cryptography

MDPI

Article

A Multi-Candidate Self-Tallying Voting Scheme Based on Smart Contracts

Xingan Dai [1], Xinquan Zhou [1], Dehua Zhou [1,*], Jinhan Zhong [1] and Chao Hong [2]

[1] College of Information Science and Technology, Jinan University, Guangzhou 510632, China; xingandx@gmail.com (X.D.); z13710288280@stu2021.jnu.edu.cn (X.Z.); zjhjz@stu.jnu.edu.cn (J.Z.)
[2] CSG Electric Power Research Institute Co., Ltd., Guangzhou 510663, China; hongchao8705@163.com
* Correspondence: tzhoudh@jnu.edu.cn

Abstract: In this paper, we propose a smart contract-based multi-candidate self-tallying voting scheme in order to guarantee the privacy of ballots in the case of multiple candidates. This scheme uses the ElGamal cryptosystem to ensure the security of the ballots, and combines it with a Distributed Encryption algorithm to make the voting scheme have self-tallying features, and guarantees the correctness of the intermediate data through zero-knowledge proofs. The experimental results show that the scheme improves the voting efficiency without compromising the security.

Keywords: distributed encryption algorithms; smart contract; zero-knowledge proof; ElGamal; self-tallying voting

1. Introduction

Electronic voting, facilitated by user-friendly online platforms and digital devices, represents a significant evolution in the conduct of voting procedures. This method offers substantial benefits, including reduced costs, heightened efficiency in vote tallying, and improved transparency. For example, Russia's adoption of electronic voting in local elections has significantly curtailed the time and potential for error associated with manual counting processes, leading to swifter announcements of electoral outcomes and a marked increase in the overall efficiency of the vote-tabulation process. During the COVID-19 pandemic, electronic voting has been instrumental in minimizing mass gatherings, allowing citizens to safely participate in the democratic process while observing social distancing protocols. Beyond symbolizing technological progress, electronic voting stands as a vital instrument for societal advancement. However, electronic voting must navigate a complex landscape of challenges and risks. In 2019, vulnerabilities within the Swiss Post's electronic voting system were uncovered, revealing the potential for electoral manipulation and raising concerns about the security and reliability of electronic voting systems. These incidents underscore the importance of robust security measures and the need for continuous scrutiny and improvement of electronic voting technologies.

Blockchain technology is an innovative distributed ledger system that ensures the security and integrity of data through decentralization, transparency, and tamper-proof records. Utilizing encryption and consensus mechanisms, blockchain enables the verification and recording of transactions without a central authority, thereby providing trust and efficiency across various domains. Initially designed to support cryptocurrencies such as Bitcoin, the application of this technology has rapidly expanded into other fields [1,2]. Advancing into 2022, China established a pioneering platform for innovation in blockchain and privacy computing technologies. By harnessing the immutable qualities of blockchain, this platform ensures the integrity and indisputability of the voting process, proposing an electronic voting system that incorporates blockchain technology. Blockchain's transparency and resistance to tampering make it an ideal candidate for addressing many of the security concerns associated with electronic voting. As blockchain technology continues to

check for updates

Citation: Dai, X.; Zhou, X.; Zhou, D.; Zhong, J.; Hong, C. A Multi-Candidate Self-Tallying Voting Scheme Based on Smart Contracts. *Cryptography* **2024**, *8*, 42. https://doi.org/10.3390/cryptography8030042

Academic Editor: Josef Pieprzyk

Received: 17 June 2024
Revised: 31 July 2024
Accepted: 2 September 2024
Published: 12 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

mature and expand its reach, an increasing cadre of researchers has turned their attention to electronic voting [3] schemes underpinned by smart contracts, offering a plethora of innovative solutions to the field.

These real-world implementations and developments justify the need for specific system requirements in electronic voting systems. Centralized versus distributed architectures, the necessity for fairness, robust encryption, and secrecy are all critical considerations. Centralized systems, while easier to manage and control, may present single points of failure or targets for attacks. In contrast, distributed systems, particularly those utilizing blockchain technology, offer enhanced security and transparency through decentralization but come with increased complexity and resource demands. Fairness in electronic voting ensures that no individual or group can unduly influence the outcome of an election, maintaining the integrity of the democratic process. This is particularly crucial in systems where votes are tallied electronically, as even minor discrepancies can significantly impact results. Encryption is paramount to protect voter privacy and prevent unauthorized access to voting data. Advanced cryptographic techniques, such as the ElGamal cryptosystem and zero-knowledge proofs, provide the necessary security framework to ensure that votes remain confidential and tamper-proof. Secrecy, or the assurance that a voter's choices remain private, is another fundamental requirement. Ensuring that each vote is both anonymous and untraceable prevents voter coercion and preserves the principle of a free and fair election.

Electronic voting holds substantial promise for improving the efficiency, accessibility, and integrity of electoral processes. However, realizing this potential requires addressing significant security and privacy challenges through advanced cryptographic techniques and robust system architectures. The ongoing developments and research in blockchain and privacy-enhancing technologies offer promising avenues for overcoming these challenges and advancing the state of electronic voting systems.

1.1. Related Work

Electronic voting schemes, initially proposed by Chaum [4] in 1981, have evolved to address the need for secure and efficient voting systems. These schemes are categorized based on encryption technologies into three main types.

The first type involves Mix-nets [4,5], which Chaum introduced. The system works by obfuscating multiple inputs through the Mix-net, resulting in outputs that are unlinkable to the original sender. However, this method requires numerous zero-knowledge proofs [6,7] to ensure the integrity of the ballots during the mixing process, which can be computationally intensive and impact system efficiency. The second type leverages homomorphic encryption [8], first proposed in 1997. This technology enables operations on encrypted data (ciphertexts) without the need for decryption. The ballots remain encrypted during counting, with decryption occurring post-counting. The security of the homomorphic encryption algorithm directly influences the level of ballot privacy and voter anonymity, with more secure algorithms providing higher privacy. The third type of system utilizes blind signatures [9,10] and ring signatures [11] to ensure voter anonymity. Blind signatures allow the counting center to verify the validity of a ballot without knowing its source, while the signing institution remains ignorant of the voter's choice. This system necessitates trust in the signer; to mitigate risks associated with untrusted signers, Liu et al. [12] introduced linkable ring signatures in 2005. This approach requires voter participation in the signature process, and by comparing linkable labels, authorities can verify voter participation. Voters signing ballots with linkable ring signatures must include the public keys of other voters to maintain anonymity.

However, such a voting protocol requires a centralized trusted party to control the entire voting process. Blockchain technology and smart contracts provide new ideas for electronic voting. These systems, as referenced in [13,14], utilize the properties of cryptocurrencies for voting. The ballots are not encrypted but are committed using random numbers to conceal the actual vote, with the sum of all random numbers equating to zero. A

centralized institution is typically required to manage the coordination between candidates and voters. As discussed in [15,16], the blockchain serves as a secure ballot box, ensuring the integrity of the voting process by synchronizing data across untrusted nodes. The use of Bitcoin as a public bulletin board is highlighted, although its limited scripting capabilities pose challenges for implementing complex voting schemes. Voting systems based on smart contracts, as noted in [17,18], are limited to supporting two candidates and a restricted number of voters. They necessitate all voters to participate before the results are finalized, and a voter's abstention can trigger a rerun of the entire voting process.

Building on blockchain technology, the concept of self-tallying electronic voting has been proposed, where voters perform the counting themselves. Kiayias [19] introduced this idea in 2002, emphasizing the potential for ballot privacy and indisputability, albeit with computational costs proportional to the number of voters. Groth et al. [20] later proposed a more efficient scheme, incorporating an anonymous broadcast channel for message confidentiality, albeit increasing the protocol's round complexity. Hao suggested a two-round public discussion scheme, which was critiqued by Khader et al. [5] for not meeting robustness and fairness standards. Lee [21] presented a national-level protocol dependent on a trusted third party. Mccorry [18] developed a self-tallying protocol on Ethereum, though it is limited to small-scale scenarios and lacks support for fractional voting. Li [22] introduced a decentralized self-counting scheme using zero-knowledge proofs to address voter abstention, but it is designed for only two candidates. Panja [23] proposed a ranking system for multiple candidates with public verifiability, yet it is not suited for large-scale voting. In 2020, Yang [24] proposed an online voting protocol that allows public verifiability without a trusted institution, utilizing an improved ElGamal cryptosystem for ballot encryption. However, this protocol does not accommodate voter abstention, which could lead to incorrect results.

1.2. Overview

This paper introduces a novel electronic voting scheme designed to address concerns about privacy, fairness, and integrity in current systems. It starts by discussing the importance of electronic voting and the need for improvement over existing methods like Mix-nets, homomorphic encryption, and blind signatures. The paper then presents a new multi-candidate voting scheme featuring ElGamal encryption, distributed encryption, zero-knowledge proofs, and a recovery mechanism to ensure vote privacy and process transparency while maintaining fairness and system integrity.

A thorough security analysis confirms the scheme's strengths through theoretical examination and simulation. Experimental results demonstrate the scheme's efficiency and effectiveness in multi-candidate elections, comparing favorably with existing systems. The discussion evaluates the scheme's practicality, application potential, and the challenges it may encounter. The conclusion reflects on the paper's contributions, summarizes the findings, and suggests directions for future research and scheme enhancement.

1.3. Our Contributions

The principal contribution of this manuscript is the introduction of an innovative multi-candidate self-tallying electronic voting scheme predicated on smart contracts, offering enhancements and optimizations over current electronic voting paradigms. Initially, this work addresses the limitations of existing models concerning candidate count and systemic equity by proposing a novel methodology grounded in the ElGamal cryptographic system and distributed encryption techniques. This approach harnesses the homomorphic attributes of the ElGamal system, facilitating the aggregation of votes without the need for decryption, thus safeguarding ballot privacy and ensuring the openness of the tallying process. Moreover, leveraging distributed encryption, the scheme articulated in this paper robustly secures ballot confidentiality and empowers voters to independently execute vote tallying. To surmount the fairness challenges inherent in the voting procedure, the incorporation of zero-knowledge-proof technology is presented. These proofs serve to

affirm the authenticity and propriety of each ballot, guaranteeing equitable consideration during the tallying phase. Thereby, the scheme circumvents potential inequities stemming from reliance on untrustworthy intermediaries, augmenting the overall trustworthiness of the voting framework. The paper introduces a resilience mechanism that empowers the final voter to attain the conclusive tally, notwithstanding any disclosure of keys by electoral administrators.

The performance of the proposed scheme within a multi-candidate voting context is substantiated through rigorous empirical analysis. The findings indicate that the scheme substantially ameliorates the efficiency of the voting system while upholding stringent security protocols, demonstrating adaptability to more extensive voting environments. Such enhancements render the proposed scheme markedly more viable and dependable for real-world implementation.

2. Preliminaries

In the process of constructing the self-tallying voting scheme for multiple candidates, this paper primarily utilizes the homomorphic properties of the ElGamal encryption system to ensure ballot privacy, employs distributed encryption to enable the self-tallying feature of the voting scheme, and uses zero-knowledge proofs to constrain participant behavior. This scheme includes the following polynomial-time algorithms:

2.1. Key Generation Algorithm KeyGen(k)

This algorithm takes as input a security parameter k, determines a large prime number p of k bits, and chooses a generator g of the cyclic group G. It then selects a random number $x \in \mathbb{Z}_p^*$. Here, G, p, and g are public parameters. The algorithm outputs the key pair for voter v_i.

$$\{pk_i, sk_i\} = \{g^x \bmod p, x\} \tag{1}$$

2.2. Encryption Algorithm Enc(pk, m)

This algorithm takes as input the plaintext $m \in G$ and the public key pk. It chooses a random number $r \in \mathbb{Z}_p^*$, and computes $c_1 = g^r \bmod p$ and $c_2 = g^m \times pk^r \bmod p$. The algorithm outputs the ciphertext c.

$$c = (c_1, c_2) = (g^r, g^m \times pk^r) \tag{2}$$

2.3. Decryption Algorithm Dec(c, sk)

This algorithm takes as input the ciphertext c and the corresponding private key sk, and outputs the corresponding plaintext of the ciphertext.

$$g^m = \frac{c_2}{c_1^{sk}} \bmod p \tag{3}$$

2.4. Signature Algorithm Sign(m, sk)

This algorithm takes as input the message m to be signed and the signer's private key sk. It chooses a random number $r \in \mathbb{Z}_p^*$, computes $R = g^r \bmod p$, and calculates the hash value $h = \mathrm{Hash}(m)$ of the message m. The algorithm outputs the signature σ.

$$\sigma = (R, s) = (g^r, \frac{h - xR}{r}) \tag{4}$$

2.5. Signature Verification Algorithm Verify(σ)

This algorithm takes as input the signature σ to be verified and the verifier's public key pk. It computes the hash value $h = \mathrm{Hash}(m)$ of the message m, and verifies the signature with the following equation:

$$g^h = pk^R \cdot R^s \tag{5}$$

If the equation holds, the validation is considered successful; otherwise, the validation fails.

2.6. Distributed Encryption Algorithm (L, Index)

Assume that G denotes a finite cyclic group of order prime q and g is a generator of the cyclic group G. The algorithm takes as input the set of public keys of all voters $L = \{g^{x_1}, g^{x_2}, \ldots, g^{x_n}\}$ and the index of one of the voters. The algorithm outputs the value y_{index} of the voter with the given index.

$$y_{\text{index}} = \frac{\sum_{j=1}^{\text{index}-1} g^{x_j}}{\sum_{j=\text{index}+1}^{n} g^{x_j}} \tag{6}$$

3. Self-Tallying Voting Scheme for Multiple Candidates

The basic idea of this system is that the election administrator deploys the voting contract by confirming the public parameters (such as the election's public key). Each voter can submit their own ballot through the smart contract, and each vote constitutes a transaction in the blockchain system. Each ballot will go through the verification of the smart contract. If the check executed in the smart contract determines that the vote sent by the voter has not passed the verification, the transaction will be rejected. After the consensus algorithm of the blockchain is mined, the vote is considered the final result.

3.1. Framework of the Self-Tallying Voting Scheme

The main participants in this voting scheme are the blockchain and the voting clients $v_i (i = 1, 2, \ldots, n)$. As shown in Figure 1, the entire voting scheme is implemented through the interaction between the blockchain and the clients. The main functions of each role in the system are as follows:

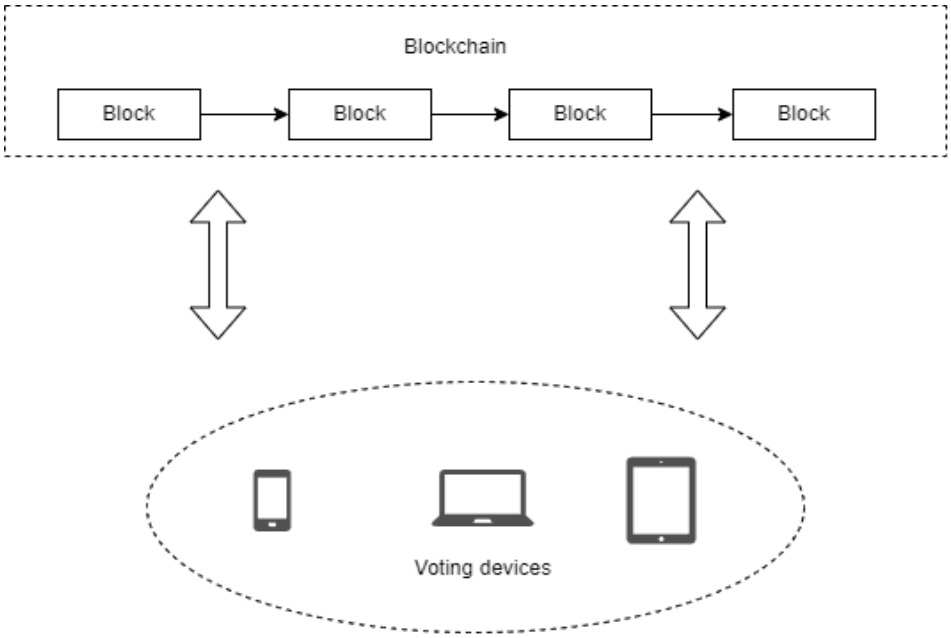


Figure 1. Framework for the self-tallying voting schemes.

The blockchain is a database that can only be appended and not modified. It can store data uploaded by clients through transactions. Each client needs to register and save its identity information on the blockchain for smart contracts and other users to view. It can also serve as a public bulletin board to save the voting information of clients, without the need for an additional trusted third party to perform the above duties.

Voting devices, i.e., all voters participating in the voting, generate their own public and private keys according to the public parameters in the blockchain and send them to the

blockchain to complete the registration. Afterward, they can use the public key generated by all registered client public keys to encrypt their own ballot information, ensuring the privacy of the ballot, before uploading it to the blockchain.

3.2. Self-Tallying Voting Scheme Model

The electoral scheme involves four main entities: the election administrator, the blockchain, the voters, and the candidates. Their roles are as follows:

Election Administrator: This individual sets up the election's security measures, manages voter registration, and ensures voters are eligible. They also create a public and private key pair for the election, placing the public key on the blockchain. The administrator is responsible for candidate enrollment, maintaining a candidate list, establishing voting rules, and deploying the voting smart contract. Once voting starts, these details cannot be changed.

Blockchain: Acts as an immutable ledger where data can be added but not altered. Both the election administrator and voters can send data to the blockchain, which is then added to the latest block, securing the integrity of the voting process. Once data are on the blockchain, they cannot be changed or removed.

Voters: Each voter has a public and private key. After their eligibility is confirmed, the election administrator adds the voter's public key to the blockchain. Voters use the smart contract to encrypt their votes with their private keys and send these encrypted votes to the blockchain, ensuring vote confidentiality and security.

Candidates: The election administrator also manages the candidate list. Candidates aim to win votes from the electorate, with the goal of accurately tallying the final results. This process prevents election outcomes from being affected by incomplete candidate lists.

Figure 2 illustrates the procedural flow involving the electorate, the election administrators, and the blockchain across the various stages of the electoral process.

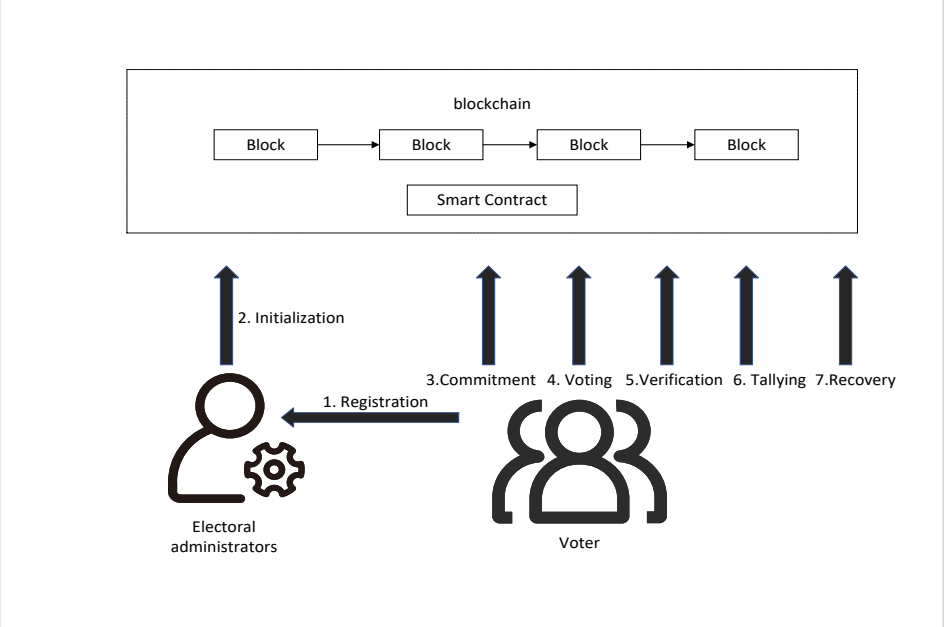


Figure 2. Workflow of the decentralized self-tallying scheme.

Electronic voting systems are designed with a series of stages to ensure a secure and fair election process. The process starts with the Registration Stage, where candidates generate key pairs and voters submit these public keys to the election administrator. This is followed by the Initialization Stage, where the election administrator deploys a smart contract on the blockchain, making its address publicly accessible. The Commitment Stage requires voters to commit to their candidate ballots, ensuring the protocol's fairness. During the Voting Stage, voters encrypt their ballots and submit them to the blockchain

for verification. The Verification Stage involves checking each ballot and ensuring all a voter's ballots meet the necessary conditions; successful verification leads to the tallying stage, while failure terminates the voting process. In the Tallying Stage, after all votes are verified, both voters and candidates can count the votes, with results being announced on the blockchain. Finally, the Recovery Stage is a fail-safe where, if the last voter does not vote, other voters can recover the committed ballot and ensure the vote is counted. This comprehensive approach guarantees that the voting system is transparent and secure.

3.3. Security Requirements

Designing a secure and reliable voting scheme to ensure the security of the election and voting process is crucial. To achieve this goal, the voting scheme needs to meet a series of security requirements to build a practical voting scheme.

Maximal Ballot Secrecy (MBS) [22]

In the following game, if any polynomial-time adversary $\mathcal{A}$ has a negligible advantage over the challenger $\mathcal{C}$, then we consider this self-tallying voting scheme to be MBS secure.

We formalize the security model, assuming that in the MBS voting scheme, a total of n voters participate in the voting, and at most n-2 voters are controlled by the adversary. Because if n-1 voters are controlled by the adversary, it is easy to obtain the voting information of the last voter from the final election result. The adversary can query the ballot commitment and the ballot of the controlled voter, and can also obtain the final voting result. In the subsequent challenge stage, the votes of two voters who are not controlled by the adversary for the candidate's ballot are {0} and {1}, respectively, and the adversary needs to distinguish which of these two ballots gives the content of {1}. The specific game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ is reflected in the specific interaction, and the specific security model is as follows:

- Initialization: There are n voters in the game process, and the adversary $\mathcal{A}$ declares the target voters v_a and v_b to be challenged, and the other voters are controlled by the adversary. The challenger $\mathcal{C}$ randomly selects a voter v_t from the target voters v_a and v_b, and the ballot of v_t is {1}.
- Setup: The challenger $\mathcal{C}$ will generate a pair of public and private keys for each voter through the key generation algorithm, and send all voters' public keys and the private keys of the controlled voters to adversary $\mathcal{A}$.
- Query: The adversary can choose any ballot for the controlled voter, and can query the challenger $\mathcal{C}$ based on this ballot, including the ballot commitment query and the voting query. $\mathcal{A}$ can ask for the commitment of a certain vote in the commitment query, generate the corresponding commitment and send it to $\mathcal{C}$ and record it in the commitment list; $\mathcal{A}$ can control the voting content of any voter in the voting query.
- Challenge: $\mathcal{C}$ generates corresponding challenge votes for the target voters v_a and v_b.
- Tallying: $\mathcal{A}$ calculates the final result of the vote based on the collected ciphertext votes.
- Guess: $\mathcal{A}$ outputs a guess result. That is, who gave the ballot of {1} among v_a and v_b.

In the above model, the reason for setting up two challenge ballots v_a and v_b, instead of one, is to prevent $\mathcal{A}$ from inferring the content of the challenge ballot from its known information. Specifically, the ballot content of the controlled voter is known to $\mathcal{A}$, and if there is only one challenge ballot when the final voting result is obtained, the adversary will have a non-negligible advantage to win in the guessing stage. The two challenge ballots have different values, because if both challenge tickets are {0} or {1}, and the adversary knows the ballot of the broken voter, then the adversary can easily infer from the final voting result. The content of the two identical ballots is {0} or {1}, and the advantage ε of winning the game is not negligible.

Definition 1. *If the probability of any polynomial-time adversary $\mathcal{A}$ winning in the game is negligible, then this scheme is MBS secure.*

$$\left| Pr[out = t] - \frac{1}{2} \right| \leq \varepsilon$$

Electronic voting systems must ensure several critical principles for a fair and secure election. Secrecy is essential, keeping all ballots confidential. Correctness guarantees that only valid votes are accurately counted. To prevent manipulation, double voting must be avoided, ensuring each voter casts only one vote. The system should be self-tallying, allowing for straightforward vote counting once all votes are in. Verifiability lets voters confirm their vote's inclusion and watch for any unfair practices. Fairness is maintained by addressing issues like voter abstention and the potential for the last voter to influence the outcome.

4. Construction Details

This section first introduces in detail the basic cryptographic algorithms used in the self-tallying voting scheme for multiple candidates, then details the overall workflow of the scheme and its specific implementation details, and finally analyzes the security of the voting scheme. Table 1 is the Symbols used by the program.

Table 1. Symbols used by the program.

Symbol	Specific Meaning
pk	Public key of the election administrator
sk	Private key of the election administrator
n_c	Total number of candidates
n_v	Total number of voters
v_i	The ith voter, $i = \{1, 2, \ldots, n_v\}$
$x_{i,j}$	The private key generated by the ith voter for voting for the jth candidate
$y_{i,j}$	The value computed by the ith voter using all the public keys for the jth candidate through Distributed Encryption
$m_{i,j}$	The vote of the ith voter for the jth candidate
$C_{i,j}$	Ballot commitment generated by the ith voter for $m_{i,j}$
$Y_{i,j}$	Public key product computed by the ith voter for the jth candidate
ZKP{…}	Zero Knowledge Proofs

4.1. Specific Construction of the Scheme

Before all voters generate keys and begin registration, the cyclic group (G, p, g) used by the system needs to be predefined. The election administrator then uses the KeyGen(k) algorithm to generate a pair of public and private keys (pk, sk). The administrator uploads their public key pk to the blockchain through a transaction, enabling all voters to access it. In this scheme, each voter can support only one candidate (represented by 1), while votes for other candidates are opposition votes (represented by 0). The specific process of this scheme is described in the following seven stages:

4.1.1. Registration Stage

Each voter uses the KeyGen(k) algorithm to generate a public and private key pair $\{pk_{i,j}, sk_{i,j}\} = \{g^{x_{i,j}}, x_{i,j}\}$ for each candidate. After generating these keys, the voter registers with the election administrator, sending the n_c public keys $\{pk_{i,1}, pk_{i,2}, \ldots, pk_{i,n_c}\}$ to the administrator. Once the voter's eligibility is verified, their public keys are added to the list of eligible voters and recorded on the blockchain. After all eligible voters have registered or the registration period set by the election administrator ends, the administrator deploys the voting contract.

4.1.2. Initialization Stage

The election administrator deploys the smart contract on the blockchain and publishes the contract address, allowing all voters to verify and access it.

4.1.3. Commitment Stage

Before voting, each voter v_i obtains each public key $g^{x_{i,j}}$ from the smart contract, where $i \in [1, n_v]$ and $j \in [1, n_c]$. Voter v_i generates a commitment for each candidate's ballot $m_{i,j}$ to ensure it will be used for their formal vote later. The process for generating the commitment is as follows: v_i chooses a random number $\rho_{i,j}$ for the j-th candidate, where $j \in [1, n_c]$, and sends $\beta_{i,j} = g^{\rho_{i,j}}$ to the smart contract, making it viewable to all voters. After this, v_i generates the ballot commitment $C_{i,j} = g^{m_{i,j}} \times (Y_{i,j})^{\rho_{i,j}}$, where $Y_{i,j} = \prod_{k=1,k\neq i}^{n_v} g^{x_{i,j}}$, and sends this commitment $C_{i,j}$ to the smart contract. To ensure that v_i generates the ballot commitment truthfully, v_i provides the corresponding zero-knowledge proof ZKP$(C_{i,j})$. The specific proof generation process is illustrated in Figure 3.

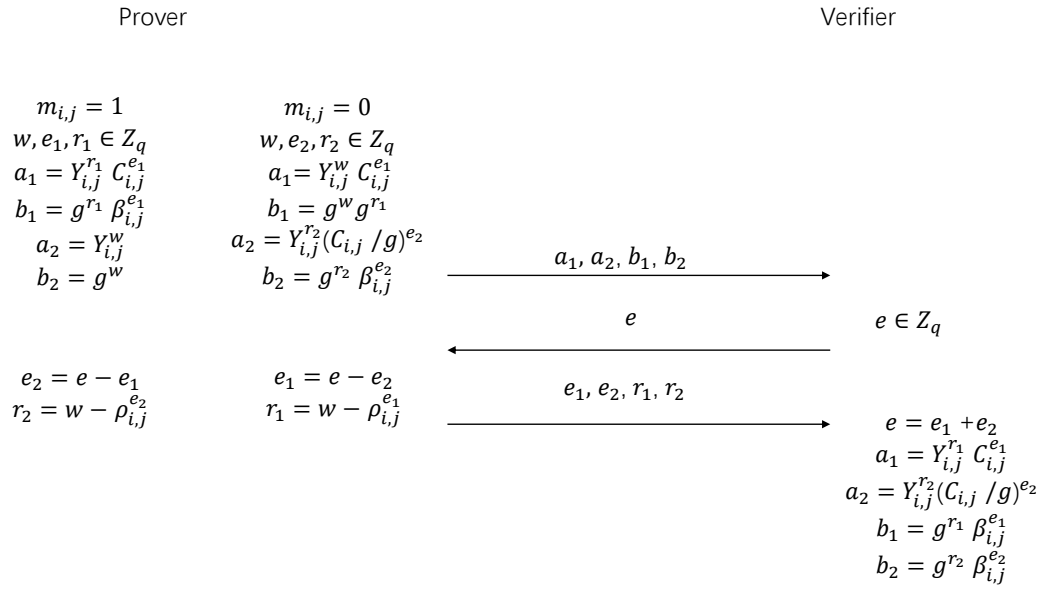


Figure 3. The generation process of $ZKP(C_{i,j})$.

Figure 3 illustrates the generation process of a Zero-Knowledge Proof (ZKP), which is used to prove that a voter v_i has indeed generated a valid commitment $C_{i,j}$ for their vote on candidate j, without revealing the specific content of the vote $m_{i,j}$.

Zero-Knowledge Proof Generation: To prove that the generated commitment is genuine, the voter v_i must provide a Zero-Knowledge Proof ZKP$(C_{i,j})$. This proof involves:

- W, e_1, r_1 and W, e_2, r_2 are values used in the proof process, all elements of the finite field Z_q.
- a_1, a_2, b_1, b_2 are values computed by the Prover based on their vote and random numbers.
- e is a computed value used in the proof.

Proof Process: The Prover presents e_1 and e_2, components of e, without revealing e itself, maintaining the anonymity of the vote. The Verifier uses these values to verify that the Prover indeed holds a valid commitment, without knowing the specific content of the commitment.

Verification Process: The Verifier uses the values provided by the Prover to verify the correctness of the proof. If the verification is successful, it indicates that the Prover has generated the commitment according to the specified rules without disclosing any information about their voting choice.

4.1.4. Voting Stage

Voters cast their votes according to their preferences. This stage consists of three steps: pre-computation, encrypted ballot, and proof generation.

During the pre-computation process, each of the n_v voters publishes their public keys for voting on the blockchain. Consequently, all voters can see these $n_v n_c$ public keys $g^{x_{i,j}}$. In this phase, voter v_i generates a distributed encryption value $y_{i,j}$ for the jth candidate using the DistributedEncryption algorithm, where $L = \{g^{x_{1,j}}, g^{x_{2,j}}, \ldots, g^{x_{n_v,j}}\}$ and $j \in \{1, 2, \ldots, n_c\}$. Voter v_i produces n_c values $y_{i,1}, y_{i,2}, \ldots, y_{i,n_c}$, each specifically used for voting for the respective candidate.

In the encrypted ballot process, voter v_i casts a vote for the candidate they support. If v_i supports the jth candidate, then $m_{i,j} = 1$ and $m_{i,k} = 0$ for $k \in \{1, \ldots, j-1, j+1, \ldots, n_c\}$. Here, $m_{i,j}$ represents the voting data of the ith voter for the jth candidate. These ballots are private, so they must be encrypted before being uploaded. Each $m_{i,j}$ ballot is encrypted twice: first using the election administrator's public key for ElGamal encryption, and second using the pre-computed $y_{i,j}$ for distributed encryption.

The voter's ballot is first encrypted using the election administrator's public key pk, resulting in the ciphertext through the encryption algorithm $\text{Enc}\,(pk, m_{i,j}) = (g^r, g^{m_{i,j}} pk^r)$. To ensure the scheme's distributed encryption characteristic, the first part g^r of this ciphertext is further transformed by encrypting it again with the voter's private key $x_{i,j}$ as follows:

$$g^r \to y_{i,j}^{x_{i,j}} \times g^r \tag{7}$$

Here, $y_{i,j}$ is pre-computed. Combining the two encryption processes, the encryption formula is as follows:

$$E(pk, y_{i,j}, x_{i,j}, m_{i,j}) = (y_{i,j}^{x_{i,j}} \times g^r, g^{m_{i,j}} \times pk^r) \tag{8}$$

where $j \in \{1, 2, \ldots, n_c\}$. Thus, the encrypted ballots of voter v_i for all candidates are represented as the following matrix:

$$Ballot_{v_i} = \begin{bmatrix} E(pk, y_{i,1}, x_{i,1}, m_{i,1}) \\ E(pk, y_{i,2}, x_{i,2}, m_{i,2}) \\ \vdots \\ E(pk, y_{i,n_c}, x_{i,n_c}, m_{i,n_c}) \end{bmatrix} \tag{9}$$

In the proof generation phase, to ensure the voter has truthfully carried out the ciphertext generation process, each voter must provide two zero-knowledge proofs alongside the ciphertext. These proofs allow anyone to verify the validity of each vote without decrypting the ciphertext, thereby protecting voter privacy while confirming adherence to the protocol. The specific zero-knowledge proof creation process is as follows:

ZKP($x_{i,j}$): This zero-knowledge proof ensures that the ciphertext for the jth candidate's ballot was generated using the voter's private key $x_{i,j}$. Below is the proof generation and verification process for $Ballot_{v_i}$. Assume the ciphertext of v_i for the jth candidate is $E(pk, y_{i,j}, x_{i,j}, m_{i,j}) = (c_1, c_2) = (y_{i,j}^{x_{i,j}} \times g^r, g^{m_{i,j}} \times pk^r)$, and this ciphertext is known to both the prover and the verifier.

In this process, $y_{i,j}$ and $g^{x_{i,j}}$ are public parameters. The verifier checks whether the right-hand equation holds. If it does, the prover's generated ciphertext is trustworthy. The detailed steps are given in Algorithm 1.

Algorithm 1 The generation process of ZKP($x_{i,j}$)

Parameters: random number $a, b \in Z_q$.
Prover:
(1) Prover Compute $A = y_{i,j} \times g^b$, where $y_{i,j}$ is the distributed encryption value precomputed by voter v_i for candidate j.
(2) Generate $B = g^a$, where g is the cyclic group generator.
(3) Calculate the hash $h = \text{Hash}(A||B)$, concatenating A and B.
(4) Compute $C = X_{i,j} \cdot h + a$ and $D = r \cdot h + b$, using the voter's public key $X_{i,j}$ and another random number r.
(5) The Prover sends A, B, C, D to the Verifier without revealing a and b.
Verifier:
The Verifier checks the equation $y_{i,j}^c \cdot g^D = A \cdot C_1^h, g^c = B \cdot (g^{x_{i,j}})^h$ to verify the proof.
If the equation holds, the proof is valid, confirming the correct use of the private key. Otherwise, the proof is invalid.
Zero-Knowledge Feature: The proof process does not reveal the private key $x_{i,j}$ to the Verifier, ensuring voter anonymity and vote privacy.

ZKP(1): This zero-knowledge proof ensures that the sum of all votes of the voter is 1. Suppose the number of candidates is 3, the voter v_i votes for these three candidates $m_{i,1}, m_{i,2}, m_{i,3}$, and satisfies the relationship $m_{i,1} + m_{i,2} + m_{i,3} = 1$. The sum of the ballots is calculated through the additive homomorphism property of ElGamal, and when the ciphertext of these three candidates is multiplied, the following is obtained:

$$\begin{aligned} & E(pk, y_{i,1}, x_{i,1}, m_{i,1}) \times E(pk, y_{i,2}, x_{i,2}, m_{i,2}) \times E(pk, y_{i,3}, x_{i,3}, m_{i,3}) \\ &= y_{i,1}^{x_{i,1}} \cdot y_{i,2}^{x_{i,2}} \cdot y_{i,3}^{x_{i,3}} \cdot g^{r_1+r_2+r_3}, g^{m_{i,1}+m_{i,2}+m_{i,3}} \cdot pk^{r_1+r_2+r_3} \\ &= y_{i,1}^{x_{i,1}} \cdot y_{i,2}^{x_{i,2}} \cdot y_{i,3}^{x_{i,3}} \cdot g^{r_4}, g^{m_{i,1}+m_{i,2}+m_{i,3}} \cdot pk^{r_4} \\ &= (c_1, c_2) \end{aligned} \tag{10}$$

Define $a = y_{i,1}^{x_{i,1}} \cdot y_{i,2}^{x_{i,2}} \cdot y_{i,3}^{x_{i,3}}$; it needs to be proved that a is generated in the correct format. The detailed steps are given in Algorithm 2.

Algorithm 2 The generation and validation of ZKP(a)

Parameters: selects random numbers $r_1, r_2, r_3 \in Z_q$ for generating temporary values in the proof.
Prover:
(1) Compute $R_1 = g^{r_1}, R_2 = g^{r_2}, R_3 = g^{r_3}$, where g is the cyclic group generator.
(2) Calculate the hash $h = \text{Hash}(A|R_1|R_2|R_3)$.
(3) Compute $B = X_{i,1} \cdot h + r_1, C = X_{i,2} \cdot h + r_2, D = X_{i,3} \cdot h + r_3$.
The Prover presents $A, B, C, D, R_1, R_2, R_3$ to the Verifier.
Verifier:
(1) The Verifier uses the formula $y_{i,1}^B \cdot y_{i,2}^C \cdot y_{i,3}^D = A \cdot z^h$ to verify the proof, where $y_{i,j}$ are distributed encryption values, z is the public key base.
(2) Compute $g^B = R_1 - (g^{x_{i,1}})^h, g^C = R_2 - (g^{x_{i,2}})^h, g^D = R_3 - (g^{x_{i,3}})^h$ to validate the values B, C, D.
If the calculations are correct, Accept the proof. Otherwise, reject the proof.
Zero-Knowledge Feature: The proof process does not reveal the voter's private keys $x_{i,j}$ to the Verifier, ensuring vote anonymity and privacy.

After the verification of the above proof, the prover needs to show that c_1/a and c_2/g have the same exponent using the proof scheme given in [25]. The detailed steps are given in Algorithm 3.

Algorithm 3 Procedure for proving that c_1/a and c_2/g have the same exponent

Parameters: The Prover selects a random number t for generating temporary values T_1 and T_2 in the proof.
Prover:
(1) Compute $T_1 = g^t$ and $T_2 = pk^t$, where g is the cyclic group generator and pk is the public key.
(2) Calculate the hash $h = \text{Hash}(T_1||T_2)$, concatenating T_1 and T_2 into a single string.
(3) Compute $s = r_4 \cdot h + t$, where r_4 is another random number selected by the Prover. The Prover sends T_1, T_2, s to the Verifier.
Verifier:
(1) The Verifier recalculates the hash $h = \text{Hash}(T_1||T_2)$ to ensure the Prover is using the same T_1 and T_2.
(2) The Verifier checks if $T_1 \cdot ((\frac{c_1}{a})^h) = g^s$, $T_2 \cdot ((\frac{c_2}{a})^h) = pk^s$
If the equation holds The proof is valid, confirming the homomorphic property of the encryption scheme. Else the proof is invalid.
Zero-Knowledge Feature: The proof process does not reveal the random numbers t and r_4 to the Verifier, ensuring the anonymity and privacy of the vote.

4.1.5. Verification Stage

The verification of the ballot is mainly carried out through the zero-knowledge proof given by the prover. First, each encrypted ballot needs to be verified to prevent calculation errors or possible malicious behavior when the voter generates the ciphertext. The specific verification process is in protocol 1. After completing the above verification process, an overall verification needs to be carried out. According to the voting rules, the sum of the votes of a voter for all candidates is 1. By using an encryption algorithm with additive homomorphism, anyone can calculate the total votes and complete the verification through ZKP(1). The specific verification process is shown in protocols 2 and 3.

4.1.6. Counting Stage

After all voters have completed voting, the election administrator needs to tally the votes and publish the results on the blockchain. Using the additive homomorphism property of the ballot ciphertext, the total votes are calculated. Since each ballot is encrypted using Formula (8), to count the votes for each candidate, the ciphertexts for the jth candidate from each voter are multiplied. This is represented as follows:

$$\begin{aligned} &\prod_{i=1}^{n_v} E(pk, y_{i,j}, x_{i,j}, m_{i,j}) \\ &= E(pk, y_{1,j}, x_{1,j}, m_{1,j}) \cdot \ldots \cdot E(pk, y_{n_v,j}, x_{n_v,j}, m_{n_v,j}) \\ &= \left(\prod_{i=1}^{n_v} y_{i,j}^{x_{i,j}} \cdot g^{r_1+\ldots+r_{n_v}}, \prod_{i=1}^{n_v} g^{m_{i,j}} \cdot pk^{r_1+\ldots+r_{n_v}}\right) \\ &= (g^r, g^{\sum_{i=1}^{n_v} m_{i,j}} pk^r) \end{aligned} \tag{11}$$

The result of Formula (11) can be regarded as the result of ElGamal encryption of $\sum_{i=1}^{n_v} m_{i,j}$. To decrypt this ciphertext, the election administrator must upload the partial decryption value $(g^r)^{sk}$ to the blockchain and generate a zero-knowledge proof ZKP(sk) to prove this value is correctly calculated using the private key sk. Any voter can calculate the total votes for the jth candidate using

$$g^{\sum_{i=1}^{n_v} m_{i,j}} = g^{\sum_{i=1}^{n_v} m_{i,j}} pk^r / (g^r)^{sk} \tag{12}$$

Similarly, the above operation is performed for each candidate to obtain the total votes for all candidates. Since any voter on the blockchain can publicly access all ciphertexts, each voter can independently calculate the votes for each candidate.

4.1.7. Recovery Stage

If the last voter and the administrator collude, they can know the voting results in advance and change the voting strategy or abstain, causing fairness issues. Therefore, the last voter's promised ballot can be recovered through the recovery phase, ensuring all voters obtain the correct result. Suppose the last voter v_{n_v} did not vote. Voter $v_i (i \in [1, n_v - 1])$ sends the recovery value $R_{i,j} = (g^{x_{i,j}})^{\rho_{ij}}$ to the smart contract. After any voter obtains the recovery value of all other voters, they can calculate the last voter's ballot $g^{m_{i,j}}$:

$$g_{m_{i,j}} = \frac{C_{i,j}}{\prod_{k=1,k\neq n_v}^{n_v} R_{k,j}} \tag{13}$$

After obtaining $g^{m_{i,j}}$, since $m_{i,j}$ takes values of 0 or 1, the value of $m_{i,j}$ can be quickly determined. Voter v_i can still calculate the sum of the votes of the first $n_v - 1$ voters through the above counting phase. In this way, the final voting result is obtained.

5. Correctness and Safety Analysis

In this section, we will theoretically analyze the security of the self-tallying voting protocol based on the ElGamal cryptosystem and distributed encryption algorithm and discuss its reliability and practicality in real-world applications.

5.1. Correctness

The correctness of the scheme is guaranteed by zero-knowledge proofs, ElGamal encryption, and blockchain. First, the scheme relies on smart contracts to automate the voting process, making it tamper-proof. The election administrator adds the election public key and other necessary parameters to the contract, enabling voters to vote securely and conveniently through the smart contract. The code logic in the smart contract ensures the correctness of the voting process, while blockchain technology guarantees that the voting results are tamper-proof.

Second, the scheme uses a zero-knowledge proof mechanism to protect voter privacy. During the voting process, voters use zero-knowledge proofs to ensure that the ciphertext they provide is generated according to the scheme's requirements. Finally, during the counting stage, the ciphertext exhibits additive homomorphism properties. Any participating entity on the blockchain can count the votes without decryption and obtain the ciphertext of the result. The election administrator then generates a partial decryption value along with the corresponding zero-knowledge proof and publishes it on the blockchain, allowing all participants to decrypt the ciphertext to obtain the final result.

Assuming that the voter knows all the ciphertext ballots and that the ballots are immutable, the counting process for the jth candidate's ballot is as follows:

$$\begin{aligned}
&\prod_{i=1}^{n_v} E(pk, y_{i,j}, x_{i,j}, m_{i,j}) \\
&= E(pk, y_{i,j}, x_{i,j}, m_{i,j}) \cdot \ldots \cdot E(pk, y_{n_{v,j}}, x_{n_{v,j}}, m_{n_{v,j}}) \\
&= (y_{1,j}^{x_{1,j}} \times g^{r_1}, g^{m_{1,j}} \times pk^{r_1}) \cdot \ldots \cdot (y_{n_v,j}^{x_{n_v,j}} \times g^{r_{n_v,j}}, g^{m_{n_v,j}} \times pk^{r_{n_v}}) \\
&= \left(\prod_{i=1}^{n_v} y_{i,j}^{x_{i,j}} \cdot g^{r_1+\ldots+r_{n_v}}, \prod_{i=1}^{n_v} g^{m_{i,j}} \cdot pk^{r_1+\ldots+r_{n_v}}\right) \\
&= (g^{r_1+\ldots+r_{n_v}}, g^{\sum_{i=1}^{n_v} m_{i,j}} \cdot pk^{r_1+\ldots+r_{n_v}}) \\
&= (g^{r}, g^{\sum_{i=1}^{n_v} m_{i,j}} \cdot pk^{r})
\end{aligned} \tag{14}$$

where g^{2m_j} represents the total number of votes received by the jth candidate.

5.2. Security Analysis

Assuming there are a total of n_v voters $v_1, v_2, \ldots, v_{n_v}$, and n_c candidates $c_1, c_2, \ldots, c_{n_c}$, we consider two types of adversaries based on security needs: passive adversaries and active adversaries. Passive adversaries do not actively participate in the voting process but

eavesdrop on the blockchain, attempting to obtain ballot information. Active adversaries may actively obstruct or manipulate the voting process, terminate voting prematurely, or collude with other voters to obtain more information about the voting.

The zero-knowledge proof provided in Figure 3 satisfies completeness, reliability, and honest verifier zero-knowledge (HVZK). The specific proof process is referenced in [22]. The MBS security of the voting scheme proposed in this chapter relies primarily on the security of this zero-knowledge proof protocol and the Decisional Diffie–Hellman (DDH) problem. The proof process is outlined below:

Theorem 1 (**MBS**). *If the probability of an adversary $\mathcal{A}$ winning in the MBS model within any polynomial time is negligible, then this electronic voting scheme is MBS secure.*

Proof. If there exists an adversary $\mathcal{A}$ within polynomial time that can win in the MBS model with a non-negligible advantage, an algorithm $\mathcal{B}$ can be constructed to break the zero-knowledge proof protocol and the DDH problem. Assume that the probability of the adversary winning in Game i is $\Pr[\text{Win}_i]$.

Game 0: Adversary $\mathcal{A}$ declares the target voters v_a and v_b, and the other voters are controlled by the adversary. The challenger $\mathcal{C}$ randomly selects a voter v_T from the target voters v_a and v_b. The ballot of v_T is $\{1\}$, while the ballot content of the other target voter is $\{0\}$. The challenger $\mathcal{C}$ generates the challenge ballots $\{C_a, V_a, Z_a\}$ and $\{C_b, V_b, Z_b\}$ for these two target voters, where C_i is the ballot commitment, V_i is the ciphertext ballot set, and Z_i is the zero-knowledge proof, $i \in \{a, b\}$. $\mathcal{A}$ outputs a guess result, indicating who cast the ballot $\{1\}$ among v_a and v_b.

$$\Pr[\text{Win}_0] = \Pr[\text{out} = T]$$

Game 1: Game 1 differs from Game 0 only in the zero-knowledge proof. In this game, $\mathcal{C}$ generates zero-knowledge proof $\{Z_a^*, Z_b^*\}$ through simulator $\mathcal{S}$ to replace $\{Z_a, Z_b\}$ in Game 0. At this point, the challenge ballots are $\{C_a, V_a, Z_a^*\}$ and $\{C_b, V_b, Z_b^*\}$. $\mathcal{A}$ wins the game if it can distinguish Game 1 from Game 0. If $\mathcal{A}$ wins the game within polynomial time with a non-negligible probability, then an algorithm $\mathcal{B}$ can be constructed to break the zero-knowledge proof protocol. The probability of the adversary winning in Game 1 satisfies the following formula:

$$|\Pr[\text{Win}_1] - \Pr[\text{Win}_0]| \leq \varepsilon_{\text{zkp}}$$

Game 2: The difference between Game 2 and Game 1 is that $\mathcal{C}$ generates two ballot commitments $\{C_a^*, C_b^*\}$ to replace $\{C_a, C_b\}$ in Game 1. The target voters v_a and v_b are the public keys g^a and g^b generated by c_j, $R \in \{g^{ab}, g^r\}$, where r is a random number. $\mathcal{C}$ chooses a random number for C_a^*, making $C_b^* = g^{m_{b,j}} \cdot R \cdot (g^b)^{\sum_{i=1, i\neq a,b}^{n_v} x_{i,j}}$. $\mathcal{A}$ wins the game if it can distinguish Game 2 from Game 1. If $\mathcal{A}$ wins the above game with a non-negligible advantage, then an algorithm $\mathcal{B}$ can be constructed to break the DDH problem. The probability of the adversary winning in Game 2 satisfies the following formula:

$$|\Pr[\text{Win}_2] - \Pr[\text{Win}_1]| \leq \varepsilon_{\text{DDH}}$$

Game 3: The difference between Game 3 and Game 2 is that $\mathcal{C}$ generates two ballot commitments $\{C_a^*, C_b^*\}$ to replace $\{C_a, C_b\}$ in Game 2. The target voters v_a and v_b are the public keys g^a and g^b generated by c_j, $R \in \{g^{ab}, g^r\}$, where r is a random number. $\mathcal{C}$ chooses a random number for C_b^*, letting $C_a^* = g^{m_{a,j}} \cdot R \cdot (g^a)^{\sum_{i=1, i\neq a,b}^{n_v} x_{i,j}}$. $\mathcal{A}$ wins the game if it can distinguish Game 3 from Game 2. If $\mathcal{A}$ wins the above game with a non-negligible advantage, then an algorithm $\mathcal{B}$ can be constructed to break the DDH problem. The probability of the adversary winning in Game 3 satisfies the following formula:

$$|\Pr[\text{Win}_3] - \Pr[\text{Win}_2]| \leq \varepsilon_{\text{DDH}}$$

Game 4: The difference between Game 4 and Game 3 is that $\mathcal{C}$ generates two ciphertext ballot sets $\{V_a^*, V_b^*\}$ to replace $\{V_a, V_b\}$ in Game 3. At this point, the challenge ballots generated by $\mathcal{C}$ are $\{C_a, V_a^*, Z_a\}$ and $\{C_b, V_b^*, Z_b\}$. For $\mathcal{A}$, this is of the same distribution. $\mathcal{A}$ wins the game if it can distinguish Game 4 from Game 3. If $\mathcal{A}$ wins the above game with a non-negligible advantage, then an algorithm $\mathcal{B}$ can be constructed to break the DDH problem. The probability of the adversary winning in Game 4 satisfies the following formula:

$$|\Pr[\text{Win}_4] - \Pr[\text{Win}_3]| \leq \varepsilon_{\text{DDH}}$$

Since the challenge ballot is indistinguishable from the adversary $\mathcal{A}$, the probability of the adversary winning in Game 4 is $\frac{1}{2}$. Summarizing the above game process, the probability of $\mathcal{A}$ winning the MBS game is less than or equal to $\frac{1}{2} + \varepsilon_{\text{zkp}} + 3\varepsilon_{\text{DDH}}$. Therefore, if the zero-knowledge proof protocol and DDH problem are secure, any $\mathcal{A}$'s advantage in winning the MBS game is negligible, and the self-tallying scheme proposed in this chapter is MBS secure.

The electronic voting system ensures Confidentiality through double encryption using ElGamal, protecting votes with both voter and administrator keys. It prevents Duplicate Voting by tracking plaintext voter identities against encrypted votes. Self-tallying allows voters to decrypt the final tally once the administrator provides partial decryption values. The system maintains Fairness by encrypting all ballots, making it impossible to influence results based on prior vote knowledge. Finally, verifiability is supported by the blockchain's immutable ledger, enabling voters to verify all steps and calculations. This system combines cryptographic security with transparent, tamper-evident record-keeping for a fair and trustworthy voting process. □

6. Experimental Analysis

In this section, an experimental analysis was conducted on the scheme proposed in the previous section, specifically referring to the calculation time of each step in the voting process. Three aspects of experimental analysis were specifically conducted, namely voting, verification, and counting. In this scheme, each ballot is encrypted twice using different keys. All tests are performed using 512-bit keys, which provide a higher level of security than one-time encryption using 1024-bit keys. Most of the calculations in this chapter are performed off-chain, and only the contract's verification of ciphertext and zero-knowledge proof is on-chain. Table 2 is the environment used for the experiment.

Table 2. Experimental environment.

Parameter Name	Parameter Value
Operating System	Win 11
Processor	AMD Ryzen 7 5700U
RAM	8 GB

Table 3 provides a comparison of the functionality and security of this scheme with the schemes of Li et al. [22], Mccorry et al. [18], and Yang et al. [24]. The specific comparison includes self-tallying, support for multiple candidates, ballot encryption efficiency, and malicious participants. From the table, it can be seen that this scheme satisfies these attributes.

From the table, it can be concluded that this scheme has the following advantages: This scheme has the attribute of self-tallying, which avoids the privacy issues brought about by third-party institutions counting votes, making the voting results more undisputed. It can solve the fairness problem in the self-tallying scheme. Compared with Li's scheme, this scheme supports voting for multiple candidates. Compared with Yang's scheme, this scheme has better efficiency in encrypting ballots, and this scheme has a recovery phase, which can better solve the problem of voters giving up voting.

Table 3. Comparison of the four options.

Scheme	Self-Tallying	Voting Functionality	Efficiency	Malicious Participants
Li [22]	yes	Two choices	Low	Support
Mccorry [18]	yes	Two choices	Low	Not Supported
Yang [24]	yes	Multiple choice	Low	Support
Our Scheme	yes	Multiple choice	High	Support

In this experiment, the time spent by voters in the process of encrypting votes was tested separately when the number of voters was 20 and under different numbers of candidates. From the figure, we can see that even if there are 20 candidates, the time cost of voting is less than 20 milliseconds. As the number of candidates increases, the time voters spend on encrypting ballots also increases accordingly, so the time voters spend on encrypting ballots is linearly related to the number of candidates. In the case of multiple-candidate voting, this scheme is more efficient in encrypting votes than Li's and Yang's schemes.

As shown in Figures 4 and 5, tests were conducted with 20 voters, and the time spent on encrypting ballots, counting votes, and generating zero-knowledge proofs for different candidates was measured. The time each voter spends is proportional to the number of candidates, and the time spent on encrypting ballots is longer than that of counting votes and generating zero-knowledge proofs. When the number of candidates is 20, the time spent on encrypting ballots, counting votes, and generating zero-knowledge proofs will not exceed 12 milliseconds.

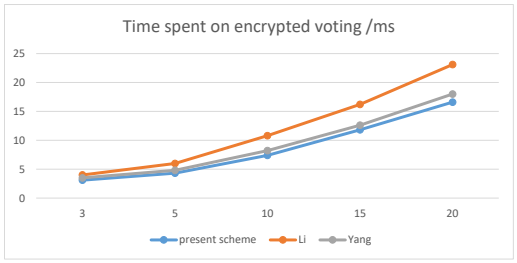


Figure 4. Encryption Time on voting.

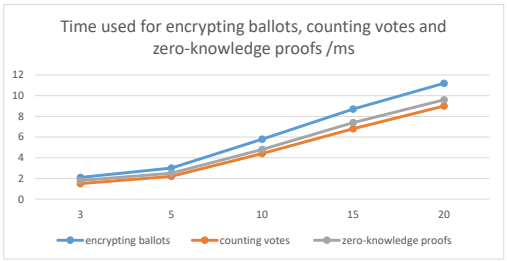


Figure 5. Time used for encrypting ballots, counting votes and zero-knowledge proofs.

Verifying a ballot primarily involves validating the digital signature of the ciphertext and the corresponding zero-knowledge proof of the ciphertext. Counting votes mainly includes verifying the partial decryption value provided by the election administrator and calculating the number of votes each candidate received.

As can be seen from Figure 6, the time spent by voters on verifying ciphertexts is much longer than the time spent on calculating ciphertexts and counting votes. However, overall, when the number of candidates is 20, the total verification time for an individual will not exceed 3.5 s.

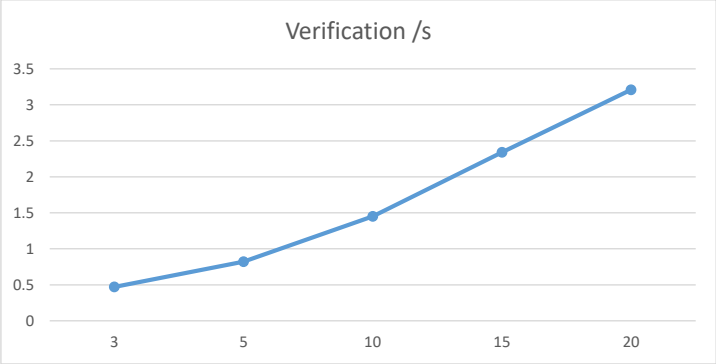


Figure 6. Time spent on the validation process.

7. Conclusions

This manuscript presents an analysis of Li's electronic voting scheme, identifying its limitations in terms of applicability to multiple candidates and issues surrounding fairness. To address these concerns, we introduce an innovative self-tallying voting scheme suitable for scenarios with multiple candidates. The scheme is constructed utilizing the ElGamal cryptosystem in conjunction with distributed encryption techniques.

A comparative analysis with Li's scheme reveals that the proposed scheme enhances the efficiency of encrypting ballots for multiple candidates without compromising the privacy of the votes. Leveraging the homomorphic properties inherent in the ElGamal cryptosystem, the scheme enables voters to tally votes directly from ciphertexts, eliminating the need for decryption. Critically, the scheme ensures fairness by requiring the election administrator to upload a partial decryption value, thereby preventing any voter from ascertaining the election outcome prematurely.

The paper concludes with an experimental analysis that substantiates the scheme's improved efficiency in handling multiple candidate elections while maintaining robust security measures. The findings demonstrate the scheme's potential to offer a more secure and efficient framework for electronic voting, particularly in elections with a larger pool of candidates. For future work, we will consider selecting more efficient encryption algorithms. Although the ElGamal encryption algorithm currently used is sufficiently secure, its encryption, decryption, and intermediate computational processes are relatively slow. In the future, we can consider using more efficient encryption algorithms to enhance the performance of the voting system.

Author Contributions: Conceptualization, X.D., X.Z. and D.Z.; project administration, D.Z.; funding acquisition, D.Z.; methodology, X.D. and D.Z.; software, X.Z.; validation, C.H. and X.Z.; formal analysis, D.Z. and X.D.; investigation, D.Z.; resources, C.H.; data curation, J.Z.; writing—original draft preparation, X.D. and X.Z.; writing—review and editing, X.D., X.Z., D.Z. and J.Z.; visualization, J.Z. and C.H.; supervision, D.Z. and C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Guangdong Provincial Key Laboratory of Power System Network Security (No. GPKLPSNS-2022-KF-05).

Data Availability Statement: All data are presented in the main text.

Conflicts of Interest: Chao Hong is employee of CSG Electric Power Research Institute Co., Ltd. The authors declare no conflicts of interest.

References

1. Wang, Y.; Zhou, J.; Cong, R.; Wang, L.; Huang, Y.; Chen, D. A Study on the Current Status and Development of Blockchain Technology in the Intelligent Management of Electric Vehicles. *South. Power Grid Technol.* **2022**, *16*, 55–67. [CrossRef]
2. Tang, X.; Yao, J.; Liu, W.; Yan, X.; Li, Z. Blockchain Technology-based Coordination Strategy for Electric Vehicles and Distributed Energy Interconnection. *South. Power Grid Technol.* **2022**, *16*, 46–54. [CrossRef]

3. Mannonov, K.M.U.; Myeong, S. Citizens' Perception of Blockchain-Based E-Voting Systems: Focusing on TAM. *Sustainability* **2024**, *16*, 4387. [CrossRef]
4. Chaum, D.L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **1981**, *24*, 84–90. [CrossRef]
5. Khader, D.; Smyth, B.; Ryan, P.; Hao, F. A fair and robust voting system by broadcast. In Proceedings of the 5th International Conference on Electronic Voting, EVOTE 2012, Bregenz, Austria, 11–14 July 2012; Lecture Notes in Informatics; Gesellschaft fur Informatik: Bonn, Germany, 2012; pp. 285–299.
6. Fiege, U.; Fiat, A.; Shamir, A. Zero knowledge proofs of identity. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1 January 1987; pp. 210–217.
7. Cramer, R.; Gennaro, R.; Schoenmakers, B. A secure and optimally efficient multi-authority election scheme. *Eur. Trans. Telecommun.* **1997**, *8*, 481–490. [CrossRef]
8. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
9. Fujioka, A.; Okamoto, T.; Ohta, K. A practical secret voting scheme for large scale elections. In Proceedings of the Advances in Cryptology—AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Australia, 13–16 December 1992; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 1993; pp. 244–251.
10. Joaquim, R.; Zúquete, A.; Ferreira, P. REVS–a robust electronic voting system. *IADIS Int. J. WWW/Internet* **2003**, *1*, 47–63.
11. Yu, B.; Liu, J.K.; Sakzad, A.; Nepal, S.; Steinfeld, R.; Rimba, P.; Au, M.H. Platform-independent secure blockchain-based voting system. In Proceedings of the Information Security: 21st International Conference, ISC 2018, Guildford, UK, 9–12 September 2018; Proceedings 21; Springer: Berlin/Heidelberg, Germany, 2018; pp. 369–386.
12. Liu, J.K.; Wong, D.S. Linkable ring signatures: Security models and new schemes. In Proceedings of the Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, 9–12 May 2005; Proceedings, Part II 5; Springer: Berlin/Heidelberg, Germany, 2005; pp. 614–623.
13. Lee, B.; Kim, K. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In Proceedings of the Information Security and Cryptology—ICISC 2002: 5th International Conference, Seoul, Republic of Korea, 28–29 November 2002; Revised Papers 5; Springer: Berlin/Heidelberg, Germany, 2003; pp. 389–406.
14. Zhao, Z.; Chan, T.H.H. How to vote privately using bitcoin. In Proceedings of the Information and Communications Security: 17th International Conference, ICICS 2015, Beijing, China, 9–11 December 2015; Revised Selected Papers 17; Springer: Berlin/Heidelberg, Germany, 2016; pp. 82–96.
15. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 16 June 2024).
16. Wood, G. A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
17. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 839–858.
18. McCorry, P.; Shahandashti, S.F.; Hao, F. A smart contract for boardroom voting with maximum voter privacy. In Proceedings of the Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, 3–7 April 2017; Revised Selected Papers 21; Springer: Berlin/Heidelberg, Germany, 2017; pp. 357–375.
19. Kiayias, A.; Yung, M. Self-tallying elections and perfect ballot secrecy. In Proceedings of the International Workshop on Public Key Cryptography, Paris, France, 12–14 February 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 141–158.
20. Groth, J. Efficient maximal privacy in boardroom voting and anonymous broadcast. In Proceedings of the International Conference on Financial Cryptography, Key West, FL, USA, 9–12 February 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 90–104.
21. Lee, K.; James, J.I.; Ejeta, T.G.; Kim, H.J. Electronic voting service using block-chain. *J. Digit. Forensics Secur. Law* **2016**, *11*, 8. [CrossRef]
22. Li, Y.; Susilo, W.; Yang, G.; Yu, Y.; Liu, D.; Du, X.; Guizani, M. A blockchain-based self-tallying voting protocol in decentralized IoT. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 119–130. [CrossRef]
23. Panja, S.; Bag, S.; Hao, F.; Roy, B. A smart contract system for decentralized borda count voting. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1323–1339. [CrossRef]
24. Yang, X.; Yi, X.; Nepal, S.; Kelarev, A.; Han, F. Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities. *Future Gener. Comput. Syst.* **2020**, *112*, 859–874. [CrossRef]
25. Schnorr, C.P. Efficient signature generation by smart cards. *J. Cryptol.* **1991**, *4*, 161–174. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.