy, and forensic concerns," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 16–22, Jul./Aug. 2016.

[7] O. Hasan, L. Brunie, and E. Bertino, "Preserving privacy of feedback providers in decentralized reputation systems," *Comput. Secur.*, vol. 31, no. 7, pp. 816–826, 2012.

[8] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo, "Protection of big data privacy," *IEEE Access*, vol. 4, pp. 1821–1834, 2016.

[9] R. Mendes and J. P. Vilela, "Privacy-preserving data mining: Methods, metrics, and applications," *IEEE Access*, vol. 5, pp. 10562–10582, 2017.

[10] M. Mukherjee *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[11] J. K. Liu, K. Liang, W. Susilo, J. Liu, and Y. Xiang, "Two-factor data security protection mechanism for cloud storage system," *IEEE Trans. Comput.*, vol. 65, no. 6, pp. 1992–2004, Jun. 2016.

[12] R. Neisse, G. Steri, D. Geneiatakis, and I. N. Fovino, "A privacy enforcing framework for Android applications," *Comput. Secur.*, vol. 62, pp. 257–277, Sep. 2016.

[13] V. D. Nguyen, Y.-W. Chow, and W. Susilo, "On the security of text-based 3D CAPTCHAs," *Comput. Secur.*, vol. 45, pp. 84–99, Sep. 2014.

[14] B. Rashidi, C. Fung, and E. Bertino, "Android resource usage risk assessment using hidden Markov model and online learning," *Comput. Secur.*, vol. 65, pp. 90–107, Mar. 2017.

[15] M. Gregory. (2016). *Electronic Voting May be Faster but Carries Security Risks*. [Online]. Available: http://www.theaustralian.com.au/business/technology/opinion/electronic-voting-may-be-faster-but-carries-security-risks/news-story/f0b6b44844214605e3860ef1887b2bb9

[16] J. Lavelle and D. Kozaki. (2016). *Electronic Voting has Advantages but Remains Vulnerable to Security, Software Problems*. [Online]. Available: http://www.abc.net.au/news/2016-07-11/electronic-voting-has-support-but-security-fears-remain/7587366

[17] S. J. Brams and P. C. Fishburn, "Going from theory to practice: The mixed success of approval voting," in *Handbook on Approval Voting* (Studies in Choice and Welfare). Springer-Verlag, 2005, pp. 19–37. [Online]. Available: https://link.springer.com/article/10.1007/s00355-005-0013-y

[18] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[19] K. E. Lauter, "Practical applications of homomorphic encryption," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2012, pp. 57–58.

[20] X. Yang *et al.*, "A verifiable ranked choice Internet voting system," in *Proc. Int. Conf. Web Inf. Syst. Eng. (WISE)*, 2017, pp. 490–501.

[21] J. Dreier, P. Lafourcade, and Y. Lakhnech, "Defining privacy for weighted votes, single and multi-voter coercion," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2012, pp. 451–468.

[22] A. O. Santin, R. G. Costa, and C. A. Maziero, "A three-ballot-based secure electronic voting system," *IEEE Security Privacy*, vol. 6, no. 3, pp. 14–21, May 2008.

[23] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, "Multi-authority secret-ballot elections with linear work," in *Advances in Cryptology—EUROCRYPT*. Zaragoza, Spain: Springer, 1996, pp. 72–83. [Online]. Available: http://www.springer.com/us/book/9783540611868

[24] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology—EUROCRYPT*. Bruges, Belgium: Springer, 2000, pp. 539–556. [Online]. Available: http://www.springer.com/us/book/9783540675174

[25] X. Yi and E. Okamoto, "Practical remote end-to-end voting scheme," in *Electronic Government and the Information Systems Perspective*. Toulouse, France: Springer, 2011, pp. 386–400. [Online]. Available: http://www.springer.com/us/book/9783642229602

[26] B. Adida, "Helios: Web-based open-audit voting," in *Proc. USENIX Secur. Symp.*, vol. 17. 2008, pp. 335–348.

[27] B. Adida, O. De Marneffe, O. Pereira, and J. J. Quisquater, "Electing a University president using open-audit voting: Analysis of real-world use of Helios," in *Proc. EVT/WOTE*, vol. 9. 2009, pp. 1–5. [Online]. Available: https://www.usenix.org/legacy/event/evtwote09/tech/

[28] A. A. Philip, S. A. Simon, and A. Oluremi, "A receipt-free multi-authority e-voting system," *Int. J. Comput. Appl.*, vol. 30, no. 6, pp. 15–23, 2011.

[29] A. Essex, J. Clark, and U. Hengartner, "Cobra: Toward concurrent ballot authorization for Internet voting," in *Proc. EVT/WOTE*, 2012, p. 3.

[30] G. Tsoukalas, K. Papadimitriou, P. Louridas, and P. Tsanakas, "From Helios to Zeus," *USENIX J. Election Technol. Syst.*, vol. 1, pp. 1–17, Aug. 2013.

[31] D. A. López-García, "A flexible e-voting scheme for debate tools," *Comput. Secur.*, vol. 56, pp. 50–62, Feb. 2016.

[32] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology—CRYPTO*. Santa Barbara, CA, USA: Springer, 1994, pp. 174–187. [Online]. Available: http://www.springer.com/us/book/9783540583332#otherversion=9783540486589
[33] B. Adida. (2012). *Helios v4*. [Online]. Available: http://documentation.heliosvoting.org/verification-specs/helios-v4
[34] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[35] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology—CRYPTO*. Santa Barbara, CA, USA: Springer, 1992, pp. 89–105. [Online]. Available: http://www.springer.com/us/book/9783540573401#otherversion=9783540480716
[36] V. Mateu, J. M. Miret, and F. Sebé, "A hybrid approach to vector-based homomorphic tallying remote voting," *Int. J. Info. Secur.*, vol. 15, no. 2, pp. 211–221, 2016.
[37] Y. Kim *et al.*, "On-demand bootstrapping mechanism for isolated cryptographic operations on commodity accelerators," *Comput. Secur.*, vol. 62, pp. 33–48, Sep. 2016.
[38] K. Bagheri, M.-R. Sadeghi, and T. Eghlidos, "An efficient public key encryption scheme based on QC-MDPC lattices," *IEEE Access*, vol. 5, pp. 25527–25541, 2017.
[39] A. Banerjee, M. Hasan, M. A. Rahman, and R. Chapagain, "CLOAK: A stream cipher based encryption protocol for mobile cloud computing," *IEEE Access*, vol. 5, pp. 17678–17691, 2017.
[40] R. R. Parmar, S. Roy, D. Bhattacharyya, S. K. Bandyopadhyay, and T.-H. Kim, "Large-scale encryption in the Hadoop environment: Challenges and solutions," *IEEE Access*, vol. 5, pp. 7156–7163, 2017.
[41] R. Xu, K. Morozov, Y. Yang, J. Zhou, and T. Takagi, "Efficient outsourcing of secure k-nearest neighbour query over encrypted database," *Comput. Secur.*, vol. 69, pp. 65–83, Aug. 2017.

XUECHAO YANG received the bachelor's degree in information technology from RMIT University in 2013 and the bachelor's degree (Hons.) in computer science in 2014. He is currently pursuing the Ph.D. degree in cyber security with RMIT University. His research interests include cryptosystems, homomorphism, and blockchain technology.

XUN YI is currently a Full Professor with the School of Science, RMIT University, Australia. He has published over 160 research papers in international journals, such as the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, the IEEE TRANSACTION ON VEHICULAR TECHNOLOGIES, the IEEE COMMUNICATION LETTERS, and the IEEE ELECTRONIC LETTERS, and conference proceedings. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and privacy-preserving data mining. He was a member of the program committees of over 40 international conferences. Recently, he has led several Discovery Projects of the Australian Research Council (ARC) and has been a member of the ARC College of Experts. Since 2014, he has been an Associate Editor of the IEEE TRANSACTION ON 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 currently a Principal Research Scientist with CSIRO Data61. 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. He has authored or co-authored over 150 publications to his credit. 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, the *ACM Computing Survey*, and the *ACM Transaction on Internet Technology*. 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.

ANDREI KELAREV was an Associate Professor with the University of Wisconsin and the University of Nebraska, USA, and a Senior Lecturer with the University of Tasmania, Australia. He is currently a Research Fellow with the School of Science, RMIT University, Australia. He is an author of two books and 198 journal articles. He is involved in the cyber security applications of machine learning and data mining. He was a Chief Investigator of a large Discovery Grant from the Australian Research Council.

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