WeVoting: Blockchain-based Weighted E-Voting with Voter Anonymity and Usability

Zikai Wang*, Xinyi Luo*, Meiqi Li*, Wentuo Sun*, Kaiping Xue*†
*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
†Corresponding author, kpxue@ustc.edu.cn

***Abstract*—E-voting plays a vital role in guaranteeing and promoting social fairness and democracy. However, traditional e-voting schemes rely on a centralized organization, leading to a crisis of trust in the vote-counting results. In response to this problem, researchers have introduced blockchain to realize decentralized e-voting, but the adoption of blockchain also brings new issues in terms of flexibility, anonymity, and usability. To this end, in this paper, we propose WeVoting, which provides weight-based flexibility with solid anonymity and enhances usability by designing a voter-independent on-chain counting mechanism. Specifically, we use distributed ElGamal homomorphic encryption and zero-knowledge proof to achieve voting anonymity with weight. Besides, WeVoting develops a counter-based counting mechanism to enhance usability compared with those self-tallying schemes. By critically designing an honesty-and-activity-based incentive algorithm, WeVoting can guarantee a correct counting result even in the presence of malicious counters. Our security and performance analyses elaborate that WeVoting achieves high anonymity in weighed voting under the premise of meeting the basic security requirements of e-voting. And meanwhile, its counting mechanism is sufficient for practical demands with reasonable overheads.**

***Index Terms*—Blockchain, Weighted E-voting, Counting Mechanism, Smart Contract.**

I. Introduction

As a fundamental way for people to freely express their will, electronic voting, or e-voting, plays a critical role in promoting fair social life due to its high efficiency and flexibility [1]. A primary concern of e-voting is its security, including eliminating illegal voters, protecting ballot confidentiality, guaranteeing counting correctness, etc. To this end, traditional e-voting systems typically utilize homomorphic encryption [2] or mix-net [3] schemes to meet security demands based on a central architecture. However, such central architecture may lead to single points of failure and prone to DDoS attacks. Therefore, the objective of building a decentralized e-voting system has received extensive attention.

Blockchain technology was first applied to Bitcoin [4] and is generally regarded as a secure distributed ledger maintained by multiple nodes. Due to its excellent efficacy in decentralized trust, tamper-resistant storage, and public verification, many studies, e.g., [5]–[7] have introduced it into e-voting to address the above security issues. They usually regard blockchain as a reliable and public storage that can securely record all the ballots and therefore concentrate on how to encrypt the ballots more securely or efficiently.

As a consequence, although existing blockchain-based e-voting schemes provide sufficient security, they do not perform well in terms of flexibility and usability. Specifically, on the one hand, weighted e-voting, in which each voter has a variable voting power (i.e., weight), is significant in many scenarios but not well-supported in existing schemes. This brings two challenges, i.e., how to protect the personalized weight that may reveal the specific voter and how to verify the weight of each ballot under the condition of anonymity. Some studies, e.g., [8], [9] propose score or range rules which provide a certain degree of variable voting power, but they are not effective enough in the weighted scenarios. On the other hand, most of those blockchain-based schemes [10], [11] adopt a self-tallying approach, in which each voter completes the counting process independently based on the ballots that are publicly stored on-chain. However, such a self-tallying mechanism severely limits the system's usability because the computing overhead to verify and decrypt the ballots is too huge for ordinary voters.

Considering the above limitations of existing blockchain-based e-voting schemes, we propose WeVoting, a blockchain-based weighted e-voting scheme that supports high-level anonymity and usability. For anonymity with weight, we use the ElGamal homomorphic encryption to encrypt a weighed ballot directly and construct a zero-knowledge proof [12], [13] to prove the validity of its weight, therefore solving the two challenges brought by weight mentioned above. To further enhance security, we distribute the ElGamal homomorphic encryption and therefore ensure that every single ballot can never ever be decrypted by anyone in the system. For voter usability in terms of counting, we design a voter-independent on-chain counting scheme. Our main idea is to select a group of counters for counting and resist malicious counters by an incentive-based counting consensus mechanism. Through simulation analysis, we prove that our proposed consensus mechanism can provide strong security even if there are 30% malicious counters in the system, sufficient for practical demands. In summary, this paper makes the following contributions:

- We propose WeVoting, a blockchain-based weighted e-voting scheme with voter anonymity and usability. We utilize the distributed ElGamal homomorphic encryption to enable efficient counting with user anonymity. Furthermore, we design a non-interactive zero-knowledge proof by the Fiat-Shamir heuristic methods to guarantee the correctness of weight.

- To reduce the computational overhead of voters and enhance usability, we propose a voter-independent and counter-based on-chain counting mechanism to replace the commonly used self-tallying mechanism. Through a carefully designed honesty-and-activity-based incentive algorithm, WeVoting can guarantee counting security and system liveness even when there are 30% malicious counters.
- Through security analysis, we prove that WeVoting supports strong voting security and can resist the malicious behaviors of various entities in an e-voting system. In addition, by simulating a long-term operation of the system, we demonstrate that the proposed incentive mechanism can identify and eliminate malicious counters and maintain fairness among honest counters.

The remainder of this paper is organized as follows. Section II presents the related works. Section III introduces the basic cryptography technology used in WeVoting. Section IV states the problem, including the system model, security assumptions, and design goals. We introduce the detailed schemes of WeVoting in Section V, analyze its security and performance in Section VI and VII, and finally conclude our work in Section VIII.

II. Related Works

Cryptography is an indispensable tool for achieving secure e-voting. Typically, in traditional electronic voting systems, homomorphic encryption [2] is used to complete counting without decrypting ballots, ensuring voter privacy. Besides, Mix-net [3] is implemented to achieve voting anonymity by disrupting the correspondence between voters and ballots. However, these traditional e-voting systems are all based on a centralized counting authority, leading to a single point of failure and a crisis of trust. Seeing the outstanding performance of blockchain in various decentralized applications, many studies, e.g., [5]–[7], discuss the advantages and potentials of blockchain-based e-voting systems.

Early studies utilize the blockchain to implement a basic secure e-voting system, neglecting flexibility and performance. For instance, McCorry *et al.* [10] proposed a decentralized voting protocol based on Ethereum that utilizes smart contracts to manage the entire voting, causing high computational overhead by Ethereum's consensus. Later, to enhance flexibility, researchers proposed rank-based [9] or score-based [8] schemes. The former allows every voter to rank the candidates, and the latter further enables voters to give each candidate a flexible score. However, they do not support weighted voting and their adoption of the self-tallying approach also causes severe usability limitations. To improve performance, Li *et al.* [14] proposed a reputation-based and online counter model by utilizing smart contracts, achieving efficient counting.

III. Preliminary

ElGamal encryption is widely applied for its inherent additive homomorphic property, specifically, by given a cyclic group $(\mathbb{G}, p, g)$, of order p with generator g and $\mathbb{Z}_p^* = \{1, \cdots, p-1\}$, user can generate a key pair $(x, y = g^x)$. For two encryptions: $\mathrm{E}(m_1) = (g^{r_1}, g^{m_1} \cdot y^{r_1})$, $\mathrm{E}(m_2) = (g^{r_2}, g^{m_1} y^{r_2})$ where $r_1, r_2 \in \mathbb{Z}_p^*$. And then, $\mathrm{E}(m_1) \cdot \mathrm{E}(m_2) = (g^{r_1+r_2}, g^{m_1+m_2} \cdot y^{r_1+r_2}) = \mathrm{E}(m_1 + m_2)$, the resulted ciphertext is an encryption of $m_1 + m_2$.

To further apply ElGamal homomorphic property in distributed multiple users scenarios, suppose each user has a key pair (x_i, g^{x_i}) and therefore the system public key can be constructed as $PK = \prod g^{x_i} = g^{\sum x_i}$. Each user's message m_i is encrypted as $\mathrm{E}(m_i) = (c_{i1}, c_{i2}) = (g^{r_i}, g^{m_i} \cdot PK^{r_i})$, where $r_i \in \mathbb{Z}_p^*$. Then $\prod \mathrm{E}(m_i) = (C_1, C_2) = (g^{\sum r_i}, g^{\sum m_i} \cdot PK^{\sum r_i})$, all participating users calculate and broadcast the decryption shares ${C_1}^{x_i}$ for obtaining the sum result of m_i by: $\frac{C_2}{\prod {C_1}^{x_i}} = \frac{g^{\sum m_i} \cdot PK^{\sum r_i}}{(\prod g^{r_i})^{\sum X_i}} = g^{\sum m_i}$.

IV. Problem Statement

A. System Model

As shown in Fig. 1, there are three kinds of entities in WeVoting, i.e., organizers, voters, and counters.

- **Organizer:** The voting organizer is responsible for initializing a vote and announcing important parameters, including voter list. It should also execute the *Committee-Selection* algorithm to randomly select a group of counters to form the counting committee for the current vote.
- **Voter:** A voter can generate a ballot with yes, no, or $abstention$ and then encrypt it to conceal his/her own will and construct a zero-knowledge proof to ensure the correctness of weight w.
- **Counter:** Any blockchain user can become a counter to participate counting phase and earn rewards by *Registration*. Counters are constrained by *Incentive*, where honest counters are rewarded, and malicious ones are punished.

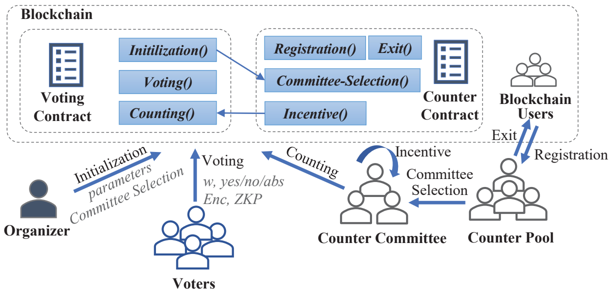


Fig. 1. System Model of WeVoting

B. Security Assumptions

We discuss security assumptions separately from four aspects, i.e., adversaries and system entities consisting of organizers, voters, and counters.

- **Adversaries** can be anyone that curious about the ballots or trying to interfere with the voting process by tampering with ballots or masquerading as legal voters.

- **Organizers** are only responsible for publishing vote parameters and invoking *Committee-Selection* algorithm and is therefore honest in the blockchain environment.
- **Voters** may repeatedly vote, be curious about the middle result, or maliciously change their voting weights.
- **Counters** aim to earn rewards through honest counting, but some malicious ones may try to upload wrong counting results to interfere with the voting result. We analyze of the effect caused by the malicious ones in Section VI.

C. Design Goals

- **Anonymity with weight:** Anonymity requires that the correspondence between the voter's identity and the ballot would remain confidential. In a weighted vote, it is also necessary that weights and voters are unlinkable, and meanwhile, the correctness of weights can be verified during counting.
- **Verifiability and Neutrality:** Every voter can verify the correctness of the final counting result and the legality of the counting process. In addition, secure e-voting requires that voters cannot acquire the intermediate result to prevent their preferences from being influenced.
- **Completeness and Correctness:** Completeness requires that all ballots are published and not be tampered with. In the counting phase, all legal ballots are counted in the final result, and all illegal ballots are discarded.
- **Fairness and Usability:** Fairness requires that honest and active counters are highly likely to be elected into the counting committee. Usability requires that the counting mechanism is efficient and does not require the voter to consume excessive computing power.

V. The Proposed Scheme

A. Overview

WeVoting is a blockchain-based weighted e-voting scheme with voter anonymity and usability. It utilizes distributed ElGamal homomorphic encryption and zero-knowledge proof to realize flexible weighted voting with anonymity. In addition, to enhance voter usability, WeVoting provides a voter-independent and counter-based on-chain counting mechanism. Through the honesty-and-activity-based *Incentive* and *Committee-Selection* algorithms, a group of counters forms a counting committee to complete the counting. In general, WeVoting consists of three phases for e-voting. Firstly, at the initialization phase, the voting organizer (VO) publishes the voting information on the blockchain through the voting contract (VC) and invokes *Committee-Selection* to select the counting committee. Besides, voters generate their key pairs and upload the public keys to VC to build the system public key. Next, during the voting phase, voters use the system public key to encrypt weighted ballots and generate zero-knowledge proofs for the ballot and weight. They should also sign the ballots with their private keys. Finally is the counting phase, where counters in the counting committee aggregate all valid ballots by verifying the signatures and zero-knowledge proofs. Then, each voter generates a decryption share based on the required decryption information and uploads it to VC. Counters can then compute the final voting result with the decryption shares and upload it to VC for consensus.

B. Three Voting Phases of WeVoting

There are three main voting phases in WeVoting, i.e., initialization, voting, and counting.

1) Initialization: During the initialization phase, the voting organizer (VO) initializes the voting and voters $(V_1, \cdots, V_n)$ register their public keys where n is the total number of voters for the current voting. Each voter V_i has two key pairs, i.e., (pk_i^s, sk_i^s) and (pk_i^e, sk_i^e), used for signing and encrypting respectively.

- Voting initialization: VO publishes (ID_i, w_i, pk_i^s) and $(\mathbb{G}, p, g)$ to VC. Where the former is V_i's voting information, including ID, weight and signing public key. $\mathbb{G}$ is a cyclic group of order p with generator g used for distributed ElGamal key generation and $\mathbb{Z}_p^* = \{1, 2, \cdots, p-1\}$. VO should also invoke *Committee-Selection* to select the counting committee for current voting.
- Voter registration: V_i chooses a random secret key $sk_i^e = X_i$ from $\mathbb{Z}_p^*$ and uploads the public key $pk_i^e = g^{X_i}$ to VC. Therefore, the system public key PK is constructed as $g^{\sum X_i} = \prod g^{X_i}$.

2) Voting: Voters generate and publish their ballots during the voting phase.

- Ballot generation: Each V_i computes its raw ballot $b_i = m_i w_i$, where m_i is from $\{1, -1, 0\}$ represents voting intentions of yes, no and abstain, and w_i is V_i's weight. Besides, V_i selects a random integer $r_i \in \mathbb{Z}_p^*$ and encrypts b_i as $(C_{1i}, C_{2i}) = (g^{r_i}, g^{b_i} \cdot PK^{r_i})$. In addition, V_i constructs a non-interactive zero-knowledge proof Z_i on (C_{1i}, C_{2i}) to prove that $b_i \in \{w_i, -w_i, 0\}$ as

$$Z_i = ZKP\left\{g^{w_i} \cdot PK^{r_i} \vee g^0 \cdot PK^{r_i} \vee g^{-w_i} \cdot PK^{r_i}\right\}.$$

  Finally, V_i generates its ballot as

$$B_i = (ID_i, w_i, pk_i^s) || (C_{1i}, C_{2i}) || Z_i.$$

- Ballot publication: To guarantee legitimacy and integrity of the ballot, V_i generates a signature $S_i = \mathrm{S}ig(sk_i^s, B_i)$ and uploads $B_i || S_i$ to VC.

3) Counting: A counting committee consists of N counters $(CT_1, \cdots, CT_N)$ is selected during the initialization phase. After the voting phase, the counters will work together to verify ballots and count the result.

- Counter: Each counter CT_i verifies each ballot B_j by S_j and Z_j and then aggregates all the legal ballots as $C_1^i = \prod_j C_{1j}$, $C_2^i = \prod_j C_{2j}$ and uploads (C_1^i, C_2^i) to VC. VC will receive N pairs of (C_1^i, C_2^i). They will be the same if all counters count correctly; otherwise there will be differences. VC takes the most result as the correct one, denote as (C_1, C_2).

- Decrypt: Each voter V_j gets C_1 from VC and calculates $S_j = C_1^{X_j}$ and uploads S_j to VC. CT_i can then decrypt the aggregated result $\sum b_i = \sum w_i m_i$ by:

$$g^{\sum b_i} = \frac{C_2}{\prod S_i} = \frac{g^{\sum b_i} \cdot PK^{\sum r_i}}{(\prod g^{r_i})^{\sum X_i}}.$$

Similarly, after receiving N counters results, VC takes the most result as the correct final result. Referring to [10] and [8], entities can easily get $\sum b_i$ by brute forcing the discrete logarithm of the result.

C. Counter Management of WeVoting

To enhance voter usability, WeVoting counts the ballots with the help of counters rather than self-tallying. Therefore, how to guarantee high security in the presence of a part of malicious counters becomes a major challenge. To this end, our main idea is to design an honest-and-liveness-based incentive mechanism to realize that honest and active counters are highly likely to be selected for counting to earn rewards, and malicious ones will be identified and punished. On the basis of the incentive mechanism, we design a set of management functions for counters

1) The Stake $\mathrm{S}(h, a)$*:* An effective incentive requires that counters with the same honesty and activity have basically equivalent chance of participating in counting and earning rewards. Thus, we use the stake $S = \mathrm{S}(h, a)$ to evaluate a counter's reliability from honesty (h) and activity (a). Specifically, h is related to the time t of correct counting and number of days d in the counting pool, formulated as

$$h = h(t, d) = t(1 + \lambda)^d.$$

where λ is an increasing factor. Thus, this design is friendly to "old" and honest counters. Besides, a is related to the frequency of participation in counting. For activity, inspired by the concept of coin age in proof-of-stake [15] which regards the holding time of the coin as a factor to measure the coin's value to prevent "old" users from accumulating exorbitant capabilities. On the contrary, we use this time-based idea to increase the selection probability of active counters and therefore improve the liveness of the system. Specifically,

$$a = a(k) = \prod_{i=0}^{k} (1 + \mu i).$$

where μ is the impact factor, and i represents the past k counting tasks. To summarize, S can be formulated as

$$S = \mathrm{S}(h, a) = \omega/(1 + a\theta^{-h}).$$

where ω is a limit value and θ is an exponential value.

2) Management Functions for Counters: As shown in Fig. 1, WeVoting provides four main functions for counter management, i.e., registration, exit, incentive, and committee-selection.

- *Registration and Exit*: Any blockchain user can become a counter by funding a predetermined initial fund to the counters contract. Then, he/she will get an initial stake and enter the counting pool. Counters can withdraw counting rewards and further exit counting pool if all the remain fund is withdrawn.
- *Incentive*: Based on the stake $\mathrm{S}(h, a)$, the *Incentive* algorithm is shown in Algorithm 1. If counter CT_i submits a correct counting result, he/she first receives the reward. Further, his/her h_i and a_i will be adjusted by modifying t, d and k and subsequently the stake S_i will increase. However, if CT_i maliciously submits a wrong counting result, his/her S_i will be reduced sharply by adjusting t. If t is much lower than the initial value $\omega/2$, CT_i will be recognized as malicious.

Algorithm 1: Incentive

```
Input: The counting final result $result$,
Counters $CT_{i\in[1,N]}$ in the current counting committee,
$S_i = \mathrm{S}(h_i(t, d), a_i(k))$ of $CT_{i\in[1,N]}$,
The counting reward $reward$ and penalty factor $\rho$.
Output: $S_i$ of $CT_{i\in[1,N]}$.
1  $N_c \leftarrow$ the number of counters that provide correct result.
2  for each $CT_i$ with $\langle result_i, balance_i, h_i(t_i, d_i), a_i(k_i)\rangle$ do
3      if $result_i == result$ then
4          $balance_i$ += $reward/N_c$;
5          $t_i += 1$;
6          $k_i = 0$;
7      else
8          $d_i = 0$;
9          if $t_i > 0$ then
10             $t_i = \lfloor t_i/\rho \rfloor$;
11         else
12             $t_i -= 2$;
13     compute $S_i = \mathrm{S}(h(d_i, t_i), a(k))$;
14 Return $S_{i\in[1,N]}$;
```

- *Committee-Selection*: Invoked by the voting organizer to select a group of counters for the current counting task according to their stakes S. The detailed algorithm is shown in Algorithm 2. The main idea is to arrange all counters on an axis and the length of the axis occupied by each counter is equal to its stake. Subsequently, we randomly select a point on this section of axis and select the corresponding counter into the counting committee. Therefore, the probability of counter CT_i being selected is $\frac{S_i}{total}$, that is, the probability is proportional to the stake. In addition, the stake computation mechanism $\mathrm{S}(h, a)$ sets the upper limit of stake to ω. Therefore, the maximum probability of being selected is restrained, preventing some counters to become too strong to destroy the decentralization of counting mechanism. To further enhance security, we can also set the lower limit of stake S_l to reject the counters with too small stake to participate into the selection. Moreover, even if a malicious counter is selected, it will be recognized and penalized by *Incentive*.

Algorithm 2: Committee-Selection

```
**Input:** Counters $CT_{i\in[1,N_{cp}]}$ in counting pool,
stake $S_i$ of each counter $CT_i$,
number of counters $N_{cc}$ in a counting committee.
**Output:** A counting committee $CC$.
1  counter set $W \leftarrow \Phi$;
2  sum of counters stake $total \leftarrow 0$;
3  for $j = 1$ to $N_{cp}$ do
4      $W[j] = S_j$;
5      $total = total + S_j$;
6  for $j = 1$ to $N_{cc}$ do
7      $index = Random[0, total]$;
8      for $k = 1$ to $N_{cp}$ do
9          if $index$ in $W[k]$ then
10             $CC[j] = CT_k$;
11 Return $CC$;
```

VI. Security Analysis

In this section, we analyze how the proposed WeVoting resists possible attacks. The details are as follows:

A. Attacks against Anonymity with weight

- **Adversaries trying to get correspondences between voters and ballots:** The ballot encryption information for every voter V_i is $(C_{i1}, C_{i2}) = (g^{r_i}, g^{b_i} * PK^{r_i})$, and decrypting a ballot requires the cooperation of all voters. If V_i doesn't give his/her secret $(g^{r_i})^{X_i}$, no one can get b_i. Therefore, a single voter's option is unable to be exposed. Besides, Although the weight is publicly recorded in the ballot B_i, since the ballot is encrypted, this only reveals whether the voter has voted, but not the specific choice, achieving anonymity with weight.

B. Attacks against Verifiability and Neutrality

- **Voters trying to repeatedly voting:** Under the disclosure of the correspondence between voter's identity and weight, the ballot is constructed as $B_i = (ID_i, w_i, pk_i^s)||(C_{i1}, C_{i2})||Z_i$. Thus, repeatedly voting can be detected by checking B_i's ID in WeVoting.
- **Voters trying to get the middle voting result before submitting ballots:** To get the middle voting result, V_i needs to compute:

$$g^{\sum b_i} = \frac{g^{\sum b_i} * PK^{\sum r_i}}{(\prod g^{r_i})^{\sum X_i}}.$$

  But it takes the cooperation of all voters to get the result.
- **Voters trying to submit wrong weight in a ballot:** By (ID_i, w_i, pk_i^s), counters can check whether ID_i and w_i are matched, and by Z_i, counters can check whether the raw ballot b_i is one of $(w_i, -w_i, 0)$. Therefore, a voter cannot use a wrong weight to generate a valid ballot.

C. Attacks against Completeness and Correctness

- **Adversaries trying to tamper with ballots:** We apply blockchain technology to WeVoting so that once the ballot is published to the blockchain, no one can tamper with it.
- **Adversaries masquerading as legal voters:** Legal V_i's (ID_i, w_i, pk_i^s) is collected and published by VO in the initialization phase. Thus, illegal voters can be recognized in the counting phase by verifying the legitimacy of V_i's signature $S_i = Sig(sk_i^s, B_i)$.
- **Counters maliciously or incorrectly counting:** Due to Algorithm 1 and the *review* function that once occur a different counting result, all counters will recheck the final result for correctness. In WeVoting, the counting mechanism filters out all malicious counters after several times of counting.

D. Attacks against Fairness and Usability

- **Counters trying to bypass the random selection mechanism:** The counting mechanism runs on the blockchain and supports *Committee-Selection* and *Incentive* algorithms to achieve secure counting. Thus, every CT_i is selected by the counting mechanism based on the stake.

VII. Performance Analysis

A. The performance of computational

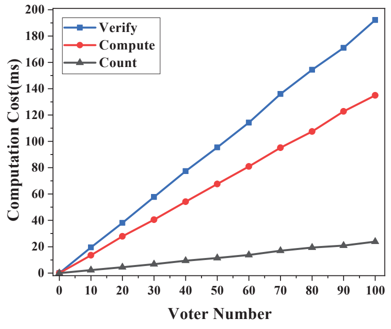


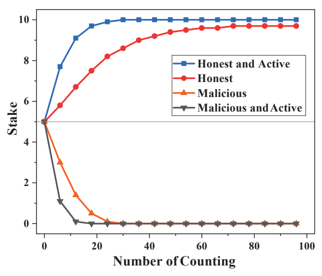
Fig. 2. Time Cost of Computational.

To test the computational consumption of our scheme, we conduct experiments on a computer with 3.90Ghz AMD Ryzen and 16.0GB RAM. As shown in Fig. 2, WeVoting completes experiments in computational cost, including verification of signature S_i, computing Z_i and counting B_i. We set the number of voters to range from 10 to 100, and the cost of ballot counting and verification increases linearly as the number of voters increases indicating reasonable overheads. All of these calculations are done off-chain by the counters, which significantly reduces the computing burden on voters compared to self-tallying schemes.

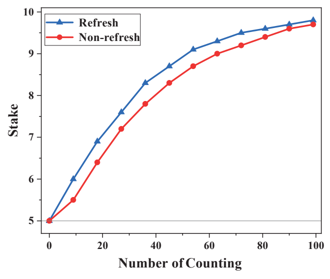
TABLE I
Consumption of Three Phases

Voting Phase	Gas Cost	ETH Cost
Initialization	715127	0.0143
Voting	2920661	0.0584
Counting	2440824	0.0488

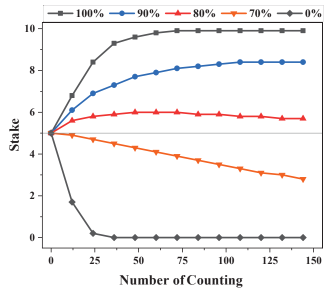
Besides, the voting process implementation is supported by the smart contract deployed on the Ethereum test network. TABLE I shows the gas cost and ETH cost for the three voting phases of WeVoting, demonstrating the acceptable performance of our scheme. Meanwhile, the lower consumption required for the counting phase indicates that the proposed counting mechanism enhances the usability of WeVoting.



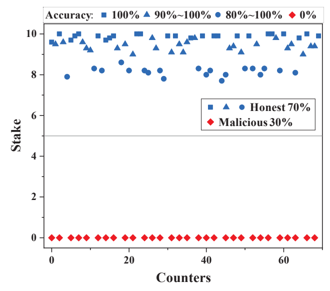
(a) S of Four Types Counters.



(b) Refersh and Non-refresh.



(c) Counter with Different Accuracy.



(d) Stable State of Counting Mechanism.

Fig. 3. The Performance of Counting Mechanism.

B. The performance of counting mechanism

WeVoting completes the usability verification experiment of the counting mechanism. As shown in Fig. 3a, the trend of different CT_i's stake S_i converges with the increasing number of counting, and activity (a) affects the number of counting required to reach the maximum probability of CT_i. Thus, WeVoting encourages honest and active CT_i participation in counting phase. Fig. 3b shows that CT_i can increase S for a higher probability of being selected by using the *Refresh* function which costs a fee to improve a. Besides, for a single counter, we assume that CT_i has a different degree of counting accuracy and analyze the effect of accuracy on S. Fig. 3c shows that honest CT_i must achieve at least 80% accurate to increase S in WeVoting. After several times of counting, Fig. 3d shows that WeVoting's counting mechanism with 30% malicious counters reaches a stable state, and all malicious counters are recognized and removed from counting. Therefore, even with 30% malicious counters in the system, WeVoting can maintain the ballot counting correct and secure.

VIII. Conclusion

In this paper, we proposed WeVoting, a blockchain-based weighted e-voting scheme that supports high-level anonymity and usability. To address the issue of weighted voting security under anonymity, we utilized distributed ElGamal homomorphic encryption and zero-knowledge proof to preserve privacy for the ballot with weight. For usability in terms of ballot counting, we proposed a voter-independent and counter-based on-chain counting mechanism supported by designing an honesty-and-activity-based incentive algorithm. Finally, we proved that WeVoting meets the voting security requirements through our security analysis. In terms of performance experiments analysis, we verified through simulations that the counting mechanism could effectively resist 30% malicious counters, supports strong security with reasonable overheads and significantly satisfies practical demands.

Acknowledgment

The work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2008, pp. 354–368.

[2] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Proceedings of the 2000 International Conference on the Theory and Applications of Cryptographic Techniques (Advances in Cryptology-EUROCRYPT)*. Springer, 2000, pp. 539–556.

[3] M. Jakobsson, A. Juels, and R. L. Rivest, "Making mix nets robust for electronic voting by randomized partial checking," in *Proceedings of the 11th USENIX Security Symposium (USENIX Security)*. USENIX, 2002, pp. 339—-353.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008, accessed: May., 2022.

[5] J. Huang, D. He, M. S. Obaidat, P. Vijayakumar, M. Luo, and K.-K. R. Choo, "The application of the blockchain technology in voting systems: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–28, 2021.

[6] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Software*, vol. 35, no. 4, pp. 95–99, 2018.

[7] R. Hanifatunnisa and B. Rahardjo, "Blockchain based e-voting recording system design," in *Proceedings of the 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*. IEEE, 2017, pp. 1–6.

[8] Y. Yang, Z. Guan, Z. Wan, J. Weng, H. H. Pang, and R. H. Deng, "Priscore: blockchain-based self-tallying election system supporting score voting," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4705–4720, 2021.

[9] X. Yang, X. Yi, S. Nepal, A. Kelarev, and F. Han, "Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities," *Future Generation Computer Systems*, vol. 112, pp. 859–874, 2020.

[10] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proceedings of the 2017 International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2017, pp. 357–375.

[11] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *Proceedings of the 2002 International Workshop on Public Key Cryptography (PKC)*. Springer, 2002, pp. 141–158.

[12] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[13] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proceedings of the 1986 Conference on the Theory and Applications of Cryptographic Techniques (Advances in Cryptology-CRYPTO)*. Springer, 1986, pp. 186–194.

[14] M. Li, X. Luo, W. Sun, J. Li, and K. Xue, "AvecVoting: Anonymous and verifiable E-voting with untrustworthy counters on blockchain," in *Proceedings of the 2022 IEEE International Conference on Communications (ICC)*. IEEE, 2022, pp. 4751–4756.

[15] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *Self-Published Paper, August*, vol. 19, no. 1, 2012.