KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS VOL. 15, NO. 5, May. 2021

Efficient Proxy Re-encryption Scheme for E-Voting System

Wenchao Li[1], Hu Xiong[2*]
1 School of Cyber Science and Technology, Beihang University, Beijing 100191, China
2 School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[e-mail: xionghu.uestc@gmail.com]
*Corresponding author: Hu Xiong

Received Septebmer 11, 2020; revised November 22, 2020; accepted April 14, 2021; published May 31, 2021

Abstract

With the development of information and communication technologies, especially wireless networks and cell phones, the e-voting system becomes popular as its cost-effectiveness, swiftness, scalability, and ecological sustainability. However, the current e-voting schemes are faced with the problem of privacy leakage and further cause worse vote-buying and voter-coercion problems. Moreover, in large-scale voting, some previous e-voting system encryption scheme with pairing operation also brings huge overhead pressure to the voting system. Thus, it is a vital problem to design a protocol that can protect voter privacy and simultaneously has high efficiency to guarantee the effective implementation of e-voting. To address these problems, our paper proposes an efficient unidirectional proxy re-encryption scheme that provides the re-encryption of vote content and the verification of users' identity. This function can be exactly applied in the e-voting system to protect the content of vote and preserve the privacy of the voter. Our proposal is proven to be CCA secure and collusion resistant. The detailed analysis also shows that our scheme achieves higher efficiency in computation cost and ciphertext size than the schemes in related fields.

Keywords: Re-encryption, e-voting system, efficient encryption, security.

This work was supported by the National Natural Science Foundation of China (Grant No. 61672002, Grant No. U1636208, No. 61862008, No. 61902013). This work was also supported by the Foundation of Jiangxi Educational Committee (Grant No. GJJ190464).

http://doi.org/10.3837/tiis.2021.05.015 **ISSN : 1976-7277**

1. Introduction

Electronic voting has been applied in politics, the economy, and even daily entertainment to collect people's opinions about political and social decisions. With the help of deep learning in data classification and detection technology, the electronic voting system can achieve higher efficiency and accuracy in statistics [1−2]. In order to make electronic elections completely democratic, a security mechanism is needed to ensure the privacy of voters. In addition, the content of the vote includes important information, such as which one candidate the voter casts, the vote is an "affirmative vote", "dissenting vote" or an "abstention vote". For these reasons, the protection of user identity and voting content is an essential function in the voting process. Once the voter's identity or the vote content was exposed during the voting process, then some corrupt candidates will force or seduce the voter to vote for him/her.

Cryptography technology serves as an information protection mechanism that can strengthen the security of voting content. In the context of cryptography, the public key encryption technology can be used to convert the voting content into the non-readable format, so that no one except the voter himself/herself can decrypt the ciphertext to obtain the specific vote content. At present, cryptography technology is widely used to strengthen the security of the network and resist various attacks [3-8]. The voting system is no exception. Some cryptography technology is applied in the voting system to ensure secure, reliable. The authors have discussed the realization of receipt-freeness function by re-encrypting ciphertext [9]. In [10], timed-release cryptography is also used to prevent the early opening of electronically-case votes in the e-voting system, thus it avoids election fraud.

However, non-readable format votes under public key encryption are difficult to count directly. In this context, voter can directly transmit his/her private key to the bulletin to decrypt the ballot, and then every candidate's ballot is counted. Unfortunately, this way is easy to cause the disclosure of the voter's private key and further threat voter's identity security. Moreover, in large-scale voting, the voting under public key encryption requires the server of the voting center to exchange information multi times with each voter, which is impractical to apply in the real environment.

We noticed that the structure of proxy re-encryption (PRE) matches voting system well with multiple requirements. Proxy re-encryption can not only ensure that the content of the ballot remains encrypted during transmission, but also can re-encrypt the ballot through the proxy to delegate the specific user to decrypt it. Blaze et al. [11] provided the notion of PRE, in which the proxy is given a conversion key that allows it to transform a message encrypted with the public key of the data sender into another message encrypted with the public key of the data receiver. Due to its better flexibility and convenience, PRE is applicable to many practical scenarios, such as cloud computing [12], personal health records [13], distributed file systems [14] , and secure email forwarding [11].

To illustrate more specifically, let us consider the process of proxy re-encryption operation, as shown in **Fig. 1**. Suppose the data sender (say, Alice) plans to share the own encrypted data with the data receiver (say, Bob). Alice wants to keep identity anonymous in this transformation process and guarantee that except Bob, nobody can access the sensitive data.

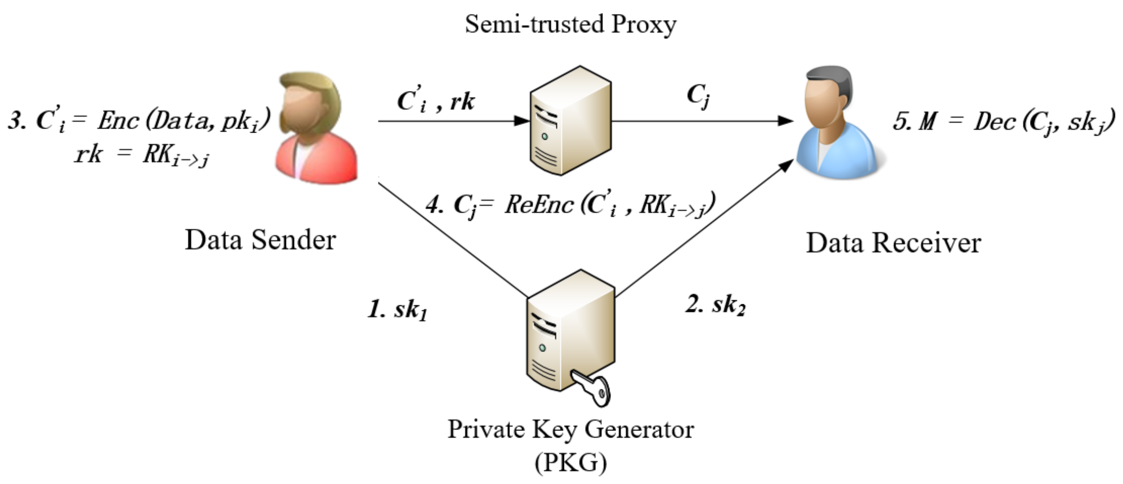

Fig. 1. PRE system model diagram

By utilizing the PRE primitive, Alice can transmit ciphertext and conversion key to proxy, and the proxy has no access to obtain the information or recognize the identity of Alice. Next, the proxy generates the conversion ciphertext. Finally, by using secret key, Bob is able to obtain the plaintext from the conversion ciphertext. In this way, data sender can keep identity anonymous in the re-encrypt process.

In our proxy re-encryption scheme, the voter's encrypted vote is re-encrypted by the proxy, and finally the candidate (delegated candidate) obtains vote by decrypting the vote ciphertext. In order to increase the openness and fairness of voting, the encrypted voting is decrypted jointly by the administrator and the candidates on the bulletin board. In this way, both the security and the count of votes can be ensured. At the same time, the direct secret information transmission (such as vote's private key) between voters, candidates, and the voting center server is avoided, thereby the privacy of voters is strengthened.

However, in proxy re-encryption, a semi-trusted proxy may also collude with candidates to obtain the voter's private key (the voter's identity information). For this collusion problem, we designed a collusion-resistant proxy re-encryption scheme. This scheme ensures that even if the candidate sends his private key to the proxy, the proxy cannot obtain the voter's information from the re-encrypted vote or the re-encryption key. Moreover, many previous proxy re-encryption schemes encrypt the data by relying on bilinear pairing, which brings heavy overhead. Our scheme avoids the use of bilinear pairs, thereby reducing the burden of computational and communication costs of the voting system.

The encryption schemes have achieved the function of voter privacy and vote content protection. However, there are still some problems in the voting system. These problems hinder the development of research and the advance of social justice. Electronic voting is expected to replace the cumbersome and no privacy guaranteed paper-based vote, and it has been researched over thirty years. Numerous protocols were proposed, and some of them have been implemented in real-world referenda such as Direct Recording Electronic (DRE) [15] systems and Helios [16,17] systems. Some DRE systems [18] have such an idea to provide variable receipts to the electorate. Yet this solution in ill-designed voting systems will leak voter's privacy and further not only pose a threat to the safety of voters but also break fairness of the election.

All of these phenomena underlined the importance of additional protection of voter privacy while providing verifiability to voters. Hence advanced security requirements called receipt-freeness and coercion-resistance are proposed. The notion of receipt-freeness was first proposed in 1994 [19]. Though their scheme was later proved unable to provide receipt-freeness by Hirt et al. [20]. Receipt-freeness refers to the incapability of a voter to prove that he voted in a certain approach to any attackers with remained verifiability. Generally there exist two cryptographic tools to achieve receipt-freeness: homomorphic encryption [20-24] and Mix-net [25-32].

Wen et al. [22] proposed the receipt freeness voting scheme named Masked Ballot. Their online voting scheme is achieved under a more practical physical assumption. The online voting scheme adapts Internet voting from any light-weight device that has access to the network. However, under their assumptions only receipt-freeness is possible. In addition, their scheme has a limitation, that is the voter needs to acquire a single-use mask before each election securely. Yu et al. [23] introduced a practical platform-independent secure and verifiable voting system. The advantage of their voting system is that the scheme suits any blockchain that supports the operation of the smart contract. There two main advantages of their scheme. The first point is taking advantage of a decentralized trust provided by the blockchain technology. In this way, the demand of a centralized trusted party to do the ballots tallying is removed. The second point is, their scheme also builds a practical platform independent secure e-voting protocol by key security primitives. However, their scheme cannot avoid requiring the administrator to upload the encryption of zero pool. Moreover, both the increases in the block size and chain length aggravate the time cost of retrieving for a certain block in the blockchain. In [24], the authors introduced a framework for practical and receipt-free remote voting. However, their scheme has two limitations. First, in both the registration phase and the vote casting phase, crypto calculations are needed, but the scheme has not provided concrete operations of ordinary voters. Second, due to the candidate representing value v, adversaries may force/bribe voters to change their votes. The author in [25] introduced some receipt-free voting schemes for large scale elections. One of the provided schemes demands the assistance of the voting commission. Meantime, a physical assumption and an untappable channel are also required. Although the other one does not need the assistance of the commission, a stronger physical assumption and a voting booth are indispensable. In paper [28], the authors proposed a secure hardware device for the voting system, called the tamper-resistant randomizer. They claimed this hardware device can replace the existence of third-party randomizer and untappable channel. In [30], a simple and efficient method to incorporate receipt-freeness in mix-net based electronic voting schemes is given. However, due to a large amount of computation for multiple mixers is required, the above voting schemes based on mix-net are mostly not efficient.

Based on the above characteristics of the voting system, we propose an improved proxy re-encryption scheme. Our proxy re-encryption scheme (1) uses dual public keys and dual private keys as the real identity and voting system identity, respectively. This method allows voters to hide their true information and vote anonymously. The disclosure of the identity of voters is prevented from the root. (2) Added the list of administrators. Each voter obtains a private key when registering, and the administrator generates a corresponding list of identity information to verify the identity of the voter during voting. (3) The administrator further randomly encrypts the re-encrypted vote to ensure that the voter cannot provide a receipt, the voter is not able to prove to a coercer what way he/she voted.

1.1 Contribution and Innovations

First, we proposed a voting system PRE scheme and showed how it works in voting system. From the perspective of cryptography, it is one of the most meaningful combinations of cryptographic protocols and practical applications. Protecting the privacy of the recipients' identities can be realized by the PRE method. Our proposal enjoys the properties simultaneously as below:

- Multiple security attributes: The scheme also has attributes such as unidirectional, key privacy, collusion-resistant.
- Verifiable: We combined the proxy re-encryption scheme and digital signature to realize a verifiable proxy re-encryption scheme.
- Scalable: Few cryptographic primitive is used in our scheme other than schnorr digital signature and PRE in order to uplift the efficiency of calculation and communication and hence enlarge the scale of the election. Our scheme does not have a paring cost. A trivial amount of overhead in communication cost and computation cost make our proposal can be used in various applications.
- Applied in voting system: In general, PRE schemes are previously used in cloud computing and distributed file systems. To the best of our knowledge, few studies in literature combined proxy re-encryption with electronic voting. In our voting protocol, we illustrate how it works to provide receipt-freeness and candidate-adaptiveness to voting systems.
- Security and efficiency: This paper provides concrete security proof chosen-ciphertext security, meanwhile, collusion-resistance is not ignored. The proxy cannot recover the data owner's private key by colluding with the data receiver. We also conduct and analyze the simulation experiment. The simulation experiment results show that this scheme, in the aspects of communication complexity and computation cost, both has a relatively good performance.
- Experimental comparison: We have run our scheme and the other compared articles on the experimental platform at the same time. Meantime, the experimental results are presented in multiple figures.

The organization of our paper is present below. Assumptions and scheme definitions are discussed in Section 2. The construction of our PRE is presented in Section 3 and a security proof of our proposal is analyzed in Section 4. In addition, this paper delineates how proxy re-encryption is used in voting system and we also give an analysis of voting requirements in Section 5. At last, the efficiency comparison and conclusion of our work is shown in Section 6 and Section 7.

2. Preliminaries

2.1 Schnorr Signature

Existentially unforgeable schnorr signature has system parameters $(\mathbb{G}, g, q, H(\cdot))$. In parameters, $\mathbb{G}$ is a finite cyclic group of prime order q. Let g be a generator of $\mathbb{G}$. $H(\cdot)$ is a cryptographic hash function, $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_q^*$.

- KeyGen $(1^\lambda) \to k$. This step generates a signing key $sk \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and verifying key $pk = g^{sk}$.
- Sign $(sk, m) \to \sigma$. sk and m are inputted, this algorithm generates $\sigma = (s, R)$, where $R = g^r$, $s = r + sk \cdot H(m||R) \bmod q$, and $r \overset{R}{\in} \mathbb{Z}_q^*$.

- Verify $(pk, m, \sigma) \to m$. On inputting pk, m, and σ, the verifying algorithm returns 1, if $g^s = R \cdot pk^{H(m||R)}$; otherwise, it outputs 0.

2.2 Complexity Assumption

The security of our proposed scheme is based on the Decisional Diffie–Hellman (DDH) assumption. The DDH assumption in $(q, \mathbb{G}, \mathbb{G}_T)$ is consisted of following de nation: On input a tuple (g, g^a, g^b, T), the algorithm returns 1 if $T = g^{ab}$ otherwise output 0. Adversary $\mathcal{A}$' advantage in solving the DDH problem is presented in the following:

$$\text{Adv}_{\mathcal{A}}^{\text{DCDH}} = |\Pr[\mathcal{A}(g, g^a, g^b, g^{ab} = 1)] - \Pr[\mathcal{A}(g, g^a, g^b, T = 1)]|$$

T is an element randomly selected from $\mathbb{G}_T$, where $g \in \mathbb{G}$, $a, b \in \mathbb{Z}_q^*$. The DDH problem is difficult in the bilinear mapping, if there not exist a probabilistic polynomial-time algorithm $\mathcal{A}$ has a non-negligible advantage in solving DDH problem.

2.3 Scheme Definition

Table 1. Description of notations

Symbol	Description
par	Public parameter
H_1 , H_2	One way hash function
$\mathbb{Z}_q^*$	Multiplicative groups of integers of order *q*
$\mathbb{G}$	Cyclic group with prime order
$pk_{i,n}, sk_{i,n}$	The public key and private key of voter (n voters $(i = 1, \cdots, n)$)
$pk_{j,l}, sk_{j,l}$	The public key and private key of candidates (l candidates $(j = 1, \cdots, l)$)
S_v	Administrator generate S_v for the verification of voters' identity
CCA	Chosen ciphertext attack
CPA	Chosen plaintext attack

The voting PRE scheme of this paper contains the following algorithms (**Setup**, **KeyGen**, **Admini strator List**, **ReKeyGen**, **Enc**, **ReEnc**, **Dec**). For ease of description, some intuitive notations and abbreviations used in our proposed protocol are shown in Table 1.

- **Setup**$(1^k) \to$ (par): k as the security parameter is input, and global public parameters par are output and distributed to users.
- **KeyGen**(par) $\to SK$: The public parameter par is input, and a pair of key $(sk_{i,n}, pk_{i,n})$ is returned.
- **Administrator List**$(sk_{i,n}) \to List$: On inputting public parameter $sk_{i,n}$, this algorithm generates voting registrants list $[x_i]$, candidate list $[x_j]$ and verification list.
- **ReKeyGen**(par, $sk_{i,n}, pk_{j,l}) \to RK$: Operated by user i, takes the secret key $sk_{i,n}$, public parameter par, and the public key $pk_{j,l}$ as input, the re-encryption key RK is generated.
- **Enc**(par, $pk_{i,n}, m) \to \sigma$: User's public key $pk_{i,n}$, a message m and public parameters par is input, and a ciphertext σ under user's public key $pk_{i,n}$ is returned. The σ' can and only can be re-encrypted by RK.
- **ReEnc**$(\sigma, RK) \to \sigma'$: Operated by proxy, on inputting the ciphertext σ', conversion key RK, the encrypted ciphertext σ' is returned.
- **Dec**$(\sigma', sk_{j,l}) \to m$: Operated by user j, on input the ciphertext σ', the plaintext m is returned.

Finally, for any par, $m \in \mathbb{G}_T$, $sk_{i,n} \in \mathbb{Z}_q^*$, our proposal needs to meet following requirements:

$$\mathbf{Dec}(\mathbf{ReEnc}(\mathbf{Enc}(\text{par}, m, pk_{i,n}),\ \mathbf{ReKeyGen}(\text{par}, sk_{i,n},\ pk_{j,l})),\ sk_{j,l}) = M.$$

2.4 System Model

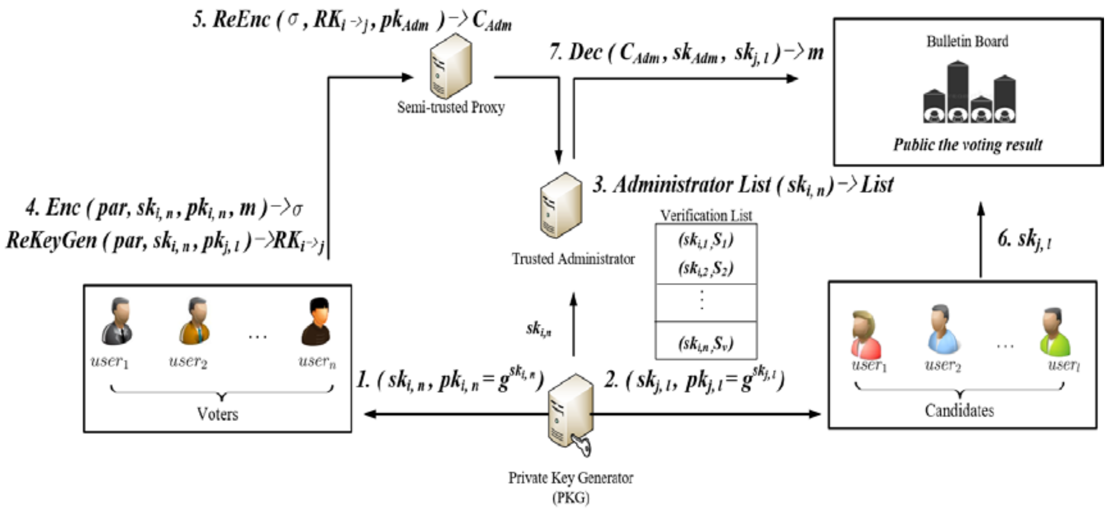
5. ReEnc (σ , RK_{i->j}, pk_Adm) -> C_Adm
Semi-trusted Proxy
7. Dec (C_Adm , sk_Adm , sk_{j, l}) -> m
Bulletin Board
Public the voting result
3. Administrator List (sk_{i, n}) -> List
Trusted Administrator
Verification List
(sk_{i,1}, S_1)
(sk_{i,2}, S_2)
(sk_{i,n}, S_v)
4. Enc (par, sk_{i, n}, pk_{i, n}, m) -> σ
ReKeyGen (par, sk_{i, n}, pk_{j, l}) -> RK_{i->j}
6. sk_{j, l}
sk_{i,n}
user_1
user_2
user_n
Voters
1. (sk_{i, n}, pk_{i, n} = g^{sk_{i, n}})
2. (sk_{j, l}, pk_{j, l} = g^{sk_{j, l}})
user_1
user_2
user_l
Candidates
Private Key Generator
(PKG)

Fig. 2. Proxy re-encryption for voting system model

As shown in **Fig. 2**, there are several entities in the voting system, including an administrator, n voters $(i = 1, \cdots, n)$, l candidates $(j = 1, \cdots, l)$, a proxy, a converter. Next, we detail the responsibilities of each entity.

Voters: The voters first send a registration request to the administrator and then receive a key pair from the private key generator (PKG). In the voting process, the voters encrypted the vote by the public key of the administrator. Then voters encrypt the first-level ciphertext to generate ciphertext and then send ciphertext and re-encryption key to proxy. The conversion key is the crucial part of voting. The voter should generate re-encryption key by his/her own private key and the public key of the candidate who he supported. Thus, the re-encrypted ciphertext can only be decrypted by the candidate who is voted.

Proxy: After received voter's conversion key and encryption ciphertext, the semi-trusted proxy then converts the vote and delivers the converted vote to the bulletin board. The user's identity can be kept secret, because the proxy cannot obtain the information of the plaintext and the private key. Thus, the identity of voters and candidates keep secret and be hidden by re-encryption. In this way, the identity of voters can be converted securely. The candidates cannot identify the voter identity of final vote.

Candidates: When the vote process end, all the candidates decrypt the vote on the bulletin board by his private key. Then these candidates publish their secret key on the Bulletin board so that all users are able to verify the vote result.

Private Key Generator (PKG): As a trusted entity, PKG is responsible for generating private keys for users. In voting system, the PKG generates the key pair and the verify key for verifying voters' identity.

Trusted Administrator: At the beginning of voting, the administrator generates voting registrants list $[x_i]$ for legitimate registered voters, candidate list $[x_j]$ and verification list. These registrant lists are kept secret to voters and candidates and stored in the administrator to verifying the identity of voters.

When the balloting process is ended, the administrator will witness that candidates decrypt the re-encrypted vote and record the count of votes for every candidate. After the candidates do the first-level decrypt for the vote, the administrator does the second-level decrypt for the vote by his secret key. There are some kinds of vote: positive vote ‘‘affirmative vote’’ or negative vote ‘‘dissenting vote’’, or “abstention vote”. The receipt-freeness and coercion-resistance can be achieved by this way, as the candidates cannot verify if the corrupted voters have voted as promised.

Normally, candidates can only obtain the re-encrypted vote in bulletin board with no knowledge of original vote (encryption ciphertext) or the identity of voter (voter’s secret key). Even candidates obtain voter’s encryption ciphertext (original vote) by in collusion with proxy or voters, they cannot decrypt the vote encrypted by administrator’s secret key or verify the voter’s vote types.

If the voter did not encrypt the vote by the administrator’s secret key, the vote will be canceled in administrator verify phase. The administrator is an important entity and it is responsible for verifying voters’ identity, publishing the list of candidates, collecting valid ballots, and announcing the vote result.

2.5 Security Model

Some important terms are defined as follows.

Uncorrupted-keys. When a corresponding private key of a public key cannot be exposed to the adversary, this kind of public key is set as the uncorrupted public key. K_{uc}^{list} is defined as the list of honest users, and the list contains the key pair of each honest user.

Corrupted-keys. When a corresponding private key of a public key can be attacked or compromised by anyone adversary, this kind of public key is defined as the corrupt public key. K_{c}^{list} is set as the list of corrupt public keys.

Definition. *Our scheme is CCA secure PRE scheme if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ wins the game with a negligible advantage. In the following game, $\mathcal{A}$ plays the role of adversary and $\mathcal{C}$ plays the role of challenger.*

Key generations. $\mathcal{C}$ creates the keys as below. $\mathcal{C}$ runs the **Public key generation oracle** to create key pair. Then, $\mathcal{C}$ adds the key pair to K_{c}^{list} and K_{uc}^{list}. $\mathcal{A}$ obtains the public key from K_{c}^{list} and K_{uc}^{list} and secret key from K_{c}^{list}.

Phase 1. Adversary $\mathcal{A}$ sends queries to O_{SK}, O_{RK}, O_{RE}, O_{DEC}.

- **Secret key generation oracle** O_{SK}: $pk_i = (pk_{i,1}, pk_{i,2})$ is inputted, and $\mathcal{C}$ retrieves $(pk_{i,1}, pk_{i,2}, sk_{i,1}, sk_{i,2})$ for a corrupted public key in K_{c}^{list}. For a uncorrupted user, $\mathcal{C}$ retrieves $(pk_{i,1}, pk_{i,2}, sk_{i,1}, sk_{i,2})$ in K_{uc}^{list}, and outputs $sk_{i,1}$, $sk_{i,2}$ to the adversary; otherwise, $\mathcal{C}$ outputs $\perp$.
- **Re-encryption key generation oracle** O_{RK}: If $g^S = A^{H_3(A||B||C_2||U_2)} \cdot C$, this oracle continues; otherwise, this oracle returns $\perp$ and halts. On inputting (pk_i, pk_j), and pk_i and pk_j are both from O_{PK}. Oracle takes (pk_i, pk_j) as input and returns $rk = (rk_1, rk_2, rk_3)$ to $\mathcal{A}$.
- **Re-encryption oracle** O_{RE}: On inputting (pk_i, pk_j, σ), $\mathcal{C}$ returns a re-encryption ciphertext $\mathbf{ReEnc}(\sigma, O_{RK}(pk_i, pk_j))$.
- **Decryption oracle** O_{DEC}: On input pk and C, this oracle finds the corresponding private key of pk. Then, this oracle decrypts C and returns the plaintext.
- **Challenge.** When $\mathcal{A}$ finishes the queries of Phase 1, a public key pk_i^* and two equal-length messages $m_0, m_1 \in \{0,1\}$ are sent to $\mathcal{C}$. Algorithm $\mathcal{A}$ recovers tuple (pk_i, sk_i, c) from K^{list}. $\mathcal{C}$ picks $d \overset{R}{\in} \{0,1\}$ and sends the challenge ciphertext generated by the pk_i^* and m_d to $\mathcal{A}$.

Phase 2. Adversary $\mathcal{A}$ continues to issue queries as in Phase 1, under extra conditions.

Guess. Eventually, adversary $\mathcal{A}$ outputs a guess $d \in \{0,1\}$ to $\mathcal{C}$.

3. An Efficient Proxy Re-Encryption Construction in Voting System

Here we describe our construction for the efficient proxy re-encryption scheme in voting system.

- **Setup**$(1^k) \to \mathsf{par}$: Select prime q, and k is the security parameter. g is a generator of group G. $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, and $H_2 : \mathbb{G} \to \mathbb{Z}_q^*$. The global public parameters are par :$(q, \mathbb{G}, g, H_1, H_2)$.
- **KeyGen**$(\mathsf{par}) \to SK$: Takes public parameter par as input. This algorithm generates $sk_{i,n} = (sk_{i,n,1}, sk_{i,n,2})$, $pk_{i,n} = (pk_{i,n,1} = g^{sk_{i,n,1}}, pk_{i,n,2} = g^{sk_{i,n,2}})$. Select $z_1, z_2 \overset{R}{\in} \mathbb{Z}_q^*$, set $sk_{i,n,1} = z_1 + sk_{i,n,2} \cdot z_2 \bmod q$. Administrator key pair (pk_{Adm}, sk_{Adm}) is also generated.
- **Administrator List**$(sk_{i,n}) \to List$: This step is operated by administrator, and administrator sets $S_v = (U_1, U_2, U_3) = (g^{H_1(x_i||0)}, g^{H_1(sk_{i,n}||1)}, g^{H_1(x_i||0)+H_1(sk_{i,n}||1)})$, where $x_i \overset{R}{\in} \mathbb{Z}_q^*$, $x_j \overset{R}{\in} \mathbb{Z}_q^*$. The administrator holds U_1, U_3 and keeps them confidential to any other user. This algorithm generates $S_v = (U_1, U_2, U_3) = (g^{H_1(x_i||0)}, g^{H_1(sk_{i,n}||1)}, g^{H_1(x_i||0)+H_1(sk_{i,n}||1)})$.
- **ReKeyGen**$(\mathsf{par}, sk_{i,n}, pk_{j,l}) \to RK_{i\to j}$: On input user i' s secret key $sk_{i,n} = (sk_{i,n,1}, sk_{i,n,2})$ and user j's public key $pk_{j,l} = (pk_{j,l,1}, pk_{j,l,2})$ where there have l campaigners, $l \in \{1, \cdots, l\}$, the RK is created as follows:
  1. Select $v \overset{R}{\in} \mathbb{Z}_q^*$, compute $Z = pk_{j,l,2}^v$,
  2. $rk_1 = z_1$, $rk_2 = H_2(Z^{sk_{i,n,1}}) \cdot z_2 \bmod q$,
  3. $rk_3 = pk_{i,n,1}^v$,
  4. Return $RK_{i\to j} = (rk_1, rk_2, rk_3)$.
- **Enc**$(\mathsf{par}, pk_{i,n}, U_2, m) \to \sigma$: Given user i's secret key and public key, this protocol works as follows. The key pair is $(pk_{i,n}, sk_{i,n})$, $n \in \{1, 2, \cdots, n\}$.
  1. Select $r, r' \overset{R}{\in} \mathbb{Z}_q^*$,
  2. $A = g^r$, $B = g^{r'}$, $C_1 = pk_{i,n,2}^r$, $C_2 = pk_{i,n,1}^r \cdot m$,
  3. $U_2 = g^{H_1(sk_{i,n}||1)}$, $S = H_2(A||B||C_1||C_2||U_2) \cdot r + r' \bmod q$,
  4. Output the ciphertext $\sigma = (A, B, C_1, C_2, U_2, S)$.
- **ReEnc**$(\sigma, RK_{i\to j}, pk_{Adm}) \to C_{Adm}$: On input $RK_{i\to j} = (rk_1, rk_2, rk_3)$, an original ciphertext $\sigma = (A, B, C_1, C_2, U_2, S)$, and administrator's pk_{Adm}, this protocol operates as follow.
  1. The proxy verifies the relation $U_3 = U_1 \cdot U_2$ and sends U_2 to the administrator. If the administrator returns true, goes to the next steps; otherwise, outputs $\perp$ and halts.
  2. If $g^S = A^{H_2(A||B||C_1||C_2||U_2)} \cdot B$, the algorithm continues to perform the following operations, otherwise, output $\perp$ and halt.
  3. Compute $C_1' = A^{rk1} = g^{rz_1}$, $C_2' = C_1 = pk_{i,n,2}^r$, $C_3' = C_2 = pk_{i,n,1}^r \cdot m$, $C_4' = rk_2 = H_2(Z^{sk_{i,n,1}}) \cdot z_2 \bmod q$, $C_5' = rk_3 = pk_{i,n,1}^v$. The transformed ciphertext as follows: $\sigma' =(A, B, C_1', C_2', C_3', C_4', C_5', U_2, S)$.
  4. $Enc_{rd}(\sigma', pk_{Adm}) = C_{Adm}$.
- **Dec**$(C_{Adm}, sk_{Adm}, sk_{j,l}) \to m$: On input a ciphertext C_{Adm} administrator's secret key sk_{Adm}, and secret key $sk_{j,l}$.
  1. Decrypt the ciphertext C_{Adm} by the use of administrator's secret key sk_{Adm} and obtain re-encrypted ciphertext $\sigma' = (A, B, C_1', C_2', C_3', C_4', C_5', U_2, S)$.

2. If $g^S = A^{H_2(A||B||C_2'||C_3'||U_2)} \cdot B$ and $U_3 = U_1 \cdot U_2$ holds, i.e., the re-encrypted message and the key pair contained in the message have not been tampered with in transit. Then, proceed to the following step. Otherwise, this protocol halts.
3. Compute $z_2 = \frac{C_4'}{H_2(C_5'^{sk_{j,l,2}})} = \frac{H_2(Z^{sk_{i,n,1}}) \cdot z_2 \bmod q}{H_2(pk_{i,n,1}^{v(sk_{j,l,2})})}$, $m = \frac{C_3'}{C_1' \cdot C_2'^{z_2}} = \frac{pk_{i,n,1}^r \cdot m}{g^{rz_1} \cdot g^{r(sk_{i,n,2})z_2}}$.

Correctness:

$$\begin{aligned} z_2 &= \frac{C_4'}{H_2(C_5'^{sk_{j,l,2}})} \\ &= \frac{H_2(Z^{sk_{i,n,1}}) \cdot z_2 \bmod q}{H_2(pk_{i,n,1}^{v(sk_{j,l,2})})} \\ &= \frac{H_2((pk_{j,l,2}^v)^{sk_{i,n,1}}) \cdot z_2 \bmod q}{H_2(pk_{i,n,1}^{v(sk_{j,l,2})})} \end{aligned}$$

$$m = \frac{C_3'}{C_1' \cdot C_2'^{z_2}} = \frac{pk_{i,n,1}^r \cdot m}{g^{rz_1} \cdot g^{r(sk_{i,n,2})z_2}}$$

4. Security Proof

Theorem. *Assuming the DDH assumption holds, the PRE scheme proposed in this paper achieves CCA2 security in the random oracle model.*

Proof. If there exists an adversary $\mathcal{A}$ can break the CCA2 security of our proposal with a non-negligible advantage, then the algorithm $\mathcal{C}$ can be built to resolve DDH problem by the use of $\mathcal{A}$. On DDH input $\langle \mathbb{G} = \langle g \rangle, g^a, g^b, T \rangle$, $\mathcal{C}$ is constructed to decide if $T = g^{ab}$. The following random oracles is built by the challenger $\mathcal{C}$.

O_{H_1}: $\mathcal{C}$ checks whether (R, β) has occurred in the H_1^{list}. If the (R, β) exists, $\mathcal{C}$ responds $\mathcal{A}$ with β. Otherwise, select $\beta \leftarrow \mathbb{Z}_q^*$, put tuple (R, β) into the list H_1^{list} and respond with $H_1(R) = \beta$.

O_{H_2}: $\mathcal{C}$ checks whether tuple $H_2(R_1, R_2, R_3, R_4, R_5, \gamma)$ has occurred in the H_2^{list}. If the $H_2(R_1, R_2, R_3, R_4, R_5, \gamma)$ exists, and $\mathcal{C}$ responds $\mathcal{A}$ with γ. Otherwise, select $\gamma \overset{R}{\in} \mathbb{Z}_q^*$, put tuple $H_2(R_1, R_2, R_3, R_4, R_5, \gamma)$ into the list H_2^{list} and respond with $H_2(R_1||R_2||R_3||R_4||R_5) = \gamma$.

$\mathcal{C}$ maintains two initially empty lists, K_{uc}^{list} and R^{list}. These two lists store key pairs and re-encryption keys, individually.

Phase 1. Adversary $\mathcal{A}$ issues a series of queries which $\mathcal{C}$ answers $\mathcal{A}$ as follows:

Public key generation oracle O_{PK}: $\mathcal{C}$ generates the uncorrupted-keys and corrupted-keys as follows.

1. The corrupted user inputs the public key, and this oracle sets $pk_i = (pk_{i,1},$ and record $(pk_{i,1}, pk_{i,2}, sk_{i,1}, sk_{i,2})$ into the list K_c^{list}, where $sk_{i,1} = z_1 + sk_{i,2} \cdot z_2 \bmod q$ and z_1, z_2, $sk_{i,1}$, $sk_{i,2}$ is randomly selected from $\mathbb{Z}_q^*$.
2. The uncorrupted user inputs the public key, and this oracle sets $pk_i = ((g^a)^{sk_{i,1}}, (g^a)^{sk_{i,2}})$ and record $(pk_{i,1}, pk_{i,2}, sk_{i,1}, sk_{i,2})$ into the list K_{uc}^{list}, where $sk_{i,1}$ and $sk_{i,2}$ are random numbers from $\mathbb{Z}_q^*$.

Secret key generation oracle O_{SK}: When a uncorrupted user inputs $pk_i = (pk_{i,1}, pk_{i,2})$, $\mathcal{C}$ checks $(pk_{i,1}, pk_{i,2}, sk_{i,1}, sk_{i,2})$ in the list K_c^{list} and backs the corresponding $sk_{i,1}$, $sk_{i,2}$ to the

$\mathcal{A}$. $\mathcal{C}$ returns a random number in the same form as the private key as the corresponding private key of the corrupted public key.

Re-encryption key generation oracle O_{RK}: If $g^S = A^{H_3(A||B||C_1||C_2||U_2)} \cdot B$, this oracle continues with the subsequent operations; otherwise, the oracle returns $\perp$ and halts. On inputting (pk_i, pk_j), where $pk_i = (pk_{i,1}, pk_{i,2})$, $pk_j = (pk_{j,1}, pk_{j,2})$. pk_i and pk_j are both from O_{PK}.

1. When the corrupted pk_i is input, $\mathcal{C}$ retrieves the corresponding tuple of pk_i from K_c^{list}. Then, $\mathcal{C}$ generates sk_i by running the real execution operations in scheme.
2. When uncorrupted pk_i and pk_j is input, $\mathcal{C}$ retrieves the corresponding tuple of pk_i from K_{uc}^{list}, obtains tuple $(sk_{i,1}, sk_{i,2})$ satisfy with $sk_{i,1} = z_1 + sk_{i,2} \cdot z_2 \bmod q$, and then run operations in **ReKeyGen** to obtain the re-encryption key. Finally, $\mathcal{C}$ backs the generated re-encryption key. If there exists no such tuple, return $\perp$.
3. If pk_i is uncorrupted, and pk_j is corrupted, output ''failure" and aborts.
4. Return $RK_{i \to j} = (rk_1, rk_2, rk_3)$ to $\mathcal{C}$.

Re-encryption oracle O_{RE}: On inputting (pk_i, pk_j, σ), $\mathcal{C}$ returns **ReEnc** $(\sigma, O_{RK}(pk_i, pk_j))$.

Decryption oracle O_{DEC}: On inputting (pk_i, σ'), $\mathcal{C}$ replies as follows.

1. When the corrupted pk_i is input, $\mathcal{C}$ retrieves the corresponding tuple of sk_i from K_c^{list}. Then, $\mathcal{C}$ outputs decrypted results by running the real execution operations in the scheme.
2. When the uncorrupted pk_i and $\sigma' = (A, B, C_1, C_2, U_2)$ are input, $\mathcal{C}$ continues as below.
   a) Retrieve $(A, B, C_1, C_2, U_2, \gamma)$ in the table H_2 and (R, β) in the H_1. As the tuple can be found and it satisfy $g^S = A^{H_2(A||B||C_1||C_2||U_2)} \cdot C$ and $U_1 = U_2 \cdot g^{H_1(R)}$, $\mathcal{C}$ continues. If the tuple cannot be found, this oracle outputs $\perp$ and halts.
   b) When corrupted user inputs pk_j and associated private key sk_j, $\mathcal{C}$ returns $\mathbf{Dec}(\mathbf{ReEnc}(\sigma, O_{RK}(pk_i, pk_j)), sk_j)$.

Challenge. When the above queries are finished, $\mathcal{A}$ transmits $pk'_{i,n}$ and $m_0, m_1 \in \{0,1\}$ to $\mathcal{C}$.

In addition, the two messages m_0, m_1 are the same length. Algorithm $\mathcal{A}$ recovers tuple $(pk_{i,n} sk_{i,n}, c)$ from K^{list}. $\mathcal{C}$ picks $d \leftarrow \{0,1\}$ and the challenge ciphertext is created as below.

1. $C'_1 = g^{rz_1} = g^{bz_1}$
2. $C'_2 = pk^r_{i,n,2} = g^{ab \cdot sk_{i,n,2}} = (pk_{i,n,2})^b$
3. $C'_3 = pk^r_{i,n,1} \cdot m = g^{ab \cdot sk_{i,n,2}} \cdot m = pk^b_{i,n,1} \cdot m$
4. $C'_4 = H_4((pk^v_{j,l,2})^{sk_{i,n,1}}) \cdot z_2 \bmod q = H_4(((g^a)^{sk_{j,l,2}v})^{sk_{i,n,1}}) \cdot z_2 \bmod q$
5. $C'_5 = pk^v_{i,n,1} = g^{av}$.

Phase 2. Adversary $\mathcal{A}$ repeats queries under the conditions as below.

1. $O_{RK}(sk_i^*, pk_j)$ is only allowed if pk_j from K_{uc}^{list}.
2. If $\mathcal{A}$ issued $\mathbf{ReEnc}(\sigma, RK_{i \to j}) \to \sigma'$ where $RK_{i \to j}$ come from **ReKeyGen** $(\mathsf{par}, sk_i, pk_j) \to$
   RK. If the pk_j come from K_c^{list}, (sk_i, σ) cannot be a derivative of (sk_i^*, σ^*).
3. $O_{DE}(\sigma', sk_j)$ is only allowed if (sk_j, σ) is not a derivative of (sk_j^*, σ^*).

Guess. Eventually, adversary $\mathcal{A}$ returns a guess $d' \in \{0,1\}$ to $\mathcal{C}$. Notice that, when $T = g^{ab}$, the $\delta = (A^*, B^*, C_1'^*, C_2'^*, C_3'^*, C_4'^*, C_5'^*, U_2^*)$ equals $\mathbf{ReEnc}(\sigma, RK_{i \to j}) \to \sigma'$. The value $T = g^{ab}$ is replaced by a random value $T \in \mathbb{G}$. Therefore, the value d in challenge ciphertext cannot be guessed by $\mathcal{A}$ with a probability higher than $1/2$.

5. Our Proposed E-Voting Scheme

In this part, we present our voting scheme. Firstly, we illustrate the design model of the scheme, briefly overview the scheme and discuss the security requirements and assumptions. Secondly, we describe the complete voting scheme in detail. And then a security proof of the requirements of the voting scheme and efficiency analysis are presented.

5.1 Model of the Scheme

5.1.1 Overview of the Proposed E-Voting Scheme

As shown in **Fig. 3**, our proposed voting system includes a front-end client interface for voters' operations, an honest-but-curious proxy to re-encrypt voters' ballots, a bulletin board to inform or publicity any voting-related information and a decryption program for candidates to decrypt ballots toward them. From **Fig. 3**, the flow of information is clear: (1) via different kind of end devices, voters prepare their ballot and cast to proxy; (2) proxy takes voters' ballot as input and re-encrypt and publish the result to the bulletin board; (3) candidates can read from the public-accessed bulletin board and decrypt re-encrypted ballots toward them and publish with voters' unforgeable anonymous signature and send their secret key used in this election to bulletin board for auditing; (4) a script used for tallying can run on bulletin board to give out the statistics about an election.

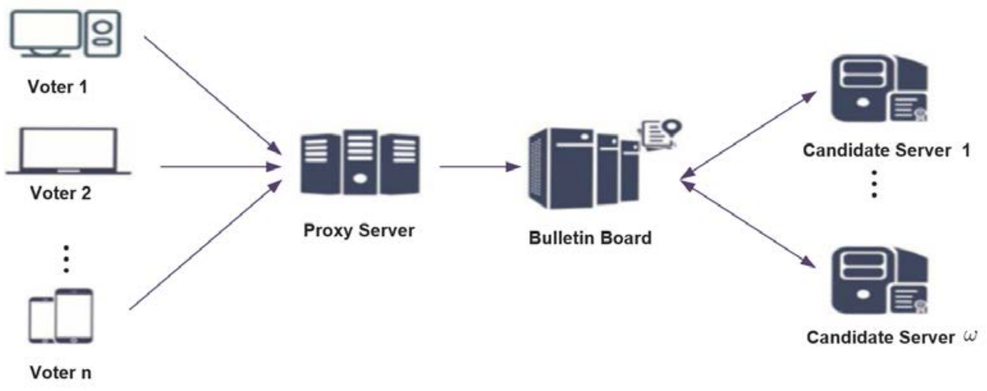

Fig. 3. Voting system model diagram

Through the flow of information in this voting system, our scheme is divided into six phases that respectively are system initialization, voter registration, ballot casting, proxy re-encryption, ballots opening and tallying, ballots verifying and auditing. Details about these six phases are discussed in section 5.2. The system communication diagram shown in **Fig. 4** illuminates communication sequence in our system. From this perspective, participants in this scheme are divided into five entities and the scheme can be divided into 16 steps. Respectively, definition and model of the entities will be discussed in section 5.1.2 and how these 16 steps work and form the six phases will also be described in section 5.2.

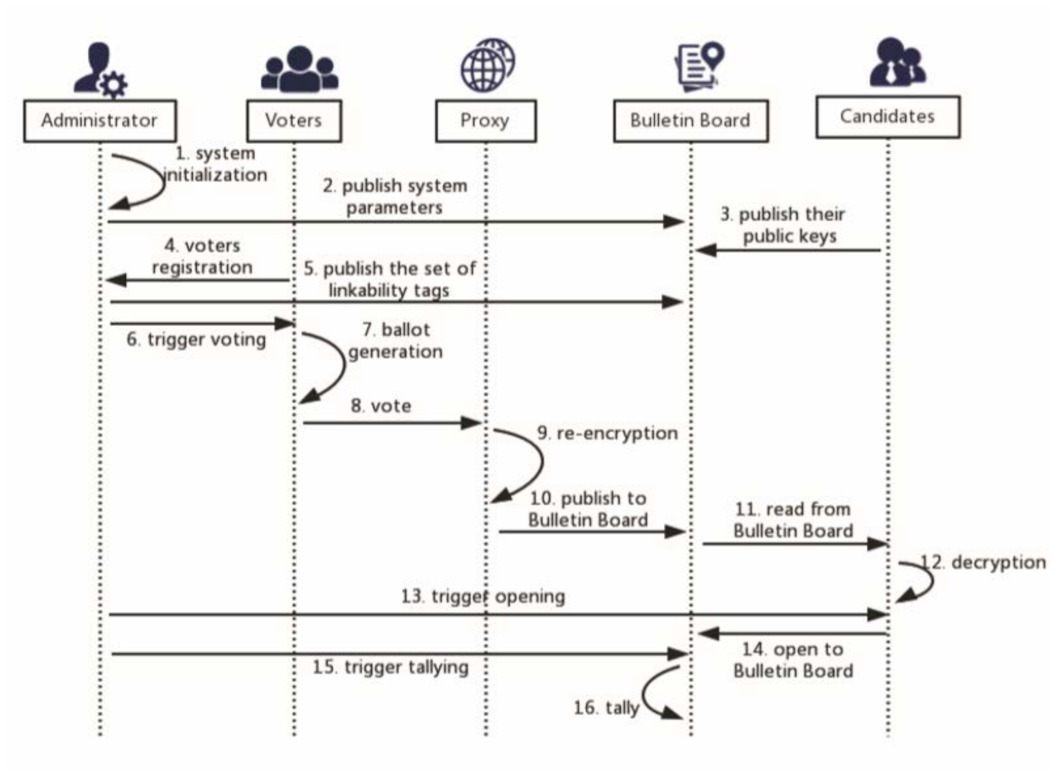

Fig. 4. Voting system model diagram

5.1.2 Entities in the E-Voting Process

Our proposal contains several entities: the administrator, the voters, the proxy, and the candidates. In addition, the bulletin board also serves as an entity to display the voting results. **Fig. 4** is the system communication diagram that illustrates the sequence of interactions between the entities in our scheme. The detailed corresponding definitions of the entities are shown as follows:

Administrator: He/she is responsible for initializing system parameters and organizing or controlling the voting process.

Voters: Voters is the main body of an election. They register into the voting system and then vote for candidates with their judgments. In the description of our scheme, we denote Alice as a voter.

Proxy: A proxy is an honest-but-curious authority that receives the encrypted ballots from incognito voters and publishes the re-encrypted ballots to the bulletin board. Honest-but-curious means that proxy will honestly follow the designed steps but try to delve into the relationship between the ballot and real-world identity of voters.

Bulletin board: It is a publicly accessible database that resists tampering attacks [33], which contains voting-related information, such as system parameters and re-encrypted votes. In some literature, it is also called the ballot box. To achieve the features above, a bulletin board can be implemented by a public block chain.

Candidates: Candidates play the role to be elected in the e-voting system. They compete to win the election and may try any approaches to get more ballots towards them on the bulletin board. In the description of our scheme, we denote Bob as a candidate.

5.1.3 Security Requirements and Assumptions

Here we enumerate the security requirements satisfied in our voting scheme. We divide the requirements into two classes: core requirements and additional requirements. Core requirements should be satisfied by all voting schemes and additional requirements are cherry-picked to fit distinct voting environments.

Core requirements:

Completeness. When all participants follow the protocol, the count of valid votes is accurate. **Privacy.** This protocol will ensure the anonymity of ballots, that is, the ballots cast by voters will not reveal the identity of voters.

Unreusability. The ballot of the same legal voter will not be counted twice.

Eligibility. Only the ballot of legal voters will be counted as the final number of votes.

Verifiability. The final result can be verified by any skeptics.

Additional requirement:

Vote-and-go. A voter can go online once his ballot is cast.

Robustness. The result of the election is proper in the assumption of a certain amount of malicious voters or partial failure of the system.

Efficiency. The computational overhead and communication overhead are not too huge in the proposed scheme. This feature allows voters to use a tablet or mobile phone to complete the vote.

Receipt-freeness. A voter is not able to prove to someone that he has finished voting for a certain person in some way.

Coercion-resistance. A voter cannot be coerced into voting in a certain approach.

In our system, we use proxy re-encryption to further help voters to vote fairly according to their wishes. In the voting system, the realization of fair voting is usually to ensure that no one can calculate part of the result before the end of the election. However, in some cases, candidates can obtain information on how many people voted for themselves during the election process, and can implement the corresponding campaign strategies based on the acquired. Therefore, we propose a new voting scheme to satisfy this environment called candidate adaptability. Before the voting begins, no voters can break the agreement to get any results. In the voting process, each candidate can only know the number of votes he obtained during the election process, but he cannot know the exact number of votes of other competitors.

Candidate-adaptiveness. In the voting process, each candidate can only obtain the number of votes he obtained during the election process, but he cannot know the exact number of votes of other competitors.

In our scheme, the proxy re-encrypt the vote, and then the re-encrypted ciphertext is random encrypted by the administrator. The random encrypted ciphertext is sent to the bulletin board as the vote. Finally, when publishing vote results, the random encrypted ciphertext is decrypted by the administrator and candidate jointly. In the above voting phase, only the administrator is aware of the relationship between the final ciphertext public in the bulletin board and the re-encrypted ciphertext of the voter. Although the candidate can obtain the ciphertext published in the bulletin board, the voter cannot prove to candidates that they have voted for that candidate. In addition, it is possible that the voter has voted the negative vote to the candidate in the voting encryption phase. In other words, even if the candidate confirms that the voter has voted for him/her, the candidate cannot confirm whether the vote is positive or negative. the encrypted vote is a meaningless and irregular text，thus candidates cannot recognize the specific content of the vote. This is also a huge advantage brought by the application of encryption technology to the voting system.

5.2 The Voting System

System initialization phase in **Fig. 4** The detailed processes of voting system are described as follows.

5.2.1 System Initialization and Voter Registration

Alice (1) sends send a registration request to the administrator and (2) then receives a key pair from the private key generator (PKG). (3) Meanwhile, the PKG sends this voter's verification key to the administrator.

Protocol 1: **Setup**$(1^k) \rightarrow \mathsf{par}$

Input : 1^k.
Output : par.

1. The system administrator random selects n registers as voters. PKG sets $pk_{i,n} = g^{sk_{i,n}}$. These key pairs are distributed to voters as their identity in voting system.
2. PKG sets $pk_{j,l} = g^{sk_{j,l}}$. These key pairs are distributed to candidates as their identity in voting system.
3. The administrator is responsible for publishing the parameters on the bulletin board and using the parameter setting system. i.e., the global public parameters: $\mathsf{par} : (q, \mathbb{G}, g, H_1, H_2)$.
4. Then every candidate publishes his/her public key $pk_{j,l}$ to the bulletin board.

Protocol 2: **KeyGen**$(\mathsf{par}) \rightarrow SK$

Input : par.
Output : SK.

1. The system administrator random selects n registers as voters. PKG picks $sk_{i,n} \overset{R}{\in} \mathbb{Z}_q^*$ and sets $(sk_{i,n}, pk_{i,n} = g^{sk_{i,n}})$. These key pairs are distributed to voters as their identity in voting system.
2. PKG picks $sk_{j,l} \overset{R}{\in} \mathbb{Z}_q^*$ and sets $(sk_{j,l}, pk_{j,l} = g^{sk_{j,l}})$. These key pairs are distributed to candidates as their identity in voting system.
3. This algorithm generates $(sk_{i,n}, pk_{i,n})$.

5.2.2 Ballot Casting

After all legitimate users' registration or the overpassed deadline of registration phase, system administrator broadcasts to all registered voters to enable voting. To trigger this phase, several approaches can be used by administrator including (1) sending e-mail to all registered voters, or (2) enabling the voting function in client interface of registered voters.

One voter, say Alice, prepares her ballot as follows: Alice (1) chooses one of vote types (positive vote or negative vote) as voting message, then (2) finds the corresponding public key $pk_{j,l}$ of candidate on bulletin board and generates the conversion key **ReKeyGen**$(\mathsf{par}, sk_{i,n}, pk_{j,l}) \rightarrow RK$, (3) invokes re-encryption protocol to generate the ciphertext **Enc**$(\mathsf{par}, pk_{i,n}, m) \rightarrow \sigma$. The completed ballot contains the ciphertext σ and the re-encryption key RK. Ballot prepared, Alice casts it to proxy via an anonymous communication channel. In our proposed scheme, after his/her voting, the voter needs not to take any computation or interaction. That is to say that our scheme achieves the so-called feature of Vote-And-Go.

Protocol 3: **Administrator List**$(sk_{i,n}) \rightarrow List$

Input : $sk_{i,n}$.

Output : $List$.

1. In this part, administrator generates voting registrants list $[x_i]$ for legitimate registered voter, where $x_i \overset{R}{\in} \mathbb{Z}_q^*$, Administrator generates candidate list $[x_j]$, where $x_j \overset{R}{\in} \mathbb{Z}_q^*$.
2. These registrants' ID should be kept secret to voters and candidates and be stored in the trust system administrator to verify the identity of voters. These secret values stored on system administrator's verification list.
3. These registrant ID are kept secret to voters and candidates, the voters only know $g^{H_1(sk_{i,n}||1)}$ part and don't know the part $g^{H_1(x_i||0)}$. This algorithm generates $SK = (x_i \overset{R}{\in} \mathbb{Z}_q^*)$ and $S_v = (U_1, U_2, U_3) = (g^{H_1(x_i||0)}, g^{H_1(sk_{i,n}||1)}, g^{H_1(x_i||0)+H_1(sk_{i,n}||1)})$. A list will store these key pairs for checking the voter's identity in system administrator.

Protocol 4: **ReKeyGen**$(\mathsf{par}, sk_{i,n}, pk_{j,l}) \rightarrow RK_{i\rightarrow j}$

Input : $\mathsf{par}, sk_{i,n}, pk_{j,l}$.

Output : $RK_{i\rightarrow j}$.

1. In this part, the voter selects his/her ideal candidate by reading the information on the bulletin board, and generates corresponding re-encryption key by using selected candidate's voting public key.
2. Return $RK_{i\rightarrow j} = (rk_1, rk_2, rk_3)$ as section 3.

Protocol 5: **Enc**$(\mathsf{par}, pk_{i,n}, m) \rightarrow \sigma$

Input : $\mathsf{par}, pk_{i,n}, m$.

Output : σ.

1. Given the public key of user i, the protocol works as follows.
2. This part is about trust voting system administrator give every voter a signature to guarantee their legitimate voting identity.
3. To limit the right of voting to n legitimate users, system administrator will distribute to signed user an identity. The key pair is $(pk_{i,n}, sk_{i,n}),\ n \in \{1, 2, \cdots, n\}$.
4. Output the ciphertext $\sigma = (A, B, C_1, C_2, U_2, S)$ as section 3.

Protocol 6: **ReEnc**$(\sigma, pk_{Adm}, RK_{i\rightarrow j}) \rightarrow C_{Adm}$

Input : σ, pk_{Adm}, RK.

Output : C_{Adm}.

1. $RK_{i\rightarrow j} = (rk_1, rk_2, rk_3)$ and $\sigma = (A, B, C_1, C_2, U_2, S)$ are inputted.
2. The transformed ciphertext $\sigma' = (A, B, C_1', C_2', C_3', C_4', C_5', U_2, S)$ is generated as section 3.
3. The administrator generates random ciphertext $Enc(\sigma', pk_{Adm}) = C_{Adm}$.

5.2.3 Ballot Re-encryption

On receiving voter's message, supposing ballot from Alice, proxy calls re-encryption algorithm in our scheme as: (1) firstly it checks the identity verification, and if it is hold, (2) it inputs ciphertext σ' and conversion key RK and computes the re-encrypted ciphertext $\mathbf{ReEnc}(\sigma, RK) \rightarrow \sigma'$, and (3) it sends the computation result σ' to the bulletin board with a timestamp $time_i$. If the check in step (1) of proxy in this phase fails, proxy will send the original message σ' from voter $voter_t$ to bulletin board with an error message suggesting $voter_t$ cast a valid ballot.

5.2.4 Ballots Opening and Tallying

Since the bulletin board is public accessed, candidates can keep an eye on re-encrypted ballots over it. Once a ballot is recorded on bulletin board, candidate Bob could try to decrypt the record with his private key $sk_{j,l}$ and find out whether the ballot is voting to him, i.e. the plaintext of decrypted message is his encoded identity. In the decryption process, Bob (1) checks the identity verification, and (2) outputs the plaintext m by invoking the decryption algorithm $\mathbf{Dec}(\sigma^{'}, sk_{j,l}) \to m$. If Bob's public key successfully decrypted it, Bob would know that he has received one vote, and if not, Bob would only know that one of his opponents has got this vote without explicit knowledge that who has.

If date of opening is due, system administrator will send the messages separated in different stages to the bulletin board and requests the candidate to publish their private key to the bulletin board. The private keys of candidates form a set on bulletin board as $sk_{j,l} = \{sk_{j,1}, sk_{j,2}, \ldots, sk_{j,l}\}$. After the above operations, the counting script on the bulletin board will be triggered to calculate the number of votes for each candidate.

This script can be implemented as smart contract over blockchain-based bulletin board.

Protocol 7: $\mathbf{Dec}(C_{Adm}, sk_{Adm}, sk_{j,l}) \to m$

Input : $C_{Adm}, sk_{Adm}, sk_{j,l}$.

Output :M.

1. The candidate sends the U_2 to the administrator for verifying the relation $U_1 \cdot U_2 = U_3$. If the administrator returns true, goes to the next steps; otherwise, outputs $\perp$ and halts.
2. Administrator checks the identity of the voter by using its voting system secret key and real secret key's variant in system administrator's verification list. If the re-encrypted ciphertext satisfies the relation, that means the voter is not only a legitimate voter (legal registrant) but also voter himself/herself (not an imposter).
3. The administrator decrypts the m in the bulletin board and gets the vote type (positive vote or negative vote).
4. After the decrypt process in the bulletin board, the message $(m, U_2, sk_{j,l})$ will be sent to administrator as a proof of being voted. If U_2 has already been queried, that mean the same voter votes for two times, administrator scraps the vote and return $\perp$; otherwise, go to the next steps;
5. If the U_2 can't satisfy the relation $U_3 = U_1 \cdot U_2$, administrator scraps the vote and return $\perp$; otherwise, administrator increases $pk_{j,l}$'s vote.
6. After the vote, the information $(sk_{i,n}, pk_{j,l}, \sigma')$ put on the bulletin board. Thus, vote results can be verified by any skeptic.

5.2.5 Ballots Verifying and Auditing

After counting all the votes, the elector and any skeptical voters can conduct additional audits. By verifying the candidate's private key set posted on the bulletin board and the re-encrypted ballot, the elector and any skeptical voters can review the authenticity of the vote. If the decrypted message does not match any value in the candidate slate, it will be regarded as a dummy vote and will not count into ballot pool of any candidate.

6. Efficiency Comparison

Table 2, **3**, and **Table 4** show the comparison of our scheme between several other PRE schemes. The comparison items include properties and costs.

Some important properties related to proxy re-encryption schemes are compared and listed in **Table 2**, such as security and hard problem and collusion-resistance. In the table, the symbol '$\surd$' will be given in the corresponding item as the scheme has this property. The symbol '$\times$' will be given in the corresponding item as the scheme has not this property. In **Table 2**, "weak" indicates that the scheme cannot achieve complete resistance to collusion attacks. The proxy can obtain the underlying encrypted messages through collusion attacks.

In **Table 3**, the t_p signifies a pairing executing time, t_s and t_{e1} signify a scalar multiplication and a exponentiation executing time in G_1, respectively; t_e denotes a exponentiation in G_t; Sig represents the operation of one-time signature, and Ver represents the operation of verification; k_1 , k_2, k_3, k_4 and n represent a security parameter, the bit-length of $\{0,1\}^{k_1}$, $\{0,1\}^{k_2}$, $\{0,1\}^{k_3}$, $\{0,1\}^{k_4}$ and $\{0,\cdots n\}$, respectively.

Table 2. A comparison of PRE schemes

Method	**Security**	**Hard problem**	**Collusion-resistant**
[14]	CPA	*DBDH*	weak
[34]-1	CPA	*q-DBDHI*	weak
[34]-2	CPA	*e-DBDH*	weak
[35]-1	CCA	*3-qDBDH*	$\surd$
[35]-2	CCA	*4-qDBDH*	$\surd$
[36]	CCA	*6-AmDBDH*	$\surd$
[37]	CPA	-	$\surd$
[38]	CCA	*DBDH and BDH*	-
[39]	CCA	*DBDH and CDH*	-
[40]	CCA	*DBDH*	$\surd$
[41]	CPA	-	$\surd$
[42]	CCA	*DBDH*	$\surd$
[43]	CCA	*DBDH*	$\surd$
Ours	CCA	*DDH*	$\surd$

Table 3. Computation cost comparison

Method	**Enc**	**ReEnc**	**Dec**
[14]	t_s+2t_e	t_p	t_p
[34]-1	t_s+2t_e	t_p	t_p
[34]-2	t_s+2t_e	t_p	t_p
[35]-1	t_p+5t_s+t_e+Sig	2t_p+4t_s+Ver	5t_p+t_s+ t_e +Ver
[35]-2	t_p+5t_s+t_e+Sig	2t_p+4t_s+Ver	5t_p+t_s+ t_e +Ver
[36]	t_p+7t_s	4t_p+5t_s	8t_p+3t_s+ t_e
[37]	2t_{e1}+t_e+ t_p	4t_p+3t_e	t_p+t_e
[38]	2t_p+t_{e1} +t_e	-	t_p+t_{e1}

[39]	$2t_p+3t_{e1}+t_e$	-	$2t_p+t_{e1}$
[40]	$3t_e+t_p+8t_{e1}$	$2t_p+3t_{e1}$	t_p+t_{e1}
[41]	$te+t_p+3t_{e1}$	t_p+t_{e1}	$2t_p+te+5t_{e1}$
[42]	$te+t_p+2t_{e1}$	$3t_p+7t_{e1}$	$3t_p+2te+t_{e1}$
[43]	$te+t_p+3t_{e1}$	t_p+5t_{e1}	$2t_p+2te+t_{e1}$
Ours	t_s+5t_{e1}	t_s+3t_{e1}	$4t_s+4t_{e1}$

Table 4. Communication complexity comparison

Method	PK	SK	RK lengthen	Original lengthen	Transformed lengthen
[14]	2|G|	$2\|Z_q\|$	|G|	$\|G\|+3\|G_t\|$	$2\|G_t\|$
[34]-1	|G|	$\|Z_q\|$	|G|	$\|G\|+2\|G_t\|$	$2\|G_t\|$
[34]-2	$\|G\|+\|G_t\|$	$2\|Z_q\|$	|G|	$\|G\|+3\|G_t\|$	$2\|G_t\|$
[35]-1	|G|	$\|Z_q\|$	|G|	$4\|G\|+\|G_t\|+\|k_s\|+\|\sigma_s\|$	$4\|G\|+\|G_t\|+\|k_s\|+\|\sigma_s\|$
[35]-2	|G|	$\|Z_q\|$	|G|	$4\|G\|+\|G_t\|+\|k_s\|+\|\sigma_s\|+\|T\|$	$4\|G\|+\|G_t\|+\|k_s\|+\|\sigma_s\|+\|T\|$
[36]	3|G|	$2\|Z_q\|$	|G|	$3\|G\|+\|G_t\|+\|Z_q\|$	$2\|G\|+\|SymEnc\|$
[37]	$\|G\|+\|G_t\|$	$2\|Z_q\|$	$2\|G\|+2\|G_t\|$	$2\|G\|+\|G_t\|$	$\|G_t\|$
[38]	|G|	|G|	-	$\|G_t\|+\|SymEnc\|$	-
[39]	|G|	$\|Z_q\|+\|G\|$	-	$\|G\|+\|G_t\|+\|k_1\|$	-
[40]	3|G|	$3\|Z_q\|$	$\|G\|+3\|Z_q\|$	$2\|Z_q\|+3\|G\|+\|G_t\|$	$2\|Z_q\|+3\|G\|+\|G_t\|$
[41]	$\|k_2\|$	|G|	$2\|G\|+n\|Z_q\|$	$2\|G\|+\|G_t\|$	$n\|Z_q\|+2\|G\|+\|G_t\|$
[42]	2|G|	$2\|Z_q\|$	4|G|	$3\|G\|+\|G_t\|+\|k_3\|$	$3\|G\|+\|G_t\|+\|k_3\|$
[43]	|G|	$\|Z_q\|$	4|G|	$3\|G\|+\|G_t\|+\|k_4\|$	$3\|G\|+\|G_t\|+\|k_4\|$
Ours	2|G|	$2\|Z_q\|$	$\|G\|+2\|Z_q\|$	$\|Z_q\|+5\|G\|$	$2\|Z_q\|+7\|G\|$

In **Table 4**, $|G|$, $|G_t|$ and $|Z_q|$ signify the length of an element in G, G_t and Z_q, respectively. Moreover, $|k_s|$, $|\sigma_s|$, $|T|$, $|M|$, $|SymEnc|$ signify the length of one-time signing key, signature, timestamp, the length of message $m \in M$, the one-time symmetric encryption ciphertext; l signifies the size of keyword set in some keyword search schemes, respectively. According to the comparison results of the table, the detailed analysis is as follows:

(1) In **Table 2**, we have present the comparison with recently proposed research schemes from security properties. Schemes [14], [34-36] are some representative classic proxy re-encryption schemes. In addition, to further analyze the advantage of our encryption schemes in the e-voting system, some applied in voting system encryption schemes [37-39] are also compared. Meantime, the latest proxy re-encryption schemes [40-43] are also compared to further justify that the results of our proposed research are better. From the compared result in **Table 2**, it can be noticed that some PRE or e-voting schemes [14, 34, 37, 41] only achieve CPA or not provided the standard security proof. Our scheme achieves a higher security level (CCA security) based on the DDH hard problem.

(2) Compared computation cost with PRE schemes and e-voting schemes in **Table 3**, our scheme does not bear paring computation cost and our scheme does not have ‘‘Dec(first

level)” due to the demand for the voting system. In our scheme, some hash operations are used in the protocol to protect the security of secret keys or verify the integrity of transformed data. In some schemes [36, 37, 40, 41, 42, 43], the data integrity is verified by the bilinear pairing operations. It is worth mentioning that the computational costs of the hash operation in our scheme which could be calculated much faster than bilinear pairing in other schemes.

(3) The result of communication complexity comparison can be find in **Table 4**. Compared with PRE schemes [14, 34, 35, 36, 37, 40, 41, 42, 43], our scheme a trivial amount of overhead in communication cost. For the communication cost of re-encryption key, our scheme only cost one length of an element in G and two length of an element in Z_q. Our proxy re-encryption scheme uses dual public keys and dual private keys as the real identity and voting system identity, respectively. The true identity information of voters is hidden and the voter is able to vote anonymously. This cost is considerably lower than latest PRE schemes [37, 40, 41, 42, 43] and marginally higher than the some of classic PRE schemes for the better security and data integrity. For the communication cost of or original lengthen and transformed lengthen, our scheme does not bear paring communication cost G_t in ciphertext and the slight high cost in the element in G is to verify the data integrity. Compared with other encryption schemes used in e-voting system [37-39], our scheme is light-weight to achieve re-encryption function and thus, our scheme can be applied in both mobile devices and desktops. Therefore, our scheme is more suitable for the voting system application requiring low overhead and re-encrypt function. In conclusion, our scheme gnerates a trivial amount of overhead in communication cost.

(4) Statistical analysis: Through the method of statistical analysis, we further compare the schemes in **Table 3**. The total “Enc” algorithms overhead of the schemes compared in **Table 3** is 295.376 ms, and the total “ReEnc” algorithms cost 334.3048 ms. The total cost of “Dec” algorithms of compared schemes in **Table 3** cost 475.319 ms. The average cost of “ Enc” algorithms for all schemes is 21.098 ms, and the average cost of “ ReEnc” algorithms for all schemes is 27.858 ms. The average cost of “Dec” algorithms for all schemes is 33.95142 ms. It can be clearly seen that the cost of our scheme in the encryption algorithm, re-encryption algorithm, and decryption algorithm is much lower than the average cost of the compared schemes.

(5) We simulated a real environment using a desktop computer equipped with an Intel Core i7-7700 processor at 3.60 GHz, 8 GB of memory, and Windows 10. We implemented our code in Microsoft Visual C++ 6.0 with the PBC library. To offer appropriate security, we used a PBC Type A pairing, which is constructed on the elliptic curve $y^2 \equiv x^3+x \bmod p$ for some prime $p \equiv 3 \bmod 4$ along with an embedding degree of 2. The detailed experimental results of our scheme and other schemes are shown in **Fig. 5**, **Fig. 6**, and **Fig. 7**.

(6) **Fig. 5** shows the computation time needed to operate the Enc algorithm in our scheme and the compared PRE schemes and e-voting schemes. It is seen that, in our scheme, the required time to produce the original ciphertext is slightly less than [14, 34, 36] and far less than the schemes [37, 38, 39, 40] . **Fig. 6** shows the time is taken to execute the ReEnc algorithm in our scheme and compared PRE schemes. It is seen that our scheme requires less time than other schemes. **Fig. 7** shows the time is taken to decrypt ciphertext in our scheme and PRE schemes and e-voting schemes. It is seen that in our scheme, to decrypt the ciphertext, it requires less time than decrypt the ciphertext in both other PRE schemes and e-voting schemes.

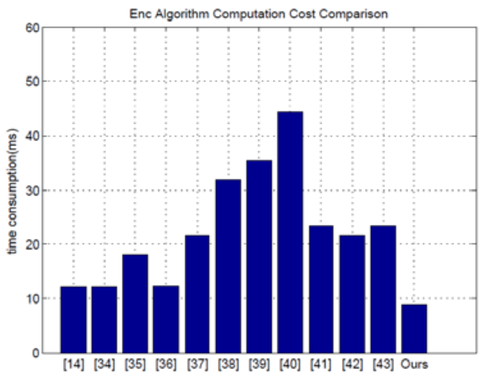

Fig. 5. Computation cost comparison of Enc

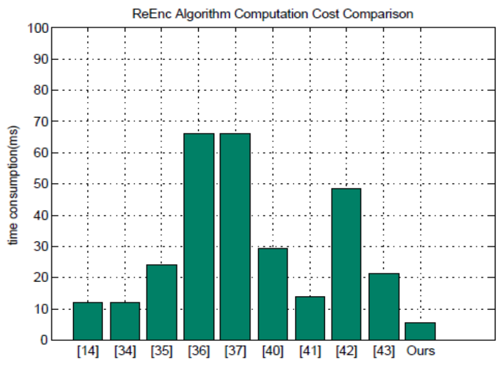

Fig. 6. Computation cost comparison of ReEnc

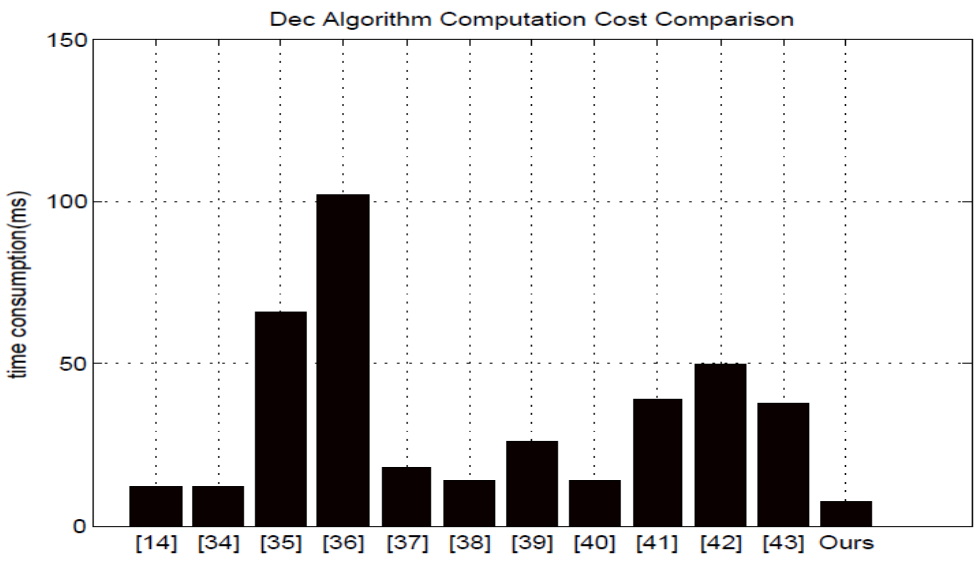

Fig. 7. Computation cost comparison of Dec

7. Conclusion

In this paper, we proposed a simple and efficient proxy re-encryption scheme and related proxy re-encryption protocol in e-voting schemes. The data security and user privacy of the voting system is ensured by this protocol. In our scheme, the protection of voter privacy is achieved by a re-encrypted ballot provided by a proxy and a randomization encryption service provided by the administrator. This method can be used in most e-voting schemes to provide receipt-freeness in a very efficient manner. The scheme is resistant to both CCA and collusion. Then, a detailed voting system architecture is provided in the paper. In the long-term research, we found that there are still meaningful points we continue to try in the future work.

- In the voting process, voters often want to vote in more convenient ways, such as mobile phones, laptops, and tablets. In the process of voting statistics and announcements, a central computer with supercomputing power is often required to count and Analyze voting. These different devices often generate different forms of ciphertext. In order to better adapt to different encryption devices, we will propose a heterogeneous proxy re-encryption scheme adapted to the voting system to help users complete voting securely and conveniently.

- Most of the proposed proxy re-encryption schemes are based on the number theoretic problem and cannot resist quantum attacks. As a new type of encryption technology, lattice-based encryption can resist quantum attacks and strengthen the security of the voting system. In the future, we will design and propose a lattice-based proxy re-encryption voting protocol to help the voting system resist quantum attacks.

Discussion: Vote buying and coercion issues are regarded as an important security requirement. These problems can be avoided to a great extent through the joint implementation of physical infrastructure, network protocol, and encryption technology. However, voting is based on social behavior. Thus, the above technical means can enhance data security and privacy, and prevent the voter provide a receipt, but cannot completely avoid human factors. If the voters cannot provide a receipt, these issues would be limited to social problems and are out of the scope of our discussion.

References

[1] W. Deng, H. Liu, J. Xu, H. Zhao, Y. Song, "An improved quantum-inspired differential evolution algorithm for deep belief network," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7319-7327, 2020. Article (CrossRef Link)

[2] H. Zhao, J. Zheng, W. Deng, Y. Song, "Semi-supervised broad learning system based on manifold regularization and broad network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 3, pp. 983-994, 2020. Article (CrossRef Link)

[3] R. Fotohi, S. F. Bari, M. Yusefi, "Securing Wireless Sensor Networks Against Denial-of-Sleep Attacks Using RSA Cryptography Algorithm and Interlock Protocol," *International Journal of Communication Systems*,Vol. 33, no. 4, pp. e4234, 2019. Article (CrossRef Link)

[4] S. Jamali, R. Fotohi, "DAWA: Defending against wormhole attack in MANETs by using fuzzy logic and artificial immune system," *the Journal of Supercomputing*, vol. 73, no. 12, pp. 5173-5196, 2017. Article (CrossRef Link)

[5] R. Fotohi, "Securing of Unmanned Aerial Systems (UAS) against security threats using human immune system," *Reliability Engineering & System Safety*, vol. 193, pp. 106675, 2020. Article (CrossRef Link)

[6] S. Jamali, R. Fotohi, "Defending against wormhole attack in MANET using an artificial immune system," *New Review of Information Networking*, vol. 21, no. 2, pp. 79-100, 2016. Article (CrossRef Link)

[7] R. Fotohi, E. Nazemi, F. S. Aliee, "An Agent-Based Self-Protective Method to Secure Communication between UAVs in Unmanned Aerial Vehicle Networks," *Vehicular Communications*, vol. 26, pp. 100267, 2020. Article (CrossRef Link)

[8] R. Fotohi, Y. Ebazadeh, M. S. Geshlag, "A new approach for improvement security against DoS attacks in vehicular ad-hoc network," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 7, pp. 10-16, 2016. Article (CrossRef Link)

[9] R. Aditya, B. Lee, C. Boyd, E. Dawson, "Implementation issues in secure e-voting schemes," in *Proc. of Abstracts and Papers (On CD-Rom) of the Fifth Asia-Pacific Industrial Engineering and Management Systems (APIEMS) Conference 2004 and the Seventh Asia-Pacific Division Meeting of the International Foundation of Production Research*. Queensland University of Technology, pp. 1-14, 2004. Article (CrossRef Link)

[10] H. Chen, R. Deviani, "A secure e-voting system based on rsa time-lock puzzle mechanism," in *Proc. of 2012 Seventh International Conference on Broadband, Wireless Computing, Communication and Applications*, pp. 596-601, 2012. Article (CrossRef Link)

[11] M. Blaze, G. Bleumer, M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 127–144, 1998. Article (CrossRef Link)

[12] J. Do, Y. Song, N. Park, "Attribute based proxy re-encryption for data confidentiality in cloud computing environments," in *Proc. of 2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, pp. 248–251, 2011. Article (CrossRef Link)
[13] T. Bhatia, A. Verma, G. Sharma, "Secure sharing of mobile personal healthcare records using certificateless proxy re-encryption in cloud," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 6, pp. e3309, 2018. Article (CrossRef Link)
[14] G. Ateniese, K. Fu, M. Green, S. Hohenberger, "Improved proxy re encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1-30, 2006. Article (CrossRef Link)
[15] B. Harris, D. Allen, "Black box voting: Ballot tampering in the 21st century," Renton, Washington: Talion, 2004. [Online]. Available: http://instinct.org/texts/black-box-voting/
[16] B. Adida, "Helios: Web-based open-audit voting," in *Proc. of USENIX security symposium*, vol. 17, pp. 335–348, 2008.
[17] B. Adida, O. Marneffe, O. Pereira, J.J. Quisquater, "Electing a university president using open-audit voting: Analysis of real-world use of helios," in *Proc. of USENIX Security Symposium*, vol. 9, no. 10, 2009.
[18] R. T. Mercuri, "Electronic vote tabulation checks and balances," 2001.
[19] J. C. Benaloh, D. Tuinstra, "Receipt-free secret-ballot elections," in *Proc. of the twenty-sixth annual ACM symposium on Theory of computing*. pp. 544–553, 1994. Article (CrossRef Link)
[20] M. Hirt, K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 539–556, 2000. Article (CrossRef Link)
[21] S. S. Chow, J. K. Liu, D. S. Wong, "Robust receipt-free election system with ballot secrecy and verifiability," in *Proc. of the Network and Distributed System Security Symposium*, vol. 8, pp. 81–94, 2008. Article (CrossRef Link)
[22] R. Wen, R. Buckland, "Masked ballot voting for receipt-free online elections," in *Proc. of International Conference on E-Voting and Identity*, pp. 18–36, 2009. Article (CrossRef Link)
[23] B. Yu, J. K. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, M. H. Au, "Platform-independent secure blockchain-based voting system," in *Proc. of International Conference on Information Security*, Springer, pp. 369–386, 2018. Article (CrossRef Link)
[24] Z. Xia, Z. Tong, M. Xiao, C.C. Chang, "Framework for practical and receipt-free remote voting," *IET Information Security*, vol. 12, no. 4, pp. 326–331, 2018. Article (CrossRef Link)
[25] K. Sako, J. Kilian, "Receipt-free mix-type voting scheme," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 393–403, 1995. Article (CrossRef Link)
[26] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *Proc. of International Workshop on Security Protocols*, pp. 25–35, 1997. Article (CrossRef Link)
[27] J. Furukawa, K. Sako, "An efficient scheme for proving a shuffle," in *Proc. of Annual International Cryptology Conference*, pp. 368–387, 2001. Article (CrossRef Link)
[28] B. Lee, K. Kim, "Receipt-free electronic voting scheme with a tamper resistant randomizer," in *Proc. of International Conference on Information Security and Cryptology*, pp. 389–406, 2002. Article (CrossRef Link)
[29] D. Boneh, P. Golle, "Almost entirely correct mixing with applications to voting," in *Proc. of the 9th ACM conference on Computer and communications security*, pp. 68–77, 2002. Article (CrossRef Link)
[30] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, S. Yoo, "Providing receipt-freeness in mixnet-based voting protocols," in *Proc. of International Conference on Information Security and Cryptology*, pp. 245–258, 2003. Article (CrossRef Link)
[31] R. Aditya, B. Lee, C. Boyd, E. Dawson, "An efficient mixnet-based voting scheme providing receipt-freeness," in *Proc. of International Conference on Trust, Privacy and Security in Digital Business*, pp. 152-161, 2004. Article (CrossRef Link)

[32] S. Tamura, H. A. Haddad, N. Islam, K. Md. R. Alam, "An Incoercible E-Voting Scheme Based on Revised Simplified Verifiable Re-encryption Mix-nets," *arXiv preprint arXiv:1512.05596*.
[33] J. Heather, D. Lundin, "The append-only web bulletin board," in *Proc. of International Workshop on Formal Aspects in Security and Trust*, pp. 242–256, 2008. Article (CrossRef Link)
[34] G. Ateniese, K. Fu, M. Green, S. Hohenberger, "Improved Proxy Re Encryption Schemes with Applications to Secure Distributed Storage," in *Proc. of the Network and Distributed System Security Symposium*, 2005. Article (CrossRef Link)
[35] B. Libert, D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Proc. of International Workshop on Public Key Cryptography*, pp. 360-379, 2008 Article (CrossRef Link)
[36] T. Isshiki, M. H. Nguyen, K. Tanaka, "Proxy re-encryption in a stronger security model extended from CT-RSA 2012," in *Proc. of Cryptographers Track at the RSA Conference*, pp. 277-292, 2013. Article (CrossRef Link)
[37] S. Zhang, H. Xiong, "$Laocoön$: Scalable and Portable Receipt-free E-voting Protocol without Untappable Channels," *arXiv preprint arXiv:1905.05562*, 2019. Article (CrossRef Link)
[38] C. Jin, G. Chen, J. Zhao, S. Gao, C. Yu, "Identity-based Deniable Authenticated Encryption for E-voting Systems," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 6, pp. 3299-3315, 2019. Article (CrossRef Link)
[39] E. Ahene, C. Jin, F. Li, "Certificateless deniably authenticated encryption and its application to e-voting system," *Telecommunication Systems*, vol. 70, no. 3, pp. 417-434, 2019. Article (CrossRef Link)
[40] Y. Zhan, B. Wang, Z. Wang, T. Pei, Y. Chen, Q. Qu, Z. Zhang, "Improved Proxy Re-Encryption With Delegatable Verifiability," *IEEE Systems Journal*, vol. 14, no. 1, pp. 592-602, 2020. Article (CrossRef Link)
[41] S. Maiti, S. Misra, "P2B: Privacy Preserving Identity-Based Broadcast Proxy Re-Encryption," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5610-5617, 2020. Article (CrossRef Link)
[42] M. Su, B. Zhou, A. Fu, Y. Yu, G. Zhang, "PRTA: A Proxy Re-encryption based Trusted Authorization scheme for nodes on CloudIoT," *Information Sciences*, vol. 527, pp. 533-547, 2020. Article (CrossRef Link)
[43] M. Su, L. Wang, "PreBAC: a novel Access Control scheme based Proxy Re-Encryption for cloud co mputing," *KSII Transactions on Internet & Information Systems*, vol. 13, no. 5, pp. 2754-2767, 2019. Article (CrossRef Link)

Wenchao Li received the M.S. degree from the University of Electronic Science and Technology of China (UESTC). She is currently pursuing the Ph.D. degree with the School of Cyber Science and Technology, Beihang University. Her research interests include public key cryptography and networks security.

Hu Xiong (Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2009. He is currently a Full Professor with UESTC. His research interests include public key cryptography and networks security.