A Secure and Private Clarke Tax Voting Protocol without Trusted Authorities

Changjie Wang and Ho-fung Leung
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Sha Tin, Hong Kong, P. R. China
{cjwang, lhf}@cse.cuhk.edu.hk

ABSTRACT
Electronic voting has become one of the most popular activities over the Internet. Security and privacy are always regarded as crucial factors in electronic voting system design. Various secure voting schemes have been proposed in the past several years to ensure the safe operation of electronic voting and most of them have focused on the common "*one man*, *one vote*" plurality voting. In this paper, we study on the security and privacy issues in the Clarke tax voting protocol, another important social choice protocol. This protocol is important in electronic voting, especially software agent based voting, because a voter's dominant strategy is truth-telling, and consequently the overhead for counterspeculation is minimized. For the very same reason, it is essential to achieve the security and the privacy protection of voters so that voters' preferences need not be made known to the public, should this protocol be practical and popular. In this paper, we first present several cryptographic building blocks, including *ElGamal cryptosystem*, *player-resolved distributed ElGamal decryption*, *proof of knowledge of* 1*-of-k plaintext* and *player-resolved mix network.* Then we propose a secure Clarke tax voting protocol making use of these techniques. In the proposed protocol, we achieve privacy protection, universal verifiability as well as other security requirements, such as secrecy, eligibility, completeness, *etc*. One important feature of the proposed protocol is that the full privacy protection of voters is guaranteed, which means that all information in voting are kept secret even in the presence of any collusion of participants involved in the voting. The only information known publicly is the final voting result, *i.e.,* the winning candidate and the tax for each voter.

Categories and Subject Descriptors
K.4.4 [**Computer and Society**]: Electronic Commerce

General Terms
Economics, Security, Theory

ICEC'04, Sixth International Conference on Electronic Commerce
Edited by: Marijn Janssen, Henk G. Sol, and René W. Wagenaar

Keywords
Electronic voting, Clarke tax voting protocol, Security, ElGamal encryption, Mix network, Privacy protection, universal verification.

1. INTRODUCTION
Secure electronic voting systems are examples of advance cryptographic protocols that are needed for practical applications in real life. In general, a (secure) electronic voting scheme can be viewed as a protocol that allows a group of voters to cast their votes, while one or more authorities collect the votes, compute and publish the outcome. It is clear that electronic voting schemes should satisfy all security requirements and achieve at least the same security level as those of ordinary paper-based elections, while the fact that the digital communication method is used may raise new security problems.

There have been many studies reported on the field of secure electronic voting systems, most of which, however, have focused only the common "*one man, one vote*" plurality protocol. The Clarke tax voting protocol is one in which the dominant strategy is truth-telling when the voters' preferences are quasilinear [1, 25] Consequently, it is often a more preferred protocol in some scenarios of electronic voting, especially intelligent agent-based voting, as it is comparatively more efficient when counterspeculation is no longer necessary. In this paper, we address the security issues in the Clarke tax voting protocol, and propose a secure and private Clarke tax voting protocol for secure electronic voting.

1.1 The Clarke Tax Protocol
We illustrate the Clarke tax algorithm with the following simple example. Amy, Betty and Cindy are good friends. One evening they decide to have Chinese food for dinner. Amy likes the spicy Sichuan cuisine very much. Betty, on the other hand, wishes to try the famous Peking duck, though Sichuan food is acceptable for her. Finally, Cindy really wants to have a delicious Cantonese seafood dinner, as she does not like spicy food. However, in a Chinese meal dishes are always shared, so they always need to share the bill after the meal, and each pays the same amount of money regardless of how much one really eats. Therefore, these friends must find a fair way to determine which cuisine they should go for.

A solution to this kind of problems was presented by Edward H. Clarke in early seventies [1, 2], which is now commonly known as the Clarke tax protocol. The basic idea of Clarke tax is that each voter is levied a tax that equals to other

voters' loss incurred by the preference he declares. In other words, a voter's tax is related to how much its vote has lowered the others' utilities, and voters that do not end up changing the outcome do not pay any tax. It is proved that the dominant strategy for a voter in a Clarke tax protocol is to declare his true preference [5].

Figure 1 shows an example of Clarke tax voting. Each row of the table shows a voter's preference. First, all voters Amy, Betty and Cindy declare their preferences for Sichuan (*S*), Peking (*P*) and Cantonese (*C*) food. According the declared preferences, they should go for the Cantonese food, as the sum of value is 21, which is the maximum among three possible options.

It can be observed that if Cindy had not cast her vote, the result would have been Sichuan food (S). Therefore, Cindy's participating in voting effectively brings other voter's total value from 18 to 6. Therefore, Cindy is levied a tax of 12. On the other hand, Amy and Betty do not need to pay any tax.

Voter ***i***	*Declared Value* v_i^j			*Sum without Voter i* $\sum_{k\neq i} v_k^j$			*Tax* tax_i
	S	*P*	*C*	*S*	*P*	*C*	
Amy	10	4	0	3	15	*21	**0**
Betty	8	10	6	5	9	*15	**0**
Cindy	–5	5	15	*18	14	6	**–12**
Sum $\sum_k v_k^j$	**13**	**19**	***21**				

Fig. 1. An Example of the Clarke Tax Voting:

$O_w = C$; $tax_{\text{Amy}} = 0$, $tax_{\text{Betty}} = 0$ and $tax_{\text{Cindy}} = -12$

Formally, the Clarke tax algorithm can be described as follows. Let Ω be the set of all possible outcomes and v_i^j denote the value of an outcome $O_j \in \Omega$ for voter *i*. The final outcome (social choice) can then be calculated as $O_w = \arg\max_{O_j \in \Omega} \sum_k v_k^j$. Let $O_{w_i'} = \arg\max_{O_j \in \Omega} \sum_{k\neq i} v_k^j$. The tax for voter *i* is then $tax_i = \sum_{k\neq i} v_k^w - \sum_{k\neq i} v_k^{w_i'}$.

1.2 Security Issues in the Clarke Tax Voting

Much research has been conducted on secure voting schemes and an extensive list of requirements for securing electronic voting is described in [3, 4]. In this paper, we will consider these security requirements in Clarke tax scheme, among which are privacy protection, universal verifiability, various forms of robustness as well as other requirements such as, secrecy, completeness, soundness, eligibility and so on.

It has been proved [5] that with Clarke tax scheme, revealing the true value of each outcome is the dominant strategy of a voter. Such information of true preference, however, is usually regarded as the personal privacy of the voters, which is often too sensitive to be made known to the public. Therefore, it is of high practical importance to achieve privacy protection in Clarke tax scheme should it be to become practical and popular. Besides the issues of secrecy, completeness, soundness, eligibility, *etc*., the two most important security requirements in Clarke tax system are *privacy protection* and *universally verifiability*.

Privacy protection means that all sensitive information (*i.e*., the voters' true values of each outcome, the sum for each outcome, and the sum for each outcome without voter *i*'s participation) in the Clarke tax scheme should be kept secret, except that the final results, *i.e.* the final elected outcome and the tax for each voter.

Universally verifiability is another important requirement of the scheme. Due to the privacy protection requirements, only the final results of the Clarke tax scheme should be known publicly. Consequently, it is necessary to achieve that any participant can verify the correctness of the results and detect any incorrect behavior of other voters with only the publicly available information.

1.3 Organization

The paper is organized as follows. We review the related work in section 2 and present the main contributions of our work. In section 3, we present several useful cryptographic building blocks used in this paper. A secure Clarke tax voting scheme with full privacy protection is proposed in section 4. We perform an analysis on security and efficiency of the proposed scheme in section 5, followed by conclusions in Section 6.

2. RELATED WORK

2.1 Secure Electronic Voting Protocols

There has been extensive research proposing various schemes on the design of secure electronic voting in the literatures. However, most of them concentrate on the common *"one man, one vote"* plurality voting protocol. One of the earliest approaches is proposed by Fujioka [4] using *blind signatures* [6] and *anonymous channels*. In this scheme, a voter prepares a ballot in plaintext, and then performs an interactive protocol with an authority that verifies the validity of the vote, *i.e.*, that the voter is eligible to vote and has not already cast his vote. During the interactive protocol, the authority issues a blind signature on the ballot, which means that the voter will obtain the authority's digital signature on the ballot, without revealing the ballot contents to the authority. Finally, all voters submit their signed ballots to another counting authority who will then check and count all valid ballots. Note that an anonymous channel (such channel can be implemented based on cryptography, using so-called *mix network* [7]) should be employed here to preserve the privacy of voters.

Another approach to secure voting scheme is using *verifiable secret sharing* [8, 9], in which there are several servers to count the votes, and voters interact with all servers to share verifiably secret votes among the servers [10, 11]. That is, each server gets a share of each voter's ballot, and these shares are constructed with respect to a threshold *t* such that *all* the servers together can cooperate to obtain complete information on each ballot, while any subset of at most *t* servers has no information at all. The voter must convince all servers that the shares are correctly constructed and thus he is prevented from voting multiple times or voting incorrectly. After all votes have been cast, all servers then jointly compute the result of the election without leaking any sensitive information.

Homomorphic encryption based secure voting schemes [4, 12, 13] are proposed and discussed most recently, in which a voter can simply publishes an particular encryption of vote using a specific public key cryptosystem with a homomorphism property, say, ElGamal encryption [14]. In such schemes, a public key known to everyone can be used for encryption of the vote. When submitting an encrypted vote, a voter not only has to identify himself to be eligible to vote, but also has to prove the correctness of the construction of his vote, *i.e.*, the encryption contains a valid vote. The privacy of voters is protected, since all individual votes are encrypted and the proof is zero-knowledge [13]. Moreover, due to the homomorphism of the encryption system, the election result can be computed efficiently. Homomorphism here means that two encryption of, say, number *a* and *b*, can be combined to produce a new encryption of $a+b$. By using this method repeatedly, all votes can be "implicitly added" without any decryption of any encrypted vote. Finally, we obtain an encryption of the voting result that can be obtained after decryption. The point of the above scheme is that the private key needed has to be *secretly shared* among a set of authorities in a threshold method. That is, each authority will hold a share of the private key, and the shares have to be constructed with a threshold value *t* so that no information of the private key leaks, unless more than *t* authorities corrupt or are broken into by an attacker. In other words, if at least $t+1$ authorities behave correctly, a decryption can be performed. This is also known as *threshold decryption*.

It is clear that most of the work reported in the literature on secure voting systems rely on threshold computation that is distributed among several authorities, out of which a fraction must be trustworthy. Assumptions on the trustworthiness of third parties or authorities are always made. In [10, 11], all servers share the votes of voters and it is assumed that at least $t+1$ servers will never collude to comprise the voters' privacy. In [4, 13], the private key for encrypting the vote is held by either a trusted third party or is shared among a set of authorities. However, in the reality, it is not always feasible to require that voters should trust any third party or other voters. It is also possible that groups of voters are formed, members of which share their knowledge and act as teams to perform maliciously. We therefore argue that distributing the trust on several authorities dose not achieve the privacy protection completely, since we can never rule out the possibility that some, or even all, of these authorities collude. Furthermore, in all previous work, the number of votes received by each candidate is also made publicly known at the end of the voting. Since it is possible that any subset of voters collude to act as a team, it is therefore easy for all but one voter to conclude and figure out the other voter's preference.

2.2 Our Contributions

As far as we know, there is no literature addressing the security in Clarke tax voting scheme. The Clarke tax voting mechanism is an important social decision protocol in which the dominant strategy is truth telling. This is of particular significance to software agent based electronic voting schemes, because efficiency of voting is improved as the overhead of counter-speculating can be eliminated.

In this paper, in addition to the security requirements suggested by others in the previous work, we focus on security issues in the Clarke tax system and propose a secure and private Clarke tax voting scheme that achieves *full privacy protection* and *universally verifiability*. We introduce the new standard for privacy protection, named *full privacy* protection. That is, in the proposed secure and fully private Clarke tax scheme, no (semi-)trusted authorities are needed, and the possibility that voters collude is not precluded. To achieve such full privacy protection, we employ the play-resolved *distributed ElGamal decryption* and *player-resolved mix network* so that to distribute the trust and computation to the all voters themselves. In such way, no information concerning the votes is revealed unless *all* voters share their knowledge. We note that it is impossible for *all* voters to form a single coalition and act as a team, or else voting is not necessary at all.

In conclusion, our main contribution is a universally verifiable secure Clarke tax voting scheme, in which each voter holds a share of secret information and jointly compute the final voting result. The only information publicly known is the final chosen outcome and the tax for each voter, while other related information is always protected.

3. PRELIMINARIES

In this section, we describe several cryptographic building blocks used in the scheme, including *ElGamal cryptosystem, play-resolved distributed ElGamal decryption*, *proof of knowledge of* 1*-of-k plaintext* and *player-resolved mix network*.

3.1 ElGamal Cryptosystem

One of the basic tool in our scheme is the ElGamal cryptosystem [14], which works over a group *GF(P)* where *P* is a big prime number.

Let *g* be a generator of *GF(P)*, which is a publicly available system parameter. A private key is an arbitrarily selected integer $x \in Z_P$. The corresponding public key is $y = g^x \bmod p$. Giving a plaintext $m \in Z_P$, its encryption under the public key is $\{\alpha, \beta\} = \{g^a \bmod p, y^a \cdot m \bmod p\}$ where $a \in Z_P$ is selected randomly. To decrypt this cipher text using the private key *x*, we just compute $\beta / \alpha^x = y^a m / (g^a)^x = m$.

The ElGamal cryptosystem has two important properties, namely the *semantic security* property and *homomorphism*. These two properties are useful in our scheme. The property of being *semantically secure* means that a player who only knows the public key can *re-encrypt* a ciphertext without knowing the private key. Suppose that $\{\alpha, \beta\} = \{g^{a_1} \bmod p, y^{a_1} \cdot m \bmod p\}$ is an encryption of some plaintext *m*, the player can then re-encrypt it by computing

$$\begin{aligned} &\{\alpha', \beta'\} \\ &= \{g^{a_1} \cdot g^{a_2} \bmod p, y^{a_1} \cdot y^{a_2} \cdot m \bmod p\}, \\ &= \{g^a \bmod p, y^a m \bmod p\} \end{aligned}$$

where $a = a_1 + a_2$. In other words, it is infeasible for another player to determine whether $\{\alpha', \beta'\}$ and $\{\alpha, \beta\}$ represent the encryption on the same plaintext. Another useful property of ElGamal system is *homomorphism*. Observe that if $\{\alpha_1, \beta_1\}$ and $\{\alpha_2, \beta_2\}$ represent ciphertext corresponding to plaintext m_1 and m_2 respectively, then $\{\alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2\}$ represents an encryption of the plaintext $m_0 \cdot m_1$.

3.2 Players-Resolved Distributed ElGamal Encryption

The ElGamal encryptions can be easily extended to a distributed version, which is useful in our proposed scheme, by using a *distributed key generation* protocol [15]. We call it the players-resolved distributed ElGamal encryption, since the private key is generated jointly by all players. First, each player independently chooses an arbitrary number x_i, then computes and publishes $y_i = g^{x_i} \bmod p$ along with a zero-knowledge proof of knowledge of y_i's discrete logarithm [16]. The public key of the system is then calculated as $Y = \prod_i y_i \bmod p$, and the corresponding private key $X = \sum_i x_i$. This process is called *distributed key generation*. To decrypt a ciphertext $\{\alpha, \beta\} = \{g^a \bmod p, Y^a m \bmod p\}$, all participants need to jointly perform the *distributed decryption*. That is, each participant computes and publishes $\alpha_i = \alpha^{x_i} \bmod p$ separately and proves its correctness by proving that $\log_g y_i = \log_\alpha \alpha_i$ [17]. The plaintext can then be decrypted by computing

$$\frac{\beta}{\prod_i \alpha_i} = \frac{Y^a m}{g^{a(\sum_i x_i)}} = \frac{(\prod_i g^{x_i})^a m}{g^{a(\sum_i x_i)}} = m \bmod p .$$

3.3 Proof of Knowledge of 1-of-*k* ElGamal Plaintext

By knowing the encryption exponent *a* of an ElGamal ciphertext $\{\alpha, \beta\} = \{g^a \bmod p, y^a \cdot m \bmod p\}$, a player can prove the knowledge of the plaintext *m* in zero knowledge by proving $\log_g \alpha = \log_y \beta / m$ [18]. In our scheme, we propose a cryptographic primitive, named proof of knowledge of 1-of-*k* ElGamal plaintext, as follows. To prove the knowledge of 1-of-*k* ElGamal plaintext, a player who submits an ElGamal ciphertext has to prove that the submission is a valid encryption, and the plaintext *m* is one of the *k* possible plaintexts, without revealing which plaintext it is exactly. We give the proof as follows.

Let *E(m)* be a submitted ElGamal ciphertext, that is, $E(m) = (\alpha, \beta) = (g^\alpha \bmod p, Y^\alpha m \bmod p),\ m \in \{G_0, G_1, ..., G_W\}$, where *g* and *Y* are the public key, α and *m* are kept secret. To prove the correctness of *E(m)*, the player generates a proof data *f* (assume that $m = G_k$):

$$f = \{(s, C, C_0, C_1, ..., C_{k-1}, C_k, C_{k+1}, ..., C_W), (u_0, u_1, ...u_{k-1}, u_k, u_{k+1}, ..., u_W), (\beta)\},$$

where,

(1) $s, C_0, C_1, ..., C_{k-1}, C_{k+1}, ..., C_W$ are randomly selected integers;

(2) $u_i = (Y^a \cdot m) / (g^a \cdot G_i) = A^a \cdot G_k / G_i$, for $i = 0, ..., W$, where $A = Y / g$. Note that $u_k = A^a$;

(3) $C = H(A^s \cdot u_0^{C_0} \cdot u_1^{C_1} ... \cdot u_{k-1}^{C_{k-1}} \cdot u_{k+1}^{C_{k+1}} \cdot ... \cdot u_W^{C_W})$;

(4) $C_k = C \oplus C_1 \oplus ... \oplus C_{k-1} \oplus C_{k+1} \oplus ... \oplus C_W$;

(5) $\beta = s - C_k \cdot a$.

To perform the verification on the correctness of *E*(*m*), anyone can check that the equation $C_1 \oplus ... \oplus C_W = H(A^\beta \cdot u_0^{C_1} \cdot ... \cdot u_W^{C_W})$ hold.

3.4 Players-Resolved Mix Network

Another key technique in our scheme is known as *mix network*, which is first introduced by Chaum [19] for privacy protection. Some studies have been done, in both implementation techniques, and applications for different purposes [20, 21, 22]. A *mix network* is multi-party protocol that involves several *mix servers*. The input to the *mix network* is a list of ciphertext items and the output is a new, permutated list of those ciphertext items such that there is one-to-one correspondence between the underlying plaintexts of input and output items. The security of a mix network is characterized by the infeasibility for an attacker to determine which output items correspond to which input items.

In this paper, we employ a mix network based on the ElGamal cipher, in which we do not assume the existence of mix servers; instead, we assign the permutation task to all participants. In such a way, the full privacy protection of correspondence between input and output is achieved, and this is what we call *players-resolved mix network*. In the following, we describe how this mix network works.

The input to the mix network consists of a list of ElGamal ciphertext $(\alpha_1, \beta_1), (\alpha_2, \beta_2), ..., (\alpha_k, \beta_k)$ encrypted by a public key. The corresponding private key is shared by all participants. Then, each participant performs a random permutation on this input list in turn by re-encrypting every ciphertext and output them randomly. The final output then be a sequence $(\alpha'_{\sigma(1)}, \beta'_{\sigma(1)}), (\alpha'_{\sigma(2)}, \beta'_{\sigma(2)}), ..., (\alpha'_{\sigma(k)}, \beta'_{\sigma(k)})$, where (α'_i, β'_i) represents a random re-encryption of (α_i, β_i), and $\sigma()$ is a random permutation function on *k* elements. The key point in above mix network is that each player must prove the correctness of his random permutation without revealing the correspondence between input and output. This proof can be done in zero-knowledge manner, as follows.

Suppose that each player possesses a *k*-input mix permutation consisting of a series of publicly verifiable 2-input mix boxes in some architecture. To achieve the public verifiability of the *k*-input mix permutation, the players only need to prove the correctness of the 2-input mix boxes. Assume that two input ElGamal ciphertexts: (α_1, β_1) and (α_2, β_2) on respective plaintexts m_1 and m_2, and the output are two randomized re-encrypted ciphertext (α'_1, β'_1) and (α'_2, β'_2) corresponding to respective plaintexts m'_1 and m'_2. To prove the correctness of the 2-input mix box, a player first proves that both (α'_1, β'_1) and (α'_2, β'_2) are correct re-encryption of either (α_1, β_1) or (α_2, β_2) (see **Appendix**), and then proves that $(\alpha_1\alpha_2, \beta_1\beta_2)$ and $(\alpha'_1\alpha'_2, \beta'\beta'_2)$ denote the encryption on the same plaintext $m_1 m_2$ using the *Plaintext Equivalence Proof* in [23].

4. SECURE CLARKE TAX SCHEME WITH FULL PRIVACY PROTECTION

4.1 An Overview of the Scheme

The proposed scheme can be considered to be a cryptographic model of secure multiparty computation, in which there are *n* voters, $V_1, V_2, ..., V_n$, and *m* possible outcomes ("candidates") $O_1, O_2, ..., O_m$. We also assume that there exists a public bulletin board for shared information. A voter has read-access to the whole board; while having write-access to only a designated area. Instead of having to reveal the true value of each candidate as in the original Clarke tax protocol, in our scheme, each voter will need to submit a particular ciphertext of his preferences using the ElGamal system. The public key for encryption is a public parameter and the corresponding private key is shared among all voters in a distributed manner. The key idea of our approach is to distribute the trust and the computation on all voters themselves. In this way, we do not need any (semi-)trusted third party as in other secure voting schemes, in which any form of collusion among these trusted parties has to be assumed impossible. Therefore, we do not make any special trust assumption in the proposed scheme. That is, no voters need to trust any third party, and it is anticipated that some group(s) of voters might collude in some form.

The voting process can be outlined as follows. Before the voting starts, all voters first perform the *distributed ElGamal key generation*, as described in the previous section, to generate a public key for system. Suppose that voter *i* chooses x_i and publishes $y_i = g^{x_i} \bmod p$. The public key of the system is then $Y = \prod_i y_i \bmod p$, and the corresponding private key $X = \sum_i x_i$. Note that the values of x_i, $1 \le i \le n$ are kept secret by each individual voter respectively. Therefore, the private key *X* cannot be retrieved unless with the cooperation of *all* voters, which is assumed to be impossible.

For ease of illustration, we suppose that there is a manager *M* to perform the collection of votes as well as some publicly verified calculations work.[1] Note that we need not assume any special trust on this manager *M*. When the voting starts, all voters cast the encrypted votes containing their preferences, and then jointly figure out the winning candidate and calculate the tax for each voter, while keeping all the sensitive information in secret. We assume that the voters' values should fall into a bounded range. For ease of presentation, throughout this paper we assume that $-50 \le v_i^j \le 50$, for $i \in [1,n], j \in [1,m]$.

4.2 Voting Procedures

We are now ready to give a formal description of the voting procedures.

First, each voter V_i generates a vector B_i that consists of *m* particular encrypted values corresponding to *m* candidates respectively, and publishes them on the bulletin board, *i.e.*

$$B_i: \quad \{E(2^{v_i^1}), E(2^{v_i^2}), ..., E(2^{v_i^j})\}, \text{ where } v_i^j \in [-50, 50].$$

For public verifiability, each voter has to prove the correctness of the construction of B_i by performing the *proof of knowledge of* 1*-of-K ElGamal plaintext* [2] on each encrypted value.

The manager *M* will then collect all submissions and calculate the component-wise product of all voters' vectors to get the sum vector $\bar{V}$ as:

$$\bar{V}: \quad \prod_{i=1}^{n} B_i = \{\prod_{i=1}^{n} E(2^{v_i^1}), \prod_{i=1}^{n} E(2^{v_i^2}), ..., \prod_{i=1}^{n} E(2^{v_i^m})\}.$$

This vector is called the *sum vector* because of the following reason. Observe that, due to the homomorphic property of ElGamal cryptosystem, the k^{th} component c_k of vector $\bar{V}$ has the form

$$c_k = \prod_{i=1}^{n} E(2^{v_i^k}) = E(2^{\bar{O}_k}), \text{ where } \bar{O}_k = \sum_{i=1}^{n} v_i^k.$$

Note that $\bar{O}_k = \sum_{i=1}^{n} v_i^k$ is the sum of values of candidate O_k. Therefore, the vector $\bar{V}$ is a ciphertext of the vector of value sum each candidate receives.

To figure out the final winning candidate, the manager *M* has to find the largest one among $\bar{O}_k = \sum_{i=1}^{n} v_i^k$, $k \in [1,m]$ with a public verified method, without revealing the exact value of $\bar{O}_k$.

In the following, we propose a method to test whether $\bar{O}_p \ge \bar{O}_q$ holds for two arbitrary encrypted sums $\bar{O}_p$ and $\bar{O}_q$ of values. Note that we know only the ciphertext $E(2^{\bar{O}_p})$ and $E(2^{\bar{O}_q})$ without having any further information such as $\bar{O}_p$ and $\bar{O}_q$ themselves. To compare $\bar{O}_p$ and $\bar{O}_q$, the manager *M* first calculates $C = E(2^{\bar{O}_p}) / E(2^{\bar{O}_q}) = E(2^{\bar{O}_p - \bar{O}_q})$. By examining whether $\bar{O}_p - \bar{O}_q \ge 0$, we can decide whether $\bar{O}_p$ is larger than $\bar{O}_q$ or not. Here, we can use the *mix and match* technique to examine whether $\bar{O}_p - \bar{O}_q \ge 0$ from $C = E(2^{\bar{O}_p - \bar{O}_q})$, without knowing the value of $\bar{O}_p - \bar{O}_q$. Recall that each value v_i^j satisfies $-50 \le v_i^j \le 50$, thus the $\bar{O}_p - \bar{O}_q$ should satisfy $-100 \cdot n \le \bar{O}_p - \bar{O}_q \le 100 \cdot n$, where *n* is the number of voters. Therefore, the problem of examining whether $\bar{O}_p - \bar{O}_q \ge 0$ is equal to examine whether $D(E(2^{\bar{O}_p - \bar{O}_q})) \in \{1, 2, 2^2, ..., 2^{100 \cdot n - 1}, 2^{100 \cdot n}\}$.[3]

---

[1] *M* might also be responsible for verifying the eligibility of the voters' identities with their public key certificates issued a trusted Certificate Authority.

[2] Here, the plaintext set should be $\{2^{-50}, 2^{-49}, ..., 2^0, .., 2^{50}\}$. That is, the voter has to prove that each encrypted value $E(m)$ is a correct ElGamal encryption such that $m \in \{2^{-50}, 2^{-49}, ..., 2^0, .., 2^{50}\}$, without revealing what *m* is exactly.

[3] Here, $D(x)$ means the ElGamal decryption on *x*.

The manager M now constructs and publishes $100\cdot n+1$ ciphertexts $C'_i = C/E(2^i)$, $0 \le i \le 100\cdot n$ in a publicly verifiable manner, that is, M generates $E(2^i)$ and proves the correctness by publishing a *proof of knowledge of ElGamal plaintext* using Schnorr identification protocol [42], and then calculates $C'_i = C/E(2^i)$ publicly.

All voters then work as a *mix network* as described in the previous section to perform a permutation with an input list $C'_i = (C'_0, C'_1, \ldots, C'_{100\cdot n})$, so as to mix them to form a new ciphertext list $\overline{C}' = (\overline{C}'_0, \overline{C}'_1, \ldots, \overline{C}'_{100\cdot n})$ which is a shuffled permutation of the re-encryption of the $C'_i = (C'_0, C'_1, \ldots, C'_{100\cdot n})$.

To determine whether $\overline{O}_p \ge \overline{O}_q$ for two arbitrary encrypted sums $\overline{O}_p$ and $\overline{O}_q$, all voters first perform the *distributed ElGamal decryption* on the list $C'_i = (C'_0, C'_1, \ldots, C'_{100\cdot n})$. Then all voters should check whether there exists one plaintext "1" in the decrypted list $D(\overline{C}'_i) = (D(\overline{C}'_0), D(\overline{C}'_1), \ldots, D(\overline{C}'_{100\cdot n}))$. If one plaintext "1" exists in the decrypted list, then it can be concluded that $\overline{O}_p \ge \overline{O}_q$. Otherwise, it can be concluded that $\overline{O}_p < \overline{O}_q$. The correctness of this approach can be established by the following theorem.

Theorem 1 *If there exists one plaintext "1" in the decryption list* $D(\overline{C}'_i) = (D(\overline{C}'_0), D(\overline{C}'_1), \ldots, D(\overline{C}'_{100\cdot n}))$ *, it is concluded that* $\overline{O}_p \ge \overline{O}_q$ *; if there is no plaintext "1" existing in the list* $D(\overline{C}'_i) = (D(\overline{C}'_0), D(\overline{C}'_1), \ldots, D(\overline{C}'_{100\cdot n}))$ *, it is concluded that* $\overline{O}_p < \overline{O}_q$.[4]

Proof Suppose the encrypted vector of each voter is:

$$B_i = \{E(2^{v_i^1}) = (g^{k_i^1}, Y^{k_i^1} 2^{v_i^1}), \ldots, E(2^{v_i^j}) = (g^{k_i^m}, Y^{k_i^m} 2^{v_i^m})\},$$

Where $v_i^j \in [-50, 50]$ and $k_i^j, 1 \le i \le n, 1 \le j \le m$ are the encryption exponent selected by voter V_i. The component-wise product of all voters' vectors can then be denoted as:

$$\overline{V} = \{c_1, \ldots, c_m\} = \{(g^{\sum_{i=1}^n k_i^1}, Y^{\sum_{i=1}^n k_i^1} 2^{\sum_{i=1}^n v_i^1}), \ldots, (g^{\sum_{i=1}^n k_i^m}, Y^{\sum_{i=1}^n k_i^m} 2^{\sum_{i=1}^n v_i^m})\}$$

As illustration before, to compare whether $\overline{O}_p \ge \overline{O}_q$ the manager M calculates $C = c_p / c_q = (g^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q}, Y^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q} 2^{\sum_{i=1}^n v_i^p - \sum_{i=1}^n v_i^q})$ and constructs $100\cdot n+1$ ciphertexts

$$C'_j = \frac{C}{E(2^j)} = \left(\frac{g^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q}}{g^{k_M^j}}, \frac{Y^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q} 2^{\sum_{i=1}^n v_i^p - \sum_{i=1}^n v_i^q}}{Y^{k_M^j} 2^j}\right)$$
$$= (g^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^j}, Y^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^j} 2^{\sum_{i=1}^n v_i^p - \sum_{i=1}^n v_i^q - j}),\ 0 \le j \le 100\cdot n$$

Here, $k_M^j, 0 \le j \le 100n$ is the arbitrary encryption exponent selected by the manager M. Note that $\overline{O}_p = \sum_{i=1}^n v_i^p$ and $\overline{O}_q = \sum_{i=1}^n v_i^q$. Without loss of generality, we suppose that $\overline{O}_p \ge \overline{O}_q$ and $\overline{O}_p - \overline{O}_q = \delta$, therefore,

$$C'_\delta = (g^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta}, Y^{\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta} \cdot 1).$$

The ciphertext list C': $(C'_0, C'_1, \ldots, C'_{100n})$ is then mixed by all voters and the output $\overline{C}'$: $(\overline{C}'_0, \overline{C}'_1, \ldots, \overline{C}'_{100n})$ is a shuffled *re-encryption* of C'. Note that there must be a factor in $\overline{C}'$, say $\overline{C}'_u$, corresponding to the C'_δ. Here, $u = \Phi(\delta)$ where $\Phi()$ is a shuffled permutation function. Then, the $\overline{C}'_u$ can be denoted as (r_δ is the re-encryption factor here):

$$\overline{C}'_u = (g^{(\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta)\cdot r_\delta}, Y^{(\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta)\cdot r_\delta} \cdot 1)$$

Now, all voters perform the distributed ElGamal decryption on the $\overline{C}'$: $(\overline{C}'_0, \overline{C}'_1, \ldots, \overline{C}'_{100n})$ and the decryption of the specific $\overline{C}'_u$ is as follows

$$D(\overline{C}'_u) = \frac{Y^{(\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta)\cdot r_w} 2^{\sum_{i=1}^n v_i^p - \sum_{i=1}^n v_i^q - \delta}}{(g^{(\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta)\cdot r_\delta})^{\sum_{i=1}^n x_i}}$$
$$= 2^{\sum_{i=1}^n v_i^p - \sum_{i=1}^n v_i^q - \delta}$$
$$= 2^{\overline{O}_p - \overline{O}_q - \delta}$$

Note that the values of $(g^{(\sum_{i=1}^n k_i^p - \sum_{i=1}^n k_i^q - k_M^\delta)\cdot r_\delta})^{x_i}, 1 \le i \le n$ are calculated by each voter separately with their secret x_i and then be published on the bulletin board with a proof of the correctness [19].

---

[4] There are some restrictions that should be noted.

(1) The value of n should not be too large, so that the calculation of $100\cdot n+1$ ciphertexts is feasible.

(2) It is clear that we have to select the module p of the ElGamal system carefully to assure that there are no common values in the set $A: \{2^{-100n}, 2^{-100n+1}, \ldots, 2^{-1}\}$ and set $B: \{1, 2, 2^2, \ldots, 2^{100n}\}$. We elaborate this requirement in more details. Suppose $\overline{O}_p < \overline{O}_q$ and $\overline{O}_p - \overline{O}_q = -\delta$, where $0 < \delta \le 100n$, then the decrypted list should be $\overline{D} = (2^{-\delta}/1, 2^{-\delta}/2, 2^{-\delta}/2^2, \ldots, 2^{-\delta}/2^{100n})$ in which there should no plaintext "1" exist. Note that $2^{-\delta}$ is the reverse of 2^δ in *GF(p)*, that is $2^\delta \cdot x = 1 \bmod p$, if let $2^{-\delta} = x$. Obviously, there is no plaintext "1" in the decrypted list if and only if $x \notin \{1, 2, 2^2, \ldots, 2^{100n}\}$. To satisfy this, we must select p such that $2^\delta \ne 1 \bmod p$ for $\delta = 1, 2, \ldots, 100n, 100n+1, \ldots, 200n$.

It is easy to deduce that if $\bar{O}_p \geq \bar{O}_q$ and $\bar{O}_p - \bar{O}_q = \delta$, there must be a decryption $D(\bar{C}'_u) = 1$, otherwise, $\bar{O}_p < \bar{O}_q$ if there is no plaintext "1" exist.

By repeating the above examination $m-1$ times, the manager *M* and the voters can find out the largest one among $\bar{O}_k = \sum_{i=1}^{n} v_i^k$, $k \in [1, m]$, the value sum received by the candidate that receives the largest sum of values, without knowing either the exact value of the sum of points of each candidate, or the difference between those sums. Finally, the manager *M* publishes the winner and all the verification information on the bulletin board, so that anyone can check the correctness of the result. It is obvious that any one can verify the legitimacy of the winner.

4.3 Calculation of individual Tax

The calculation of the tax for each voter is similar the above procedures, as described in the following.

To calculate the tax for voter V_i, the manager *M* first calculates the sum of values for each candidate without V_i's vote, by computing the component-wise product of all vectors product, except the V_i's vector, that is:

$$\bar{V}'_i = \{ \prod_{j=1, j\neq i}^{n} E(2^{v_j^1}), \prod_{j=1, j\neq i}^{n} E(2^{v_j^2}), ..., \prod_{j=1, j\neq i}^{n} E(2^{v_j^m}) \}$$

With the above method, the manager *M* as well as any voter can calculate the final winning candidate without V_i's vote, while keeping all individual values in secret. If the final voting result is just same as that with the participation of V_i, the tax for V_i is zero. Otherwise, the tax tax_i for V_i is calculated as follows.

Suppose the original winning candidate is O_w and the winning candidate without V_i's participant is $O_{w'_i}$, then the tax for V_i should be:

$$tax_i = \sum_{k\neq i} v_k^w - \sum_{k\neq i} v_k^{w'_i}$$

By calculating $Y' = E(2^{\sum_{k=1,k\neq i}^{n} v_k^w}) / E(2^{\sum_{k=1,k\neq i}^{n} v_k^{w'_i}})$, and requesting all voters to perform a further *distributed ElGamal decryption* on Y', we can get $2^{\sum_{k=1,k\neq i}^{n} v_k^w - \sum_{k=1,k\neq i}^{n} v_k^{w'_i}} = 2^{tax_i} \bmod p$. Though the calculation of tax_i is generally regarded as a difficult problem based on usually discrete logarithm problem assumption, in this particular setting of *n* and value range, we can easily calculate the tax_i by trying all possible values in [–100*n*, 100*n*].

5. ANALYSIS

5.1 Security Analysis

5.1.1 Privacy Protection and Public Verifiability

In the proposed secure Clarke tax scheme, only the final winning candidate[5] and the individual tax for each voter are published at the end of the voting, and other information, such as the voters' values of each candidate, the sum of the values of each candidate and difference between those sums are always kept secret. In such a way, we ensure the *privacy protection* of each voter in maximum degree.

In the protocol, each voter publishes his value for each candidate in a particular ElGamal encrypted vector form using a public key *Y*, instead of in plaintext. Without the knowledge of the corresponding private key *X*, nobody can get the values of voters without breaking the ElGamal cryptosystem, which is assumed to be infeasible. Making use of the homomorphic property of ElGamal encryption, we get the encrypted vector of the sum of the each candidate's values. By employing the *player-resolved mix and match* protocol, we perform the comparison among the encrypted sum value, without leaking any further information. Since the basic building blocks of encryption scheme, mix scheme and mix and match protocol are proved to be secure [14, 23, 24], it is easily to deduce that our scheme is secure against any passive attack.

Since the all related information is in ciphertext form, it is possible that any voter or the manager *M* perform maliciously to disturb the voting. For example, a voter may submit wrong encrypted values for some candidates or make an incorrect re-encryption during the mix work. In the proposed scheme, however, such active attacks can be easily resisted since we employ the zero-knowledge proof mechanism. For example, when publishing the encrypted values vector, a voter has to also publish a *proof of knowledge of* 1-*of-k ElGamal plaintext*, and during the mix network, voters have to prove the correctness of the output of each mix work by giving the *proof of the correctness of* 1-*of-k ElGamal re-encryption* and *Plaintext Equivalence Proof*. In such way, any participator can verify the correctness of each step of the scheme, and malicious voters will be detected immediately without additional communication and information revelation. Thus, the *public verifiability* (*robustness*) of the scheme is achieved.

5.1.2 Collusion-Proof

Another important security feature of our scheme is that we achieve the full privacy protection of the voters. That is, we do not assume the existence of any kind of trusted parties. Moreover, no collusion of voters can possibly lead to the revelation of any private information, unless all voters together form a single collusion, which is assumed impossible, otherwise voting is not necessary at all.

Theorem 2 *The proposed secure Clarke tax voting scheme is collusion-proof, which means that all private information of voters is always kept secret even in the presence of any form of collusion of voters and administrator, unless all voters are involved in the collusion.*

[5] That is, the only available result is which candidate is the winner, while the exact sum of the value of the winner is unknown.

Recall that in our scheme, each voter's submission is an encrypted vector of m ElGamal encryptions $B_i = \{E(2^{v_i^1}), E(2^{v_i^2}), ..., E(2^{v_i^j})\}$. The only method to get the true value of a voter is to decrypt the $E(2^{v_i^j})$ using the proper private key X. In our scheme, however, we do not assume one or more trusted (semi-trusted) authorities to maintain the private key. Instead, we distribute the trust over all voters, which means that all voters jointly generate the key pairs $(X = \sum_{i=1}^{n} x_i, Y = g^X \bmod p)$ where x_i is the secret number generated by each voter respectively (see section 3). As a result, the only way to decrypt the encrypted vector is that all voters participate in the computing with their secret numbers x_i. Therefore, any group of colluding voters cannot open any specific encrypted value vector, without the cooperation of all other voters.

While employing the mix network, we use a similar idea. That is, we do not suggest the existence of any mix server and the mix permutation is instead performed by all voters one by one. In such way, even (n – 1) voters collude, they still cannot decide the correspondence between the input ciphertexts and the output re-encrypted ciphertexts.

5.2 Efficiency Analysis

We discuss the communication efficiency and the computation efficiency of the proposed scheme in this section.

Table 1. Communication complexity of proposed secure Clarke tax scheme

	Pattern	Round	Volume
Voting (Per one voter)	Voter → Bullet in Board	1	$O(m)$
Determining the winner (Per one voter)	M → Bulletin Board	$m-1$	$O(100n+1)$
	Voter → Bullet in Board	$2(m-1)$	$O(100n+1)$ per round
Calculation of individual tax (Per one voter)	M → Bulletin Board	$m-1$	$O(n(100n+1))$ per round
	Voter → Bullet in Board	$2(m-1)+1$	$O(n(100n+1))$ in first $2(m-1)$ rounds, and at most $O(n)$ in the last round

Table 1 gives the communication complexity of the proposed scheme when there are n voters and m candidates, with the value distribution being [–50, 50]. It is shown that our protocol has low round communication complexity (only one round) in voting procedure, and approximate m – 1 rounds for determining the winning candidate and calculation of the individual tax. It is clear that most communication complexity is due to the fact that it is player-resolved, which is used for full privacy protection purpose.

Tables 2 shows the computational complexity of the proposed scheme. It is clear that the computational complexity of the scheme is approximately determined by m, n and the value distribution, so our schemes has heavy cost implication in case of a large number of voters and candidates or a wide range of value distribution. Therefore, a reasonable restriction of m and n and value distribution should be applied for an efficient implementation of our scheme.

Table 2. Computational complexity of proposed secure Clarke tax scheme

	Computational complexity	
Key pairs generation	One modulo exponential computation and one proof for each voter	
Voting vector generation	m encryptions and proofs for each voter	
Determining the winner	***M***	**Per one voter**
	$100n+1$ encryption and proofs; $m \cdot n$ ciphertext multiplications; $(m-1)\cdot(100n+2)$ ciphertext divisions.	$m-1$ mix permutations ($(100n+1)$-input); $m-1$ modulo exponential computations and proofs.
Calculation of individual tax	***M***	**Per one voter**
	$m+(m-1)\cdot(100n+2)$ ciphertext divisions.	$n\cdot(m-1)$ mix permutations ($(100n+1)$-input); at most $n\cdot(m-1)$ modulo exponential computations and proofs.

6. CONCLUSION

In this paper, we address the security issues in Clarke tax protocol and propose a secure electronic Clarke tax voting scheme that achieves full privacy protection of the voters and universal verifiability, as well as other suggested security properties. We propose the conception of *full privacy protection*, *i.e.* all information in Clarke tax voting system are concealed during and after the voting, except the final result of who is the winning candidate and the tax for each voter. No one can compromise the voters' privacy without the cooperation of *all* voters. This requirement is stronger and more realistic than those achieved by cryptographic electronic voting systems found in the literature. Our scheme also satisfies the universal verifiability criterion, which means that any participant can publicly verify the correctness of both the each step in the voting procedure and the final voting results.

7. REFERENCES

[1] Clarke, E. H. "Multipart pricing of public goods", *Public Choice* 11, pp17-33, 1971.

[2] Clarke, E. H. "Multipart Pricing of public goods: An example", *Public Price for Public Products*, Urban Inst., Washington, 1972.

[3] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale election", In advance in Cryptology – Auscrypt'92, LNCS, Springer-Verlag, pp.244-251, 1992.

[4] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election schemes", In Advance in cryptology – Eurocrypt'97, Vol 1233, LNCS, Springer-Verlag, pp. 103-118, 1997.

[5] Ephrati, Eithan and Rosenschein, Jeffery S. "Voting and multi-agent consensus", Technical report, Computer Science Department, Hebrew University, Jerusalem, Israel,1991.

[6] Chaum,D., "Blind signatures for untraceable payments", *Advances in Cryptology-CRYPTO'82 Proceedings*, pp199-203, Plenum Press, 1983.

[7] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers", In *Advances in Cryptology-Eurocrypt'98*, Vol. 1403, LNCS, Springer-Verlag, pp.437-447, 1998.

[8] M. Stadler, "Publicly verifiable secret sharing", In *Advances in Cryptology-Eurocrypt'96*, Vol. 1070, LNCS, Springer-Verlag, pp. 190-199, 1996.

[9] E.Fujisaki and T. Okamoto, "A practical and provably secure scheme for publicly verifiable secret sharing and its application", In *Advances in Cryptology-Eurocrypt'98*, Vol 1403, LNCS, Springer-Verlag, pp.32-46, 1998.

[10] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting", In advance in Cryptology – Crypto'99, Vol. 1666, LNCS, Springer-Verlag, pp 148-164, 1999.

[11] R. Crammer, M. Franklin, B. Schoenmakers and M. Yung, "Multi-authority secret ballot elections with linear work", In advance in cryptology-Eurocrypt'96, Vol.1070, LNCS, Springer-Verlag, pp.72-83, 1996.

[12] J. Benaloh, *Verifiable secret-ballot elections*, PhD thesis, Yale University, Department of Computer Science, New Haven, CT, September 1987.

[13] K. Sako and J. Killian, "Secure voting using partial compatible homomorphism", In *Advances in Cryptology-Crypto'94*, Vol. 839, LNCS, Springer-Verlag, pp.411-424, 1994.

[14] ElGamal, T., A Public-key cryptosystem and a signature scheme based on discrete logarithms, *In Advances in Cryptology - CRYPTO'84 Proceedings*, Springer-Verlag, p10-18, 1985.

[15] Wang, C.J. and Leung, H. F., "Secure Double Auction Protocols with Full Privacy Protection". In: *Proceedings of the 6th Annual International Conference on Information Security and Cryptography*, LNCS, Springer-Verlag, 2003.

[16] Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology*, 4, pp 161-174, 1991.

[17] Chaum, D., Pedersen, T.P.:Wallet databases with observers. In: *Advances in Cryptology – Proceedings of the 12th Annual International Cryptology Conference*, LNCS 740, Springer Verlag, 1992.

[18] R. Cramer, I. Damgard and B. Schoenmakers, "Proofs of partical knowledge and simplified design of witness hiding protocol", Proceedings of CRYPT'94, pp.174-187, 1994.

[19] Chaum, D. "Untraceable electronic mail, return addresses, and digital pseudonyms", *Communications of the ACM*, 24(2), pp.84-88, 1981.

[20] Horster, P., Michels, M. and Petersen, H. "Some remarks on a receipt free and universially verifiable mix-type voting scheme", ASIACRYPT'96, Vol. 1163, LNCS, pp.125-132, Springer-Verlag, 1996.

[21] Park, C., Itoh, K. and Kurosawa, K. "All/nothing election scheme and anonymous channel", EUROCRYPT'93, Vol. 921, LNCS, Springer-Verlag, 1993.

[22] Sako, K, and Kilian, J. "Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth.", EUROCRYPT'95, Vol. 921, LNCS, Spinger-Verlag, 1993.

[23] Jakobsson, M. and Juels, A. "Millimix: Mixing in small batches", DIMACS Technical Report 99-33, June, 1999.

[24] Jakobsson, M and Juels, A., "Mix and Match: Secure Function Evaluation via ciphertexts", *Proceedings of ASIACRYPT 2000*, LNCS, Springer-Verlag, pp.162–177, 2000.

[25] Groves, T., "Incentives in Teams", *Econometrica*, 41:617–631, 1973.

Appendix : Proof of the Correctness of 1-of-*k* ElGamal Re-encryption

To prove the correctness of the 2-input mix box, a player first has to prove that the each output ciphertext is a correct re-encryption of one of the input ElGamal encryptions. In the following, we give the general zero-knowledge proof of such problem.

Suppose there are n input ElGamal ciphertexts $E(m_1), E(m_2), ..., E(m_n)$ where $E(m_i) = (g^{k_i}, Y^{k_i} m_i)$. Here, g and Y are public and $k_1, k_2, ...k_n, m_1, m_2, ...m_n$ are kept secret. Without loss of generality, we assume that the output is the re-encryption of w-th input ciphertext, that is, $i = w, k'_w = k_w + s_w$, where s_w is encryption exponent generated and keep privately by the player. Now, the player proves that $E(m_w)$ is a re-encryption of one of the $E(m_1), E(m_2), ..., E(m_n)$, without reveal what w is. Firstly, the player calculates $r_i = g^{k'_w} / g^{k_i}, R_i = Y^{k'_w} m_w / Y^{k_i} m_i, H_i = R_i / r_i$ for $1 \le i \le n$. Let $h = Y / g$, and then $H_w = Y^{s_w} / g^{s_w} = h^{s_w}$. The player then proves the knowledge of s_w corresponding one of the H_i by showing the proof data

$$f = \{S, C, C_1, ..., C_{w-1}, C_w, C_{w+1}, ..., C_n\},$$

where,

(1) α, $C_1, ..., C_{w-1}, C_{w+1}, ..., C_n$ are randomly selected integers from Z_q;

(2) $C = H(L \| z)$, where $L = H_1 \| H_2 \|,...,\| H_n$, and $z = h^{\alpha} \prod_{i=0, i \neq w}^{n} H_i^{C_i}$

(3) $C_w = C - (C_1 + C_2 + ... + C_{w-1} + C_{w+1} + C_n) \bmod q$

(4) $S = \alpha - C_k \cdot s_w \bmod q$

Verification. Anyone can verify the correctness of $E(m_i) = (g^{k_i'}, Y^{k_i'} m_i)$, by checking whether the equation: $C_1 + ... + C_K = H(L \| h^S H_1^{C_1} H_2^{C_2}, ... H_n^{C_n})$ holds.