A Privacy Preserving and Format-Checkable E-voting Scheme

Yuhong Sun(✉), Shiyu Wang, Fengyin Li, and Hua Wang

School of Computer Science, Qufu Normal University, Rizhao 276826, Shandong, China
sun_yuh@163.com

Abstract. Electronic voting (e-voting) is widely used because of its convenience and efficiency. In response to the security problems in e-voting, such as the legality of voters, privacy disclosure, etc., this paper proposes a novel e-voting scheme that can check the format of ballots without disclosing its content based on homomorphic encryption. Firstly, voters encrypt their ballots with Paillier encryption before sending them to the counter. Then, the counter decomposes the encrypted ballots using the proposed *n*-ary conversion protocol, and performs the format check of the ballots. Only ballots with the correct format are counted. During the whole process of voting, no one except the voter himself can know each ballot's content, even the counter, so that the privacy of ballots is preserved. Finally, the counter performs an additive homomorphism operation on the encrypted ballots and the voting manager decrypts it to tally the result. Besides the requirements including the legality, privacy, and integrity, we furtherly consider the validity of the ballots in e-voting and make the scheme more practical than the existing methods.

Keywords: E-voting · Format-checkable · Paillier encryption · *n*-ary conversion protocol

1 Introduction

In recent years, electronic voting (e-voting) is becoming a popular method of replacing the traditional paper voting [1]. The e-voting [2] is superior to the traditional voting in efficiency and economy, but it faces several security threats such as the voter legitimacy, ballot uniqueness, privacy preserving, etc.

The existing e-voting schemes, mostly use the cryptographic protocol to guarantee the privacy and the correctness of the voting result, and can be roughly classified into three main categories: schemes based on the mix-net [3] schemes based on blind signature [4] or ring signature [5], and schemes based on homomorphic encryption [6].

The recent work, such as Kumar et al. [7], who proposed a secure end-to-end verifiable e-voting scheme based on identity-based blind signatures, where the end-to-end verification allows the voter to check whether his ballot was recorded correctly as he intended. The shortcoming of this scheme is that the selected candidate's name appears on the ballot and the privacy of the ballot cannot be preserved well. In 2019, Yining et al.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
L. Wang et al. (Eds.): WASA 2022, LNCS 13472, pp. 475–488, 2022.
https://doi.org/10.1007/978-3-031-19214-2_40

[8], proposed an e-voting scheme based on secret sharing and k-anonymity, which satisfies unconditional security, but suffers from the problem that it is hard to check whether the voter has cheating behavior. And in 2016, Shahandashti et al. [9] proposed a voting strategy named 'DRE-ip' that improved the DRE (Direct-Recording Electronic) system and can publicly verify the voting result without decrypting the ballots. The shortcoming of this system is that the voting result, even the format check by NIZK, relies on the recording party heavily. Although the blockchain technology to e-voting can increase the security and transparency of voting schemes and reduce the reliance on third-party institutions, an unavoidable fact is that they suffer from the latency due to the verification in a p2p network, such as [10, 11], etc.

In fact, whether the blind signature-based schemes or the homomorphic encryption-based schemes must meet such a contradiction: the ballot should be hidden by operation of encryption or blinding for the privacy before it is issued to the receiver, but the receiver cannot confirm whether the ballot in a correct format. Examples such as in literatures [7–14], there are not any format checks before the ballots are accepted. However, in reality, the voter may cast a ballot that includes more than one "approval" for the same candidate. The encryption or blinding operation will prevent this behavior from being discovered.

To solve the above problem, we propose a novel e-voting scheme based on Paillier encryption that can check the format of the ballot, so that the cheating behavior from voters can be revealed by the counter. Specifically, the contribution of this paper can be summarized as follows: i) We investigated the existing e-voting schemes and analyzed the necessity of format check in reality. ii) A privacy preserving and format check e-voting scheme based on Paillier encryption is proposed, that can check the format of the ballot without disclosing its content. iii) We analyzed the proposed scheme from multiple security requirements. iv) We gave the comparison of the proposed scheme and other e-voting schemes from the performance.

The rest of the paper is organized as follows. In Sect. 2, some preliminaries are introduced, and the system model is presented in Sect. 3. Section 4 presents the proposed format-checkable e-voting scheme. Section 5 provides security analysis. Finally, Sect. 6 concludes the paper.

2 Preliminaries

2.1 Paillier Cryptosystem

KeyGen. Randomly select two large prime numbers p, q, and $g \in Z_{N^2}$, let $N = p \cdot q$, $\lambda = 1cm(p-1,\ q-1)$, $l(u) = \frac{u-1}{N}$, $gcd\left(l(g^{\lambda} mod\ N^2), N\right) = 1$. The public key is (N, g), and the secret key is λ.

Encrypt. Randomly select integer $r \in Z_{N^2}^*$, for the plaintext $m \in Z_N$, the ciphertext is $c = g^m \cdot r^N mod\ N^2$.

Decrypt. The plaintext $m = \frac{l(c^{\lambda} mod\ N^2)}{l(g^{\lambda} mod\ N^2)} mod\ N$.

The Paillier cryptosystem has the additive homomorphism [15] property:

$$E(m_1) \cdot E(m_2) = (g^{m_1} \cdot r_1^N) \cdot (g^{m_2} \cdot r_2^N) = g^{m_1+m_2}(r_1 r_2)^N = E(m_1 + m_2)\ mod\ N^2$$

2.2 Boneh-Boyen Signature

In this paper, we use the Boneh-Boyen signature scheme as [16].

KeyGen. Let G, G_T be prime p order cyclic groups, g is a generator of G, and there exists a bilinear pairing e: $G \times G \rightarrow G_T$. The secret key is $x \in Z_p$, and the corresponding public key is $y = g^x$.

Sign. The signature on a message m is $\sigma = g^{1/(x+m)}$.

Verify. Verification is done by checking that $e(\sigma, y \cdot g^m) = e(g, g)$.

2.3 Secure Bit-Decomposition Protocol (SBD)

Suppose that there are two parties, Alice and Bob. Bob holds the Paillier encrypted value $E(x)$, where $0 < x \leq 2^{\mu}$ (μ is the domain size of x in bits). Let $\left(x_0, x_1, \cdots, x_{\mu-1}\right)$ denotes the binary representation of x. The goal of SBD is to convert the encryption of x into the encryptions of the individual bits of x, without disclosing any information regarding x to both parties. We use the SBD protocol in this paper as [17, 18]: $\text{SBD}(E(x)) \rightarrow (E(x_0), \cdots, E(x_{\mu-1}))$. The details of the protocol are shown in algorithm 2.

3 System Model and Notations

3.1 System Entities

The main participants of this scheme include the election commission authority (*ECA*), voters (V_is), the authentication center (*AC*), and the counting center (*CC*), The entities and their interactions are shown in Fig. 1, and their functions are described as follows.

Election commission authority (*ECA*): *ECA* initializes the system, interacts with the *CC* during the counting phase, decrypts and announces the final voting results.

Authentication center (*AC*): *AC* authenticates voters, verifies the legitimacy of each voter's identity, and issues him the unique voting identification ID_i, which is unrelated to his identity information.

Voters (V_i): A voter V_i must be authenticated to obtain a unique voting identification ID_i before participating in the voting.

Counting center (*CC*): *CC* collects the encrypted ballots, checks the format of each ballot, and counts the ballots with the correct format.

3.2 Trust Assumption

In our scheme, the trust assumptions are as follows.

1) *ECA* is assumed to be honest, it is authoritative and usually acts as the voting manager in reality.
2) *AC* and *CC* are assumed to be semi-honest, or honest but curious.
3) The voter V_i is not assumed to be honest, because he may cast a ballot consisting of more than one "approval" for the same candidate.

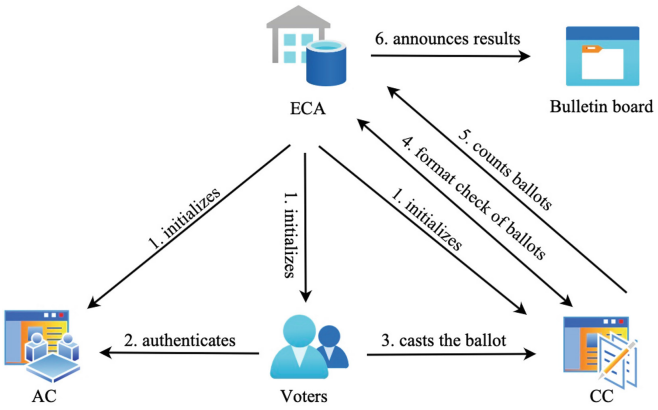

Fig. 1. System model

3.3 Notations Description

The notations used throughout the paper are listed in Table 1.

Table 1. Notations and descriptions

Notations	Descriptions
S_i	Identity information of V_i
ID_i	Unique voting identification of V_i
$(v_0, v_1, \ldots, v_{m-1})$	V_i's ballot in plaintext for each candidate
$E(M_i)$	Paillier encryption of decimal ballot M_i
$E(M_i) \rightarrow (E(c_{m-1}), \ldots, E(c_0))$	The n-ary conversion of $E(M_i)$
SBD $(E(M_i)) \rightarrow (E(\beta_0), \ldots, E(\beta_{\mu-1}))$	The SBD conversion of $E(M_i)$
$List^{V_i}$	List of all legal voters
$List^{AC}$	List of authenticated voters
$List^{CC}$	List of voters who have cast their ballot

4 Our Scheme

4.1 Scheme Overview

Our voting scheme can be applied in the scenario of k-out-of-m (voters can choose k individuals from m candidates, $k \geq 1$). The implementation of the proposed scheme consists of four phases.

Initialization phase: Each participating entity generates its own public/private keys following the Paillier cryptosystem and the Boneh-Boyen signature system.

Authentication phase: *AC* authenticates each voter based on V_i's identity information S_i according to $List^{V_i}$, and issues the encrypted ID_i to the legal voter. Lastly, *AC* publishes authenticated voters' ID_i in the $List^{AC}$.

Voting phase: During the voting phase, V_i generates an encrypted ballot and sends it to *CC*. *CC* firstly decomposes the encrypted ballot, then it checks whether the format of the ballot is correct.

Counting phase: After format check, *CC* counts the encrypted ballots with the correct format by homomorphic addition and sends the result to *ECA*. *ECA* decrypts the counting result and publishes the final voting result. The *CC* publishes ID_i as the $List^{CC}$.

4.2 Initialization Phase

Given a security parameter κ, each participant generates its own public/secret keys, and publishes their public keys. *AC* holds the $List^{V_i}$ with all legal voters' identity information, and generates $List^{AC}$ with initial values null. *CC* also initializes $List^{CC}$ with null.

The Paillier cryptosystem is choosen for the property of additive isomorphism. Firstly, *ECA* selects the parameters as (p, q, g), generates its public key $(N, g)_{ECA}$ and secret key λ_{ECA}, where $N = p \cdot q$, $\lambda = 1cm(p-1,\ q-1)$. Similarly, V_i generates public key $(N, g)_{V_i}$ and secret key λ_{V_i}. *AC* generates public key $(N, g)_{AC}$ and secret key λ_{AC}.

We choose the short signature of Boneh-Boyen signature system. Firstly, V_i selects the parameters as (G, G_T, g), generates its public key y_{V_i} and secret key $x_{V_i} \in Z_p$, where $y_{V_i} = g^{x_{V_i}}$. Similarly, *AC* generates public key y_{AC} and secret key $x_{AC} \in Z_p$. *CC* generates public key y_{CC} and secret key $x_{CC} \in Z_p$.

4.3 Authentication Phase

Before voting, each voter must be authenticated. The authentication phase is listed as follows.

Step1. V_i signs his identity information S_i to generate $sig_{V_i}(S_i)$, then encrypts S_i and $sig_{V_i}(S_i)$ together using *AC*'s public key as $E_{AC}(S_i, sig_{V_i}(S_i))$, and sends it to *AC*.

Step2. After receiving $E_{AC}(S_i, sig_{V_i}(S_i))$, *AC* decrypts it to get $(S_i, sig_{V_i}(S_i))$, then checks whether S_i is in $List^{V_i}$. If it is, *AC* verifies $sig_{V_i}(S_i)$. The V_i is regarded as a legal voter if $sig_{V_i}(S_i)$ is verified. Otherwise, *AC* rejects this authentication request.

Step3. Next, *AC* checks whether S_i exists in $List^{AC}$. If it is not, which means V_i is authenticated for the first time, then V_i is recorded in $List^{AC}$. Otherwise, it indicates V_i has already authenticated before, and *AC* rejects this request.

Step4. V_i is considered legal and eligible to vote if it is verified both in *Step2* and *Step3*. Then *AC* generates a unique voting identification ID_i for V_i independent of S_i to ensure the anonymity of V_i. Each voter only knows his own ID_i, and no one can associate the ID_i with his real identity S_i except *AC*. Then *AC* signs and encrypts the ID_i as $E_{V_i}(ID_i, sig_{AC}(ID_i))$, and returns it to V_i.

Step5. V_i decrypts $E_{V_i}(ID_i, sig_{AC}(ID_i))$ to get ID_i and $sig_{AC}(ID_i)$, and verifies $sig_{AC}(ID_i)$. If it is valid, V_i keeps $(ID_i, sig_{AC}(ID_i))$ as his voting certificate, and AC appends the S_i and ID_i to $List^{AC}$.

Step6. In the end, all ID_i in $List^{AC}$ are published. Voters can check whether they have been regarded as legitimate voters. If an authenticated voter cannot find his ID_i, he can apply for authentication again to AC.

4.4 Voting Phase

This scheme assumes that there are N' voters $(V_1, \cdots, V_{N'})$ and m candidates $(C_1, \cdots, C_m)$, and each voter can vote for all candidates. Firstly, the ballot is represented as an m-bit n-ary number. When the candidate $C_j (0 \leq j \leq m-1)$ is approved, the j-th bit of the ballot is 1. Otherwise, it is 0. If there are too many voters, CC can divide N' voters randomly into t groups, where each group consists of n_t voters. When CC receives a ballot from a voter, it adopts the proposed n-ary conversion protocol to decompose the encrypted ballots, where $n = n_t + 1$. Then, CC checks the format of encrypted ballots without disclosing the content. The voting phase for each group is listed as follows.

Step1. V_i $(0 \leq i \leq n_t - 1)$ creates a ballot $(v_0, v_1, \cdots, v_{m-1})$ for all candidates as his wish, and converts this ballot from n-ary to a decimal M_i, where $M_i = v_{m-1} \cdot n^0 + \cdots + v_0 \cdot n^{m-1}$. Then, he generates $E_{ECA}(M_i)$ using ECA's public key. Lastly, V_i sends $(ID_i, sig_{AC}(ID_i), E_{ECA}(M_i))$ to CC.

Step2. After receiving $(ID_i, sig_{AC}(ID_i), E_{ECA}(M_i))$, CC firstly checks whether the ID_i exists in $List^{CC}$. If it exists, it means that the voter has cast a ballot before, and CC neglects him. Otherwise, $sig_{AC}(ID_i)$ is verified by CC and if it is valid, it indicates that the V_i is an authorized voter.

Step3. If above verifications are valid, CC decomposes $E_{ECA}(M_i)$ from a decimal number to an n-ary number without decryption. The specific steps are shown in algorithm 1. In this process, the SBD protocol in algorithm 2 is invoked to decompose $E_{ECA}(M_i)$ into bits in binary, and algorithm 4 is invoked to decide whether the last bit of the data is 0 or 1 in the next operation.

Step4. After decomposing $E_{ECA}(M_i)$, CC performs the format check on the decomposed ballot $(E_{ECA}(c_{m-1}), \cdots, E_{ECA}(c_0))$. Firstly, CC inputs $(E_{ECA}(c_{m-1}), \cdots, E_{ECA}(c_0))$ and 2 into algorithm 5. If the algorithm outputs false, this means that the format of the ballot is incorrect. Then, CC rejects the ballot and sends feedback to the V_i. Otherwise, CC appends ID_i to $List^{CC}$.

Here, we give an example to show how the encrypted ballot decomposition works. Suppose there are three voters and four candidates, so $n = 3 + 1 = 4$. V_i creates a ballot (0011) according to his own will, and calculates $M_i = 1 + 1 \times 4 = 5$, then he encrypts M_i to get $E_{ECA}(5)$ using ECA's public key. After CC receives the $E_{ECA}(5)$, it executes algorithm 1.

Firstly, *CC* executes the SBD protocol to decompose $E_{ECA}(5)$ into $(E_{ECA}(0), E_{ECA}(1), E_{ECA}(0), E_{ECA}(1))$, which is the initial value of q for the first cycle. After the first cycle of decomposition operation in algorithm 1, *CC* gets $E_{ECA}(c_0) = E_{ECA}(A) = E_{ECA}(1)$ as shown in Table 2. The last set of $q = (0001)$ is used as the initial value of q for the next cycle. And similarly, *CC* gets $E_{ECA}(c_1) = E_{ECA}(1)$, $E_{ECA}(c_2) = E_{ECA}(0)$, $E_{ECA}(c_3) = E_{ECA}(0)$ in the next three cycles. Finally, the decomposed encrypted ballot is $(E_{ECA}(c_3), E_{ECA}(c_2), E_{ECA}(c_1), E_{ECA}(c_0)) = (E_{ECA}(0), E_{ECA}(0), E_{ECA}(1), E_{ECA}(1))$.

Algorithm 1 n-ary Conversion

Input: $E_{ECA}(M_i)$, m ($m \leq \mu$, μ is the number of bits in the binary format of M_i)

Output: $(E_{ECA}(c_{m-1}), \cdots, E_{ECA}(c_0))$

1. ***@CC***:
2. executes **SBD**
3. $(E_{ECA}(\beta_0), \cdots, E_{ECA}(\beta_{\mu-1}))\leftarrow \text{SBD}(E_{ECA}(M_i))$
4. marks $(E_{ECA}(q_0), \cdots, E_{ECA}(q_{\mu-1}))\leftarrow(E_{ECA}(\beta_0), \cdots, E_{ECA}(\beta_{\mu-1}))$
5. initializes $(E_{ECA}(c_{m-1}), \cdots, E_{ECA}(c_0))\leftarrow(E_{ECA}(0), \cdots, E_{ECA}(0))$
6. **for** j=0 to m-1 **do**
7. **for** executing μ times **do** (for j=0 to μ-1)
8. initializes $(E_{ECA}(a_0), \cdots, E_{ECA}(a_{\mu-1}))\leftarrow(E_{ECA}(0), \cdots, E_{ECA}(0))$
9. $E_{ECA}(a_{j-1})=E_{ECA}(a_j)$
10. $E_{ECA}(a_{\mu-1})=E_{ECA}(q_0)$
11. $E_{ECA}(q_{j-1})=E_{ECA}(q_j)$
12. calculates $E_{ECA}(A)=E_{ECA}(a_{\mu-1})\cdot 2^0+\cdots+E_{ECA}(a_0)\cdot 2^{\mu-1}$
13. executes **ComparePro** $(E_{ECA}(A), n)$
14. calculates $E_{ECA}(q_{\mu-1})=E_{ECA}(1)\cdot E_{ECA}(Q)^{N-1}=E_{ECA}(1-Q)$
15. $E_{ECA}(B)=E_{ECA}(-q_{\mu-1})^n=E_{ECA}(-nq_{\mu-1})$
16. $E_{ECA}(A)=E_{ECA}(A)\cdot E_{ECA}(B)=E_{ECA}(A+B)$
17. **end for**
18. $E_{ECA}(c_j)=E_{ECA}(A)$
19. **end for**

Algorithm 2 SBD $(E_{ECA}(M_i)) \rightarrow E_{ECA}(\beta_0), \cdots, E_{ECA}(\beta_{\mu-1}))$

Input: $E_{ECA}(M_i)$
Output: $(E_{ECA}(\beta_0), \cdots, E_{ECA}(\beta_{\mu-1}))$

1. $L \leftarrow 2^{-1}$
2. $T \leftarrow E_{ECA}(M_i)$
3. **for** δ=0 to μ-1 **do**
4. ***@CC***:
(1) $Y \leftarrow T \cdot E_{ECA}(b)$, $b \in Z_N$
(2) sends Y to ECA
5. ***@ECA***:
(1) receives Y from CC
(2) $d \leftarrow D_{ECA}(Y)$
(3) if d is even $\alpha \leftarrow E_{ECA}(0)$
else $\alpha \leftarrow E_{ECA}(1)$
(4) sends α to CC
6. ***@CC***:
(1) receives α from ECA
(2) if α is even $E_{ECA}(\beta_\delta) \leftarrow \alpha$
else $E_{ECA}(\beta_\delta) \leftarrow E_{ECA}(1) \cdot \alpha^{N-1}$
(3) return $E_{ECA}(\beta_\delta)$
7. $Z \leftarrow T \cdot E_{ECA}(\beta_\delta)^{N-1}$
8. $T \leftarrow Z^L$ (update T with the encrypted value of current quotient in algorithm 3)
9. **end for**

Algorithm 3 Binary$(M_i) \rightarrow (\beta_0, \cdots, \beta_{\mu-1})$

Input: M_i
Output: $(\beta_0, \cdots, \beta_{\mu-1})$

1. **for** δ=0 to μ-1 **do**
2. $\beta_\delta \leftarrow M_i \ mod \ 2$ (β_δ is updated to current quotient)
3. $M_i \leftarrow \left\lfloor \frac{M_i}{2} \right\rfloor$
4. **end for**

Algorithm 4 ComparePro

Input: $E_{ECA}(A)$, n
Output: Q
1. ***@CC***:
(1) encrypts n to get $E_{ECA}(n)$
(2) calculates $E_{ECA}(S)=E_{ECA}(A)\cdot E_{ECA}(n)^{N-1}=E_{ECA}(A-n)$
(3) chooses two random numbers r_1, r_2, $r_1>r_2$, $L(r_1)<L(N)/8$
(4) encrypts r_2 to get $E_{ECA}(r_2)$
(5) flips a coin c randomly
if c=0, calculates $E_{ECA}(R)=E_{ECA}(r_2)\cdot E_{ECA}(S)^{r_1}=E_{ECA}(r_2+r_1S)$
if c=1, calculates $E_{ECA}(R)=E_{ECA}(r_2)\cdot E_{ECA}(S)^{N-r_1}=E_{ECA}(r_2-r_1S)$
(6) sends $E_{ECA}(R)$ to *ECA*
2. ***@ECA***:
(1) receives $E_{ECA}(R)$ from *CC*
(2) decrypts $E_{ECA}(R)$ to get R
(3) if $R>\frac{N}{2}$, u=1
else u=0
(4) sends u to *CC*
3. ***@CC***:
(1) receives u from *ECA*
(2) if c=0, calculates $Q=u$
if c=1, calculates Q=1-u
return Q (if Q=0, it shows $S \geq 0$, $A \geq n$, if Q=1, it shows $S < 0$, $A < n$)

Algorithm 5 FormatCheck

Input: $(E_{ECA}(c_{m-1}), \cdots, E_{ECA}(c_0))$, 2
Output: True or False
1. **for** j=0 to m-1 **do**
2. $z \leftarrow$**ComparePro** $(E_{ECA}(c_j), 2)$
3. if z=0 break
4. **return** False
5. **end for**
6. if $j=m$-1
7. **return** True

Table 2. Example of n-ary conversion ($j = 0$)

Round	a	q	n	Operations
1	0000 0000	0101 1010	4	Shift left a and q together $A \leftarrow 0 < n, Q \leftarrow 1, q_{\mu-1} \leftarrow 0$ $B \leftarrow -n \cdot q_{\mu-1} \leftarrow 0, A \leftarrow A + B \leftarrow 0$
2	0000 0001	1010 0100	4	Shift left a and q together $A \leftarrow 1 < n, Q \leftarrow 1, q_{\mu-1} \leftarrow 0$ $B \leftarrow -n \cdot q_{\mu-1} \leftarrow 0, A \leftarrow A + B \leftarrow 1$
3	0001 0010	0100 1000	4	Shift left a and q together $A \leftarrow 2 < n, Q \leftarrow 1, q_{\mu-1} \leftarrow 0$ $B \leftarrow -n \cdot q_{\mu-1} \leftarrow 0, A \leftarrow A + B \leftarrow 2$
4	0010 0101 0001	1000 0000 0001	4	Shift left a and q together $A \leftarrow 5 > n, Q \leftarrow 0, q_{\mu-1} \leftarrow 1$ $B \leftarrow -n \cdot q_{\mu-1} \leftarrow -4, A \leftarrow A + B \leftarrow 1$

4.5 Counting Phase

After voting, *CC* can obtain t groups, where each group consists of n_t encrypted ballots $E_{ECA}(M_i)$. Then, *CC* performs homomorphic addition on ballots for each group. Finally, *ECA* decrypts the counting result and announces the final voting result. The counting phase is listed as follows.

Step1. CC calculates each group of ballots homomorphically to obtain $E_{ECA}(M_\varepsilon) = \prod_{i=1}^{n_t} E_{ECA}(M_i)$, where $(1 \leq \varepsilon \leq t)$.

Step2. For each group, *CC* signs $E_{ECA}(M_\varepsilon)$ to generate $sig_{CC}(E_{ECA}(M_\varepsilon))$, and sends $(E_{ECA}(M_\varepsilon), sig_{CC}(E_{ECA}(M_\varepsilon))$ to *ECA*.

Step3. ECA verifies $sig_{CC}(E_{ECA}(M_\varepsilon))$, decrypts $E_{ECA}(M_\varepsilon)$ to get the voting result M_ε, and converts M_ε to the n-ary number, where each digit is the election result for each candidate in ε-th group. Then, *ECA* aggregates the result of each group and announces the results on a bulletin board.

Step4. In the end of counting phase, $List^{CC}$ is published. V_i can check whether his ID_i exists in $List^{CC}$. If V_i cannot find his ID_i, he can reapply to *CC*.

4.6 Complexity

The complexity in the presented scheme is mainly on the computation cost and communication cost. Since the initialization phase and authentication phase are preparation for the voting, we consider the costs during the voting and the counting phases. Firstly, a voter creates his ballot by one encryption. Then, the *CC* checks the legality of the voter by signature and needs one verification. The *CC* runs algorithm1 to decompose the encrypted ballot. In the process, algorithm 2 is called to convert the encrypted ballot to binary, where μ times of loops for the multiplication and addition and two communications between *CC* and *ECA* are needed. Then, algorithm 1 calls algorithm 4 to decide

the last bit of the data in the next operation. Algorithm 4 needs two encryptions and one decryption with two communications between *CC* and *ECA*. By the all, algorithm 1 will be completed with the complexity of $O(\mu + m^*\mu^*1) = O(m\mu)$ encryptions and decryptions and $2m\mu$ communications, where m is the number of candidates and μ is the bit length of each ballot. Next, algorithm 5 is used to check the format of ballot by revoking algorithm 4 in m times, and the complexity of the algorithm 5 is O(m). Last, the *CC* calculates the encrypted ballot results by multiplying them only by N' times.

5 Scheme Analysis

5.1 Security Analysis

Correctness. Each voter creates his ballot as an n-ary number and converts it to a decimal number before encryption. The counter uses the n-ary conversion protocol to decompose the ballots and obtains each encrypted digit of the n-ary number. Since $n = n_t + 1$, the addition on the n_t ballots cannot produce any carry-over, so *CC* only needs to do the homomorphic addition once on the encrypted ballots for one group. The *AC* can decrypt and aggregate the voting result to get the final result.

Format Check. The *CC* adopts the ComparePro protocol to check the correctness of the format of the encrypted ballot. If a voter casts a ballot that contains more than one "approval" for the same candidate, the ComparePro protocol can detect the incorrect ballot, and *CC* rejects the ballot. This solution overcomes the lack of format correctness checking in existing voting systems, and avoids the possibility of fraud by voters.

Privacy. Privacy includes the privacy of the voter's identity and the privacy of the ballot. In the authentication phase, if a voter is authenticated by *AC*, he can get a voting identification ID_i that is unrelated to his identity. *AC* can only confirm the legitimacy of the voter, but cannot know who he is. During the voting phase, voters encrypt their ballots, so that no one can know the contents of the ballots or link them to the real identities of voters. In the format checking, the *ECA* and *CC* are assumed not to be colluded, they also cannot know the plaintext of the ballot. The privacy of the voter and the ballot content can be preserved.

Legitimacy. The *AC* will authenticate the voter's identity. Only if the voter's identity information exists in the $List^{V_i}$, the voter will be regarded as a legal voter, and will be issued a unique voting identification ID_i unrelated to his identity. So no illegal person can obtain the authorization.

Uniqueness. During the authentication phase, only the voter who has been authenticated by the *AC* can obtain the unique ID_i, and the $List^{AC}$ can guarantee that the voter cannot be authenticated repeatedly. Throughout the voting process, the voter cannot vote repeatedly, the $List^{CC}$ can check if the voter has voted before.

Fairness. Throughout the scheme, the ballot of each voter can only be known by himself, and no one including the *ECA*, *AC* or *CC* can know the content of the ballot. Any intermediate voting result cannot be revealed during the scheme until the final results are published.

Integrity. Throughout the voting, all authentication information and counting information are verified and counted by the *AC* and *CC* respectively. The relevant information is published in the $List^{AC}$ and $List^{CC}$. Each voter can check whether his information is correctly recorded in the above lists according to his ID_i.

Accuracy. Each voter has to pass the authentication of the *AC* before casting a ballot, and the *CC* only receives ballots from the legal voters. Therefore, the final counting result is legal, and the addictive homomorphism ensures the correctness of the final counting results.

5.2 Performance Comparison

In this section, the scheme is mainly analyzed in comparison with the schemes proposed in literature [7–12]. The literature [7] proposes an end-to-end verifiable scheme using identity-based blind signature. As mentioned in Sect. 1, the scheme cannot preserve the privacy of each voting, and voters can only vote for one candidate at a time. The e-voting scheme based on secret sharing in [8] lacks a rigorous authentication process and format check for ballots. The DRE-ip based voting system such as [9], depends on the recording machine. The blockchain based voting scheme such as [10] cannot be used to vote for multiple candidates, besides the latency in verification. The e-voting scheme in [11] and [12] also lacks of the format check for ballots, besides the voters' identities authentication and ballots' uniqueness. The comparison results are shown in Table 3.

Table 3. Performance comparison of different e-voting schemes

Scheme	Legitimacy	Privacy	Uniqueness	Format check	Multi choice
[7]	√	√≠	√	×	×
[8]	√≠	√	√	×	√
[9]	√	√	√	×	√
[10]	√	√	√	×	×
[11]	√	√	√	×	√
[12]	√	√	×	×	√
Ours	√	√	√	√	√

6 Conclusion

In this paper, we propose a privacy-preserving e-voting scheme that can check the format of the ballot while protecting the privacy of the content. The scheme implements a *k*-out-of-*m* choice for candidates. Voters generate ballots according to their wishes and encrypt the ballots by Pallier encryption. The counting center uses the *n*-ary conversion

protocol and the ComparePro protocol to check the formats of ballots, and only those ballots that pass the check will be counted. The counting result is obtained through only one additive homomorphic operation by the counting center. On the one hand, this scheme does not disclose the privacy of voters and ballots. On the other hand, it guarantees the correct formats of the ballots. This scheme satisfies the basic properties of privacy, legality, accuracy, and other requirements of e-voting systems. It is more useful and practical than existing e-voting systems. However, in the proposed scheme, it is the counter center, and not the voter, undertakes more computation cost for the format check. It may result the inefficiency with too many voters in reality and this is the future work of this paper.

References

1. Pu, H.Q., Cui, Z., Liu, T.: A review of research on secure e-voting schemes. Comput. Sci. **47**(9), 8 (2020)
2. Mursi, M.F.M., Assassa, G.M.R., Abdelhafez, A.: On the development of electronic voting: a survey. Int. J. Comput. Appl. **61**(16), 1–11 (2013)
3. Peng, K.: A general and efficient countermeasure to relation attacks in mix-based e-voting. Int. J. Inf. Secur. **10**(1), 49–60 (2011)
4. Ku, W.C., Wang, S.D.: A secure and practical electronic voting scheme. Comput. Commun. **22**(3), 279–286 (1999)
5. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
6. He, Q., Shen, W.: Homomorphic encryption-based multi-candidate e-voting scheme. Comput. Syst. Appl. **28**(2), 146–151 (2019)
7. Kumar, M., Chand, S., Katti, C.P.: A secure end-to-end verifiable internet-voting system using identity-based blind signature. IEEE Syst. J. **14**(2), 2032–2041 (2020)
8. Liu, Y., Zhao, Q.: E-voting scheme using secret sharing and K-anonmity. World Wide Web **22**(4), 1657–1667 (2019)
9. Shahandashti, S.F., Hao, F.: DRE-ip: a verifiable E-voting scheme without tallying authorities. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 223–240. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45741-3_12
10. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 357–375. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_20
11. Chaieb, M., Koscina, M., Yousfi, S., Lafourcade, P., Robbana, R.: Dabsters: a privacy preserving e-voting protocol for permissioned blockchain. In: Hierons, R.M., Mosbah, M. (eds.) ICTAC 2019. LNCS, vol. 11884, pp. 292–312. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32505-3_17
12. Waheed, A., Din, N., Umar, A.I.: Novel blind signcryption scheme for e-voting system based on elliptic curves. Mehran Univ. Res. J. Eng. Technol. **40**(2), 314–322 (2021)
13. Alam, K., Tamura, S., Rahman, S.: An electronic voting scheme based on revised-SVRM and confirmation numbers. IEEE Trans. Dependable Secure Comput. **99**, 400–410 (2019)
14. Ajish, S., Anilkumar, K.S.: Secure mobile internet voting system using biometric authentication and wavelet based AES. J. Inf. Secur. Appl. **61**(14), 102908 (2021)

15. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
16. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4
17. Samanthula, B.K.K., Chun, H., Jiang, W.: An efficient and probabilistic secure bit-decomposition. InL ACM SIGSAC Symposium on Information, pp. 541–546. ACM (2013)
18. Liu, X., Deng, R.H., Choo, K.: An efficient privacy-preserving outsourced calculation toolkit with multiple keys. IEEE Trans. Inf. Forensics Secur. **11**(11), 2401–2414 (2016)