1998 International Computer Symposium
Workshop on Cryptology and Information Security
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A Receipt-Free Voting Scheme Using Partially Compatible Homomorphisms

Chih-Hung Wang and Tzonelih Hwang

Institute of Information Engineering
National Cheng-Kung University
Tainan, Taiwan, R.O.C.
Email: hwangtl@server2.iie.ncku.edu.tw

ABSTRACT

A receipt-free voting scheme is proposed based on partially compatible homomorphic encryptions. The new scheme differs from the previous works in two features: receipt-freeness and the limitation for the maximum number of 'yes' votes a voter can cast. Although there are solutions for "receipt-freeness" of a voting scheme, they are either impractical or inefficient. Solution based on partially compatible homomorphisms is not existed so far.

1. INTRODUCTION

Many promising electronic voting schemes [5, 7, 9, 10, 12-14, 19, 21] have been proposed to satisfy several important security properties including: completeness, soundness, privacy, unreusability, fairness and verifiability as discussed in [12]. However, these properties are not good enough to realize the security requirements of a physical voting booth mechanism used nowadays. Most existing electronic voting schemes give each voter a receipt for him to check whether his vote is counted or not. By this receipt, however, each voter can prove his voting choice and thus it provides a chance for a malicious user (a briber) to buy the votes. To solve this problem, one has to use a receipt-free voting scheme.

Benaloh and Tuinstra [1] proposed the first work on receipt-freeness but used a special hardware assumption. Independently, Niemi and Renvall [16] tried to solve the bribery problem by using a physical voting booth where a voter performs multiparty computation with all centers. Both approaches need impractical assumptions on the use of physically secure equipments.

A practical solution for bribery problem had been proposed by Sako and Kilian at Eurocrypt'95 [20], which assumes the existence of untappable private channel. Their scheme is based on the mixer/anonymous channel technology [5, 12 17], which has two critical problems: the batch process and the individual verifiability [19]. Due to the batch process, if one vote is omitted, then the election will be blocked and each voter has to revote. Though individual verifiability allows each voter to check whether his own votes are counted correctly by the center or not, it virtually restricts one to check that for the others' votes. To solve this problem, Sako and Killian [20] proposed a protocol of universal verifiability to allow a voter to audit all voters' votes, but it performs sequentially through all mixers, which is very time-consuming.

Another approach to construct a secure voting scheme uses the number theoretic techniques [2, 10, 13, 19] (e.g., partially compatible homomorphisms). This approach does not require the batch process. Each voter can easily verify whether the election is performed properly or not. However, this approach suffers from both the communicational cost and computational complexity. Sako and Kilian [19] have proposed an improvement with moderate communication cost and low round complexity. Their protocol is also implemented on current generation PC's and obtains satisfactory results.

Unfortunately, Sako-Kilian's scheme [19] failed to provide the receipt-free property, which is essential to solve the bribery problem. Each voter in their scheme has to publish a few encryptions of subvotes for verification. These encryptions provide the voter a way to prove his voting choice to bribers. The purpose of this paper, therefore, is to construct a receipt-free voting scheme based on partially compatible homomorphisms. The newly proposed scheme can also be extended to provide a way to verify whether the maximum number of 'yes' votes that a voter can cast is less than or equal to a threshold value Γ.

This paper is structured as follows. Section 2 reviews the previous techniques. Section 3 proposes

several new basic protocols and a new voting scheme with receipt-free property. The extended scheme which limits the maximum number of 'yes' votes is also presented here. Section 4 gives the security analysis and discussions. We conclude this paper in Section 5.

2. TECHNIQUES USED

Techniques used to construct the new voting scheme are discussed here. While Section 2.1 reviews the existing techniques for partially compatible homomorphisms, Section 2.2 proposes two new ones.

2.1 The Previous Techniques

Two techniques are reviewed in this section: the partially compatible homomorphic encryptions, and two zero-knowledge proofs for partially compatible homomorphic encryptions, which provide methods for the voter to securely split his votes over two or many centers.

Partially Compatible Homomorphic Encryptions

Let x, y be in Z_q, where q is a large integer, and $x+y$ be in S, where S has only a few elements (e.g., $S = \{1, -1\}$). x, y, and $x+y$ are secret values. Let $E()$ be a probabilistic encryption. If $E()$ has additive homomorphic property (i.e., $E(x+y) = E(x)E(y)$), then given $E(x)$ and $E(y)$ one can reveal the value of $x+y$ (i.e., $s_i = x+y$, for some $s_i \in S$ $E(s_i) = E(x)E(y)$). Similarly, if $E()$ has multiplicative homomorphic property (i.e., $E(xy) = E(x)E(y)$), then given $E(x)$ and $E(y)$, one can reveal the value of xy.

A partially compatible additive homomorphic encryptions family is a family of probabilistic encryptions $\{E_1, \cdots E_n\}$ where each E_i satisfies additive homomorphism ($E_i(x+y) = E_i(x)E_i(y)$), however, for different encryptions E_i and E_j, there is no feasible way to find an E_k, for some k, such that $E_k(x+y) = E_i(x)E_j(y)$ (see [2, 10, 19]). Thus, given $E_i(x)$ and $E_j(y)$, it reveals nothing about $x+y$ even if $x+y$ is in a small set S. Consequently, let $x_1, x_2, \cdots, x_n$ be chosen uniformly from Z_q to sum up to an s. If $(n-2)$ of values $\{x_1, x_2, \cdots, x_n\}$ are known (for simplicity, assume they are $x_1, x_2, \cdots, x_{n-2}$), then their sum $s' = \sum_{i=1}^{n-2} x_i$ can be calculated. In this case, given $\{E_i(x_i)\}_{i=1,2\cdots,n}$, it also reveals nothing about $s = x_1 + x_2 + \cdots + x_n$ because one cannot obtain $x_{n-1} + x_n (= (s - s'))$ from $E_{n-1}(x_{n-1})$ and $E_n(x_n)$. This result will be used in Section 3.

Zero-knowledge Proof for Partially Compatible Homomorphic Encryptions

Two efficient interactive zero-knowledge proofs, *prove*± 1 and *prove-sum*, have been proposed by Sako and Kilian [19]. Before describing them, we give a simple example of the voting scheme to explain their usage.

Assume there exist two centers C_1 and C_2 in the voting scheme. The centers randomly select two public encryptions E_1 and E_2 such that the set of $\{E_1, E_2\}$ belongs to a family of partially compatible additive homomorphic encryptions and no one has any idea about the decryptions of these encryptions. Each voter V_t, $t \in [1, \sigma]$, splits his vote into two subvotes $x_1^{(t)}$ and $x_2^{(t)}$. Then he publishes $E_1(x_1^{(t)})$ and $E_2(x_2^{(t)})$ and uses *prove*± 1 to prove $x_1^{(t)} + x_2^{(t)} \in \{1, -1\}$. The sum of $x_1^{(t)}$ and $x_2^{(t)}$ represents the vote of 'yes' $(+1)$ or no (-1). Note that the voter hides the values of $x_1^{(t)}$ and $x_2^{(t)}$.

Upon voting, the voter V_t privately sends $x_i^{(t)}$ to C_i, $1 \le i \le 2$, and each C_i checks whether $x_i^{(t)}$ is consistent with $E_i(x_i^{(t)})$. Then, in the votes counting phase, C_i sums up $x_i^{(t)}$ for all voters V_t, $t \in [1, \sigma]$, and posts subtally $T_i = \sum_{t=1}^{\sigma} x_i^{(t)}$. Therefore, each voter can verify $E_i(T_i) = \prod_{t=1}^{\sigma} E_i(x_i^{(t)})$ and compute $T = T_1 + T_2$ which is the result of the voting.

Further, if there are n centers $C_1, \cdots, C_n$ in the voting scheme, the voter will split his vote into n subvotes $x_1^{(t)}, \cdots, x_n^{(t)}$. Then V_t publishes all $E_i(x_i^{(t)})$, $1 \le i \le n$, and proves $\sum_{i=1}^{n} x_i \in \{1, -1\}$. This proof will be broken into two stages. First, the voter V_t randomly generates $A^{(t)}$ and $B^{(t)}$ such that $A^{(t)} + B^{(t)} \in \{1, -1\}$ and uses *prove-sum* to prove $\sum_{i=1}^{n} x_i^{(t)} = A^{(t)} + B^{(t)}$. Then he uses *prove* ± 1 to prove $A^{(t)} + B^{(t)} \in \{1, -1\}$.

We now describe these two proofs in the following. (For simplicity, we drop the superscript t). Note that the value q is a large prime, $q-1$ should contain a large prime factor, and ω is a parameter of security level.

(1) *Proving* $x_1 + x_2 \in \{1, -1\}$ *without revealing* x_1 *and* x_2

PROTOCOL-*prove* $\pm 1(x_1, x_2)$: Given $E_1(x_1)$ and $E_2(x_2)$, where the set $\{E_1, E_2\}$ belongs to a family of partially compatible additive homomorphic encryptions and no one has any idea about the decryptions of these encryptions, prove that $x_1 + x_2 \in \{1, -1\}$ $(mod\ q)$.

For $\lambda = 1, 2, \cdots \omega$

1. The prover randomly chooses $r^{(\lambda)} \in Z_q$ and $s^{(\lambda)} \in \{1, -1\}$ and computes a bit commitment $R^{(\lambda)}$ for $r^{(\lambda)}$. Then he posts $(Y_1^{(\lambda)}, Y_2^{(\lambda)}, R^{(\lambda)})$ where $Y_1^{(\lambda)} = E_1(s^{(\lambda)}(x_1 + r^{(\lambda)})) = (E_1(x_1)E_1(r^{(\lambda)}))^{s^{(\lambda)}}$, and $Y_2^{(\lambda)} = E_2(s^{(\lambda)}(x_2 - r^{(\lambda)})) = (E_2(x_2)E_2(r^{(\lambda)})^{-1})^{s^{(\lambda)}}$.
2. The verifier has the following two choices:

2.1 Asks the prover to reveal $r^{(\lambda)}$ and $s^{(\lambda)}$ and then checks that $r^{(\lambda)}$ is consistent with $R^{(\lambda)}$, and $Y_1^{(\lambda)}$ and $Y_2^{(\lambda)}$ are constructed as above correctly.

2.2 Asks the prover to reveal $s^{(\lambda)}(x_1 + r^{(\lambda)})$ and $s^{(\lambda)}(x_2 - r^{(\lambda)})$ and then checks whether the following equations hold.

$$Y_1^{(\lambda)} = E_1(s^{(\lambda)}(x_1 + r^{(\lambda)})),$$

$$Y_2^{(\lambda)} = E_2(s^{(\lambda)}(x_2 - r^{(\lambda)})), \text{ and}$$

$$s^{(\lambda)}(x_1 + r^{(\lambda)}) + s^{(\lambda)}(x_2 - r^{(\lambda)}) \in \{1, -1\}.$$

End For

(2) *Proving* $x_1 + x_2 + \cdots x_n = A + B$ *without revealing* $x_1, \cdots x_n$

PROTOCOL-*prove* $-$ *sum*$(x_1, \cdots, x_n, A, B)$: Given $E_1(x_1), \cdots, E_n(x_n), E_a(A), E_b(B)$, where the set of $\{E_a, E_b\ E_1 \cdots E_n\}$ belongs to a family of partially compatible additive homomorphic encryptions and no one has any idea about the decryptions of these encryptions, prove that $x_1 + \cdots + x_n = A + B\ (mod\ q)$.

For $\lambda = 1, 2, \cdots \omega$

1. The prover randomly chooses $r_i^{(\lambda)} \in Z_q$, for $0 \le i \le n$, and computes a bit commitment $R_i^{(\lambda)}$ for each $r_i^{(\lambda)}$. Then he posts $(Y_1^{(\lambda)}, \cdots, Y_n^{(\lambda)}, Y_a, Y_b, R_i^{(\lambda)})$, where $Y_i^{(\lambda)} = E_i(x_i + r_i^{(\lambda)}) = E_i(x_i)E_i(r_i^{(\lambda)}), 1 \le i \le n$, $Y_a = E_a(A + r_0^{(\lambda)}) = E_a(A)E_a(r_0^{(\lambda)})$, and $Y_b = E_b(B + (\sum_{i=1}^n r_i^{(\lambda)}) - r_0^{(\lambda)}) = E_b(B)\Big(\prod_{i=1}^n E_b(r_i^{(\lambda)})\Big)E_b(r_0^{(\lambda)})^{-1}$.

2. The verifier has the following two choices:

   2.1 Asks the prover to reveal $r_i^{(\lambda)}$ $(0 \le i \le n)$ and then checks that $r_i^{(\lambda)}$ is consistent with $R_i^{(\lambda)}$, and $Y_1^{(\lambda)}, \cdots, Y_n^{(\lambda)}$, Y_a and Y_b are constructed as above correctly.

   2.2 Asks the prover to reveal $\{(x_i + r_i^{(\lambda)})\}_{1 \le i \le n}$, $(A + r_0^{(\lambda)})$ and $(B + (\sum_{i=1}^n r_i^{(\lambda)}) - r_0^{(\lambda)})$ and then checks whether the following equations hold.

$$Y_i^{(\lambda)} = E_i(x_i + r_i^{(\lambda)}), 1 \le i \le n,$$

$$Y_a = E_a(A + r_0^{(\lambda)}),$$

$Y_b = E_b(B + (\sum_{i=1}^n r_i^{(\lambda)}) - r_0^{(\lambda)})$, and $\sum_{i=1}^n (x_i + r_i^{(\lambda)}) = (A + r_0^{(\lambda)}) + (B + (\sum_{i=1}^n r_i^{(\lambda)}) - r_0^{(\lambda)})$.

End For

2.2 New Techniques

Receipt-free Technique

Considering the simple example in the Section 2.1, $E_i(x_i^{(t)})$'s published by the voter can be treated as the "receipts" of the voting. A malicious user (a briber) can ask the voter to release his $x_i^{(t)}$'s after voting. Then the briber checks whether each $x_i^{(t)}$ is consistent with $E_i(x_i^{(t)})$. If the verification is successful and $\sum_{i=1}^n x_i^{(t)} = 1$, the briber believes the voter had cast a 'yes' vote. In this case, the voter is entrapped to sell his vote since his choice can easily be proved.

Therefore, the main idea of our new scheme is to prevent that a voter can prove his choice by using these encryptions of subvotes. Instead, the voter V_t has to post two sets of encryptions $E_i(x_{i1}^{(t)})$ and $E_i(x_{i2}^{(t)})$, $1 \le i \le n$, where $(\sum_i x_{i1}^{(t)}, \sum_i x_{i2}^{(t)}) = (1, -1)$ or $(-1, 1)$. V_t decides his vote in the voting phase by privately sending either $x_{i1}^{(t)}$ or $x_{i2}^{(t)}$ to the centers. Applying this method, even if the voter reveals all $x_{i1}^{(t)}$'s and $x_{i2}^{(t)}$'s to the briber, he also cannot prove what he had voted with these two sets of encryptions. However, this method may cause some problems when the centers count their subtallies. We will explain it later.

Prove-subtally Protocol

Although the voter V_t posts two sets of encryptions, he has to select one of them according to his choice in the voting phase. However, one cannot check the center C_i's subtally using the equation $E_i(T_i) = \prod_{t=1}^{\sigma} E_i(x_i^{(t)})$ because he knows nothing about the set of encryption the V_t really chose (i.e., they do not know which of $E_i(x_{i1}^{(t)})$ and $E_i(x_{i2}^{(t)})$ the $E_i(x_i^{(t)})$ equals to) . Basically, the center knows this secret information and has to keep it, otherwise the receipt-freeness cannot be achieved. Therefore, we provide a *prove-subtally* protocol that allows a center to prove that his subtally is counted correctly without revealing the voter's choice between these two sets of encryptions. This protocol is a "shuffling protocol" which hides the encryptions $E_i(x_{i1}^{(t)})$ and $E_i(x_{i2}^{(t)})$ by using a secret/public key pair of the center.

3. A NEW VOTING SCHEME WITH RECEIPT-FREE PROPERTY

3.1 Basic Protocols

Besides the *prove* ± 1 and *prove-sum* protocols described in Section 2.1, three new protocols *prove-Q*,

multi-commitment, and *prove-subtally* are required in our scheme. They are discussed individually in the following.

(1) PROTOCOL-*prove-Q*$(x_1, x_2, \cdots, x_n)$:

Given $E_1(x_1), \cdots, E_n(x_n)$, and Q, where the set, $\{E_1 \cdots E_n\}$, belongs to a family with partially compatible additive homomorphic encryptions and no one has any idea about the decryptions of these encryptions, prove that $x_1 + \cdots + x_n = Q \pmod q$ without revealing x_i's, $1 \le i \le n$.

For $\lambda = 1, 2, \cdots \omega$

1. The prover randomly chooses $r_i^{(\lambda)} \in Z_q$, for $1 \le i \le n$, such that $\sum_i r_i^{(\lambda)} = Q$, and computes a bit commitment $R_i^{(\lambda)}$ for each $r_i^{(\lambda)}$. Then he posts $Y_i^{(\lambda)} = E_i(x_i - r_i^{(\lambda)}) = E_i(x_i)(E_i(r_i^{(\lambda)}))^{-1}, 1 \le i \le n$.
2. The verifier has the following two choices:
   - 2.1 Asks the prover to reveal $r_i^{(\lambda)}$ $(1 \le i \le n)$ and then checks that $r_i^{(\lambda)}$ is consistent with $R_i^{(\lambda)}$; verifies that $\sum_i r_i^{(\lambda)} = Q$, and $Y_1^{(\lambda)}, \cdots, Y_n^{(\lambda)}$ are constructed as above correctly.
   - 2.2 Asks the prover to reveal $\{(x_i - r_i^{(\lambda)})\}_{1 \le i \le n}$, and then checks whether the following equations hold.

$$Y_i^{(\lambda)} = E_i(x_i - r_i^{(\lambda)}), 1 \le i \le n,$$

$$\sum_{i=1}^{n}(x_i - r_i^{(\lambda)}) = 0.$$

End For

(2) PROTOCOL-*multi-commitment*:

The purpose of this protocol is for $n+1$ users (e.g., $C_1, \cdots, C_n$ and V_t) to cooperatively determine a random number $e_t \in [1, n]$. Assume V_t is an initiator.

Committing

1. V_t asks each C_i, $1 \le i \le n$, to compute a bit commitment $\hat{M}_{it}$ for a random number $M_{it} \in Z_q$ and publish $\hat{M}_{it}$.
2. V_t computes a bit commitment $\hat{R}_t$ for a random number $R_t \in Z_q$ and also publishes $\hat{R}_t$.

Opening

1. All C_i's and V_t open their own commitments $\hat{M}_{it}$ and $\hat{R}_t$ respectively by revealing the keys used in committing.
2. V_t obtains e_t by computing $e_t = f(M_{1t} \oplus M_{2t} \cdots \oplus M_{nt} \oplus R_t)$, where f is an one-way hash function and $1 \le f(x) \le n$.

(3) PROTOCOL-*prove-subtally*:

Let q' be a large prime and $q' | q - 1$ (i.e., $q = hq' + 1$). Assume the prover has a secret key/public key pair (ϵ, y), where $y = g^\epsilon \pmod q$ and g is a public parameter. The value g is defined as $g = (g')^h \mod q$, where g' is a generator in Z_q, to avoid the attack in [18]. The prover publishes the following pairs of encrypted messages: $\{(E(x_{11}), E(x_{12})), (E(x_{21}), E(x_{22})), \cdots, (E(x_{u1}), E(x_{u2}))\}$, and $\{(E'(x'_{11}), E'(x'_{12})), (E'(x'_{21}), E'(x'_{22})), \cdots, (E'(x'_{u1}), E'(x'_{u2}))\}$, where $\{E, E'\}$ belongs to a family of compatible additive homomorphic encryptions and no one has any idea about the decryptions of these two encryptions. Assume the prover knows only one message, x_{id_i}, out of each pair (x_{i1}, x_{i2}) and similarly x'_{id_i}, out of (x'_{i1}, x'_{i2}), for $1 \le i \le u$. In other words, $d_i \in \{1, 2\}$, the secret of prover, indicates the d_ith message known by the prover in the ith pair. Given a value T, the prover wants to show $T = \sum_{i=1}^{u}(x_{id_i} + x'_{id_i})$ without revealing the secret (x_{i1}, x_{i2}), (x'_{i1}, x'_{i2}) and d_i.

To perform this proof, the prover has to hide the encrypted messages $(E(x_{i1}), E(x_{i2}))$ and $(E'(x'_{i1}), E(x'_{i2}))$ using his public key. He chooses $4u$ random numbers r_{i1}, r_{i2}, r'_{i1}, and $r'_{i2} \in Z^*_{q'}$ $(1 \le i \le u)$ and u private permutations, $\pi_i(\cdot)$ $(1 \le i \le u)$, from $\{1, 2\}$ onto $\{1, 2\}$ to generate:

$H_{i,\pi_i(1)} = (H^{(1)}_{i,\pi_i(1)}, H^{(2)}_{i,\pi_i(1)}) =$
$(g^{r_{i1}} \bmod q,\ E(x_{i1})y^{r_{i1}} \bmod q)$,
$H_{i,\pi_i(2)} = (H^{(1)}_{i,\pi_i(2)}, H^{(2)}_{i,\pi_i(2)}) =$
$(g^{r_{i2}} \bmod q,\ E(x_{i2})y^{r_{i2}} \bmod q)$, and
$H'_{i,\pi_i(1)} = (H^{'(1)}_{i,\pi_i(1)}, H^{'(2)}_{i,\pi_i(1)}) =$
$(g^{r'_{i1}} \bmod q,\ E'(x'_{i1})y^{r'_{i1}} \bmod q)$,
$H'_{i,\pi_i(2)} = (H^{'(1)}_{i,\pi_i(2)}, H^{'(2)}_{i,\pi_i(2)}) =$
$(g^{r'_{i2}} \bmod q,\ E'(x'_{i2})y^{r'_{i2}} \bmod q)$.

The prover than publishes all $(H_{i,\pi_i(1)}, H_{i,\pi_i(2)})$, $(H'_{i,\pi_i(1)}, H'_{i,\pi_i(2)})$, $\hat{d}_i$ $(= \pi_i(d_i))$, and A $(= \sum_i(r_{id_i} + r'_{id_i}))$. Note that publishing $\hat{d}_i$ leaks no information about the real index d_i because the prover uses $\pi_i(\cdot)$ to permute the original orders of $(E(x_{i1}), E(x_{i2}))$ and $(E'(x'_{i1}), E'(x'_{i2}))$. This proof will be given in the following two stages:

Phase 1: Prove Shuffling

The prover shows $(H_{i,\pi_i(1)}, H_{i,\pi_i(2)})$ and $(H'_{i,\pi_i(1)}, H'_{i,\pi_i(2)})$ are shuffled from $(E(x_{i1}), E(x_{i2}))$ and $(E'(x'_{i1}), E'(x'_{i2}))$ respectively in the above manner.

For $\lambda = 1, 2 \cdots \omega$

1. The prover randomly chooses $r_{i1}^{(\lambda)}, r_{i2}^{(\lambda)}, r_{i1}^{'(\lambda)}, r_{i2}^{'(\lambda)} \in Z_{q'}^*$, for $1 \le i \le u$. Then he posts $Y_{i,\pi_i^{(\lambda)}(1)} = (Y^{(1)}_{i,\pi_i^{(\lambda)}(1)}, Y^{(2)}_{i,\pi_i^{(\lambda)}(1)}) = (g^{r_{i1}^{(\lambda)}} \bmod q,\ E(x_{i1})y^{r_{i1}^{(\lambda)}} \bmod q)$, $Y_{i,\pi_i^{(\lambda)}(2)} = (Y^{(1)}_{i,\pi_i^{(\lambda)}(2)}, Y^{(2)}_{i,\pi_i^{(\lambda)}(2)}) = (g^{r_{i2}^{(\lambda)}} \bmod q,\ E(x_{i2})y^{r_{i2}^{(\lambda)}} \bmod q)$, and $Y'_{i,\pi_i^{(\lambda)}(1)} = (Y^{'(1)}_{i,\pi_i^{(\lambda)}(1)}, Y^{'(2)}_{i,\pi_i^{(\lambda)}(1)}) = (g^{r_{i1}^{'(\lambda)}} \bmod q,\ E'(x'_{i1})y^{r_{i1}^{'(\lambda)}} \bmod q)$, $Y'_{i,\pi_i^{(\lambda)}(2)} = (Y^{'(1)}_{i,\pi_i^{(\lambda)}(2)}, Y^{'(2)}_{i,\pi_i^{(\lambda)}(2)}) = (g^{r_{i2}^{'(\lambda)}} \bmod q,\ E'(x'_{i2})y^{r_{i2}^{'(\lambda)}} \bmod q)$,where $\pi_i^{(\lambda)}(\cdot)$ is a private permutation from $\{1,2\}$ onto $\{1,2\}$ used in the λ-th iteration.
2. The verifier has the following two choices:
   2.1 Asks the prover to reveal $r_{i1}^{(\lambda)}$, $r_{i2}^{(\lambda)}$, $r_{i1}^{'(\lambda)}$ and $r_{i2}^{'(\lambda)}$, for $1 \le i \le u$, and then checks that Y_{i1}, Y_{i2}, Y'_{i1} and Y'_{i2} are constructed as above correctly.
   2.2 Asks the prover to reveal $(r_{i1}^{(\lambda)} - r_{i1})$, $(r_{i2}^{(\lambda)} - r_{i2})$, $(r_{i1}^{'(\lambda)} - r'_{i1})$ and $(r_{i2}^{'(\lambda)} - r'_{i2})$, for $1 \le i \le u$, and then checks if the following equations hold.

$$H^{(1)}_{i,\pi_i(1)} \cdot g^{(r_{i1}^{(\lambda)} - r_{i1})} = Y^{(1)}_{i,\pi_i^{(\lambda)}(1)} \pmod q$$

$$H^{(1)}_{i,\pi_i(2)} \cdot g^{(r_{i2}^{(\lambda)} - r_{i2})} = Y^{(1)}_{i,\pi_i^{(\lambda)}(2)} \pmod q$$

$$H^{(2)}_{i,\pi_i(1)} \cdot y^{(r_{i1}^{(\lambda)} - r_{i1})} = Y^{(2)}_{i,\pi_i^{(\lambda)}(1)} \pmod q$$

$$H^{(2)}_{i,\pi_i(2)} \cdot y^{(r_{i1}^{(\lambda)} - r_{i1})} = Y^{(2)}_{i,\pi_i^{(\lambda)}(2)} \pmod q$$

and

$$H^{'(1)}_{i,\pi_i(1)} \cdot g^{(r_{i1}^{'(\lambda)} - r'_{i1})} = Y^{'(1)}_{i,\pi_i^{(\lambda)}(1)} \pmod q$$

$$H^{'(1)}_{i,\pi_i(2)} \cdot g^{(r_{i2}^{'(\lambda)} - r'_{i2})} = Y^{'(1)}_{i,\pi_i^{(\lambda)}(2)} \pmod q$$

$$H^{'(2)}_{i,\pi_i(1)} \cdot y^{(r_{i1}^{'(\lambda)} - r'_{i1})} = Y^{'(2)}_{i,\pi_i^{(\lambda)}(1)} \pmod q$$

$$H^{'(2)}_{i,\pi_i(2)} \cdot y^{(r_{i1}^{'(\lambda)} - r'_{i1})} = Y^{'(2)}_{i,\pi_i^{(\lambda)}(2)} \pmod q$$

End for

3. If the above steps are correct, the verifier computes the following values according to the $\hat{d}_i$ published by the prover: $\prod_{i=1}^{u} H^{(1)}_{i,\hat{d}_i} = g^{A_1} \pmod q$; $\prod_{i=1}^{u} H^{(2)}_{i,\hat{d}_i} = E(T_1)y^{A_1} \pmod q$ and $\prod_{i=1}^{u} H^{'(1)}_{i,\hat{d}_i} = g^{A_2} \pmod q$; $\prod_{i=1}^{u} H^{'(2)}_{i,\hat{d}_i} = E'(T_2)y^{A_2} \pmod q$, where $A_1 = \sum_{i=1}^{u} r_{id_i}$, $A_2 = \sum_{i=1}^{u} r'_{id_i}$, $T_1 = \sum_{i=1}^{u} x_{id_i}$ and $T_2 = \sum_{i=1}^{u} x'_{id_i}$.

Phase 2: Prove Sum(T)

Let $W_1 = (W_{11}, W_{12}) = (g^{A_1} \bmod q,\ E(T_1)y^{A_1} \bmod q)$, $W_2 = (W_{21}, W_{22}) = (g^{A_2} \bmod q,\ E'(T_2)y^{A_2} \bmod q)$.

The prover shows $T = T_1 + T_2$ as follows:

1. The verifier checks $W_{11}W_{21} = g^A \pmod q$.

For $\lambda = 1, 2, \cdots \omega$

2. The prover randomly chooses $r^{(\lambda)} \in Z_q$ and computes a bit commitment $R^{(\lambda)}$ for $r^{(\lambda)}$. Then he posts $(R^{(\lambda)}, Y_1^{(\lambda)}, Y_2^{(\lambda)})$, where

$$Y_1^{(\lambda)} = E(T_1 + r^{(\lambda)})y^{A_1} = W_{12}E(r^{(\lambda)})$$

$$Y_2^{(\lambda)} = E'(T_2 - r^{(\lambda)})y^{A_2} = W_{22}(E'(r^{(\lambda)}))^{-1}.$$

3. The verifier has the following two choices:
   3.1 Asks the prover to reveal $r^{(\lambda)}$ and then checks that $r^{(\lambda)}$ is consistent with $R^{(\lambda)}$, and $Y_1^{(\lambda)}$ and $Y_2^{(\lambda)}$ are constructed as above correctly.
   3.2 Asks the prover to reveal $(T_1 + r^{(\lambda)})$ and $(T_2 - r^{(\lambda)})$ and then checks if the following equations hold.

$$(T_1 + r^{(\lambda)}) + (T_2 - r^{(\lambda)}) = T$$

$$Y_1^{(\lambda)}Y_2^{(\lambda)} = E(T_1 + r^{(\lambda)})E'(T_2 - r^{(\lambda)})y^A$$

End for

4. If the above steps are correct, the verifier accepts

$$T = T_1 + T_2 = \sum_{i=1}^{u} (x_{id_i} + x'_{id_i})$$

Theorem 1 The prover can successfully forge the T in the *prove-subtally* protocol with negligible probability of $2^{-\omega}$.

Proof. (Proof will be given in the final paper.)

Theorem 2 If the prover follows the prove-subtally protocol, the secret d_i cannot be revealed.
Proof. (Proof will be given in the final paper.)

3.2 The New Voting Scheme

Assume that the system has n centers $C_1, C_2, \cdots C_n$, and m candidates. All centers firstly agree on a randomly selected family of partially compatible additive homomorphic encryptions $\{E_1, E_2, \cdots, E_n, E_a, E_b, E_\varphi\}$, and have no idea about the decryptions of these encryptions. To vote for each candidate k, each voter V_t, for $1 \le t \le \sigma$, chooses $n+3$ different sub-votes $x_i^{(t,k)}$, $1 \le i \le n+3$, such that $Z_1^{(t,k)}(= x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_n^{(t,k)}+x_{n+1}^{(t,k)})$, $Z_2^{(t,k)}(= x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_{n-1}^{(t,k)}+x_{n+2}^{(t,k)}+x_{n+3}^{(t,k)}) \in \{1,-1\}$ $(1 \le k \le m)$ and $Z_1^{(t,k)} = -Z_2^{(t,k)}$. If $Z_i^{(t,k)} = +1$ then it represents a "yes" vote , and vice versa. These $n+3$ subvotes are encrypted with the predetermined family of partially compatible additive homomorphic encryptions mentioned above. These encryptions of all sub-votes are published for verification. Upon voting, the voter chooses the subset of votes which sum up to either $Z_1^{(t,k)}$ or $Z_2^{(t,k)}$ depending on his choice for a 'yes' or a 'no' vote. Then, each voter privately sends two subvotes, $(x_n^{(t,k)}, x_{n+1}^{(t,k)})$ or $(x_{n+2}^{(t,k)}, x_{n+3}^{(t,k)})$, via a physical untappable channel [20] to the center C_{e_t} $(1 \le e_t \le n)$, which is cooperatively determined by the voter and all centers. C_{e_t} is so selected to achieve receipt-freeness as will be discussed in Section 4. The other subvotes are distributed randomly to the other $n-1$ centers , one for each center, via a secure channel implemented by cryptographic method. Finally, each center counts his subtally and publishes it. Then he proves the correctness of his subtally without revealing the voter's choice between $Z_1^{(t,k)}$ and $Z_2^{(t,k)}$ by performing *prove-subtally* protocol. The new scheme is described as follows by three phases: precomputation, voting ,and vote-counting:

Precomputation

Centers : $C_1, C_2, \cdots C_n$

Voters : $V_1, V_2, \cdots V_\sigma$

The possible votes of voter V_t for candidate k are $Z_1^{(t,k)}$ and $Z_2^{(t,k)}$, $1 \le k \le m$.

Step 1. Each voter V_t chooses $n+3$ sub-votes $\{x_i^{(t,k)}\}_{i=1,\cdots,n+3}$ to form $Z_1^{(t,k)}, Z_2^{(t,k)} \in \{1,-1\}$ such that $Z_1^{(t,k)} = x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_n^{(t,k)}+x_{n+1}^{(t,k)} = A_1^{(t,k)}+B_1^{(t,k)}$ $Z_2^{(t,k)} = x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_{n-1}^{(t,k)}+x_{n+2}^{(t,k)}+x_{n+3}^{(t,k)} = A_2^{(t,k)}+B_2^{(t,k)}$, where A_i, B_i, $1 \le i \le 2$, are arbitrary integers which add up to $Z_i^{(t,k)}$. He publishes $\{E_i(x_i^{(t,k)})\}_{i=1,2,\cdots,n}$, $E_\varphi(x_{n+1}^{(t,k)})$, $E_n(x_{n+2}^{(t,k)})$, $E_\varphi(x_{n+3}^{(t,k)})$, $E_a(A_1^{(t,k)})$, $E_a(A_2^{(t,k)})$, $E_b(B_1^{(t,k)})$, and $E_b(B_2^{(t,k)})$.

Step2. V_t proves the validity of the following equations to any trusted authority by performing *prove-sum* protocol as described in Section 2.1: $x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_n^{(t,k)}+x_{n+1}^{(t,k)} = A_1^{(t,k)}+B_1^{(t,k)}$ $x_1^{(t,k)}+x_2^{(t,k)}+\cdots+x_{n-1}^{(t,k)}+x_{n+2}^{(t,k)}+x_{n+3}^{(t,k)} = A_2^{(t,k)}+B_2^{(t,k)}$.

Step3. V_t shows that both $A_1^{(t,k)}+B_1^{(t,k)}$ and $A_2^{(t,k)}+B_2^{(t,k)}$ are in $\{1,-1\}$ using *prove* ± 1 protocol, and then shows that $(A_1^{(t,k)}+B_1^{(t,k)})+(A_2^{(t,k)}+B_2^{(t,k)}) = 0$ by performing *prove-Q* protocol (set $Q=0$) as described in Section 3.1.

Voting

Step1. V_t performs *multi-commitment* protocol with $C_1, C_2, \cdots, C_n$ to get a random value $e_t \in [1,n]$.

Step2. V_t privately sends $x_i^{(t,k)}$ to C_i $(1 \le i \le n-1, i \ne e_t)$ and $x_{e_t}^{(t,k)}$ to C_n (if $e_t \ne n$). Especially, he sends one of $(x_n^{(t,k)}, x_{n+1}^{(t,k)})$ and $(x_{n+2}^{(t,k)}, x_{n+3}^{(t,k)})$ to C_{e_t} depending on his decision to candidate k , through a physical untappable channel (for simplicity, we assume V_t sends the centers his subvotes by this way instead of a random choice). See **Figure 1.**

Vote-counting

Step1. Each center C_i, $(1 \le i \le n-1, i \ne e_t)$, verifies $x_i^{(t,k)}$'s for all t and k whether they are consistent with $E_i(x_i^{(t,k)})$ or not. C_n verifies the correctness of $x_{e_t}^{(t,k)}$ with $E_{e_t}(x_{e_t}^{(t,k)})$ (if $e_t \ne n$). Especially, C_{e_t}, for voter V_t and candidate k, verifies one of the two pairs, $(x_n^{(t,k)}, x_{n+1}^{(t,k)})$ and $(x_{n+2}^{(t,k)}, x_{n+3}^{(t,k)})$, to see if it is consistent with $\left(E_n(x_n^{(t,k)}), E_\varphi(x_{n+1}^{(t,k)})\right)$ or $\left(E_n(x_{n+2}^{(t,k)}), E_\varphi(x_{n+3}^{(t,k)})\right)$.

Step2. Each C_i computes and posts his subtally for the candidate k $T_i^{(k)} = \sum_{t\in[1,\sigma],e_t=i}(x_{s_1^{(t,k)}}^{(t,k)} + x_{s_2^{(t,k)}}^{(t,k)}) +$
$\begin{cases} \sum_{t\in[1,\sigma],e_t\ne i} x_i^{(t,k)}, & \text{for } i = 1,2\cdots,n-1 \\ \sum_{t\in[1,\sigma],e_t\ne i} x_{e_t}^{(t,k)}, & \text{for } i = n \end{cases}$

, where $(s_1^{(t,k)}, s_2^{(t,k)})$ decided by V_t equals to either $(n, n+1)$ or $(n+2, n+3)$. Besides, C_n has to publish $G_j^{(k)} = \sum_{t\in[1,\sigma], e_t=j} x_j^{(t,k)}$, $1 \le j \le n-1$, for further verification.

Step3. Each C_i, $1 \le i \le n$, proves the correctness of his posted subtally $T_i^{(k)}$. Let $\hat{T}_i^{(k)} = \sum_{t\in[1,\sigma], e_t=i} (x_{s_1^{(t,k)}}^{(t,k)} + x_{s_2^{(t,k)}}^{(t,k)})$. The center C_i publishes $\hat{T}_i^{(k)}$ and proves it is correct without revealing $(s_1^{(t,k)}, s_2^{(t,k)})$ by performing the *prove-subtally* protocol. Note that in this case, $\{E_n, E_\varphi\}$ corresponds to $\{E, E'\}$ in the *prove-subtally* protocol. Similarly, $\hat{T}_i^{(k)}$ corresponds to T and $\{(x_n^{(t,k)}, x_{n+2}^{(t,k)}), (x_{n+1}^{(t,k)}, x_{n+3}^{(t,k)})\}$ corresponds to $\{(x_{i1}, x_{i2}), (x'_{i1}, x'_{i2})\}$. Then everyone can check if the following equations hold: $E_i(T_i^{(k)} - \hat{T}_i^{(k)}) = \prod_{t\in[1,\sigma], e_t\neq i} E_i(x_i^{(t,k)})$, for $i = 1, 2\cdots, n-1$, $E_j(G_j^{(k)}) = \prod_{t\in[1,\sigma], e_t=j} E_j(x_j^{(t,k)})$, for $1 \le j \le n-1$, and $T_n^{(k)} = \hat{T}_n^{(k)} + \sum_{j=1}^{n-1} G_j^{(k)}$.

Step4. The total tally of the vote for candidate k is

$$T^{(k)} = \sum_{i=1}^{n} T_i^{(k)}.$$

3.3 The Property of Limiting the Number of 'Yes' Votes

Th new scheme can be easily extended to limit the maximum number of "yes" votes that each voter can cast to a threshold value Γ. For $m(\ge 2)$ candidates, the sum of all votes for each voter is less than or equal to $2\Gamma - m$. This property is important for our scheme to implement a practical election. The modifications are described by adding some extra steps into the precomputation phase and voting phase as the following:

Precomputation

Step4. V_t performs *prove-Q* protocol to show that the following equation holds (set $Q = Q_t$):

$$\sum_{k=1}^{m}\sum_{i=1}^{n-1} x_i^{(t,k)} = Q_t$$

Voting

Step3. The center C_{et} publishes a value δ_t for V_t and performs the protocol *prove-subtally* to prove

$$\sum_{k=1}^{m} (x_{s_1^{(t,k)}}^{(t,k)} + x_{s_2^{(t,k)}}^{(t,k)}) = \delta_t$$

without revealing $(s_1^{(t,k)}, s_2^{(t,k)})$.

Step4. Each one then checks if $Q_t + \delta_t \le 2\Gamma - m$.

4. SECURITY ANALYSIS AND DISCUSSION

The security of the proposed scheme is based on the security of the family of partially compatible homomorphic encryptions. Besides, this scheme not only provides the receipt-free property, but also meets the general security requirements of electronic voting systems [12].

Definition 0.1 *A electronic voting scheme is* **secure** *and* **receipt-free** *if it satisfies the following requirements [12]:*

1. *Completeness: all valid votes are tallied correctly.*
2. *Soundness: a dishonest voter cannot disrupt the voting.*
3. *Verifiability: no one can fake the result of the voting.*
4. *Privacy: the privacy of the voter is preserved.*
5. *Unreusability: no voter can vote twice.*
6. *Fairness: nothing can affect the voting.*
7. *Receipt-freeness: the voter has no receipt to prove what he had voted after voting.*

Theorem 3 (**Completeness**) If the voter casts a valid vote, it will be tallied correctly.
Proof. (Proof will be given in the final paper.)

Theorem 4 (**Soundness**) A dishonest voter disrupts the election that will be detected by other voters or centers.
Proof. (Proof will be given in the final paper.)

Theorem 5 (**Verifiability**) Even if all centers collude, they cannot forge the result of the election.
Proof. (Proof will be given in the final paper.)

Theorem 6 (**Privacy**) If two centers or more are honest, the privacy of the voter can be preserved.
Proof. (Proof will be given in the final paper.)

Theorem 7 (**Unreusability**) No voter can vote twice in the voting scheme.
Proof. (Proof will be given in the final paper.)

Theorem 8 **(Fairness)** If two centers or more are honest, no one can get any information about the tally result before the vote-counting phase.
Proof. (Proof will be given in the final paper.)

Theorem 9 **(Receipt-freeness)** If the center C_{e_t} is honest, the voter V_t cannot prove what he had vote after voting.
Proof. (Proof will be given in the final paper.)

5. CONCLUSIONS

This paper proposed a receipt-free voting scheme based on partially compatible homomorphisms. The new scheme is also extended to have the property of *limiting the maximum number of 'yes' votes.* However, for the voter V_t, the trustworthyness of C_{e_t} is essential to the security of receipt-freeness. How to split the control factors of receipt-freeness over all n centers is a further research problem.

Acknowledgement. This work was supported by the National Science Council of Republic of China under the contract number NSC88-2213-E006-008.

6. REFERENCES

[1] Benaloh, J.C., Tuinstra, D., "Receipt-free secret-ballot elections", STOCK 94, pp. 544-553, 1994.

[2] Benaloh, h., Yung, M., "Distributing the power of a government to enhance the privacy of voters", ACM Symposium on Principle of Distributed Computing, pp.52-62, 1986.

[3] Boyd, C., "Some Applications of Multiple Key Ciphers", Proceedings of Eurocrypt'88, Springer-Verlag, pp. 455-467, 1988.

[4] Boyd, C., "A New Multiple Key Cipher and an Improved Voting Scheme", Proceedings of Eurocrypt'89, Springer-Verlag, pp.617-625, 1989.

[5] Chaum, D., "Untraceable electronic mail, return address, and digital pseudonyms", Communications of the ACM, pp. 84-88, 1981.

[6] Chaum, D., "Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA", Advances in Cryptology - Eurocrypt'88, Springer-Verlag, pp. 177-182, 1988.

[7] Chaum, D., "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability", Journal of Cryptology, Vol.1, No.1, pp. 65-75, 1988.

[8] Chaum, D., Crepeau, C., and Damgard, I., "Multiparty Unconditionally Secure Protocols", Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pp. 11-19, May, 1988.

[9] Cohen, J. and Fisher, M., "A Robust and Verifiable Cryptographically Secure Election Scheme", 26th Annual Symposium on Foundations of Computer Science, IEEE, pp. 372-382.

[10] Cramer, R., Franklin, M., Schoenmakers, B., and Yung, M., "Multi-Authority Secret-Ballot Elections with Linear Work", Advances in Cryptology - Eurocrypt'96, Springer-Verlag, pp. 72-83, 1996.

[11] Fiat, A. Shamir,A., "How to prove yourself: Practical solutions to identification and signature problems", Advances in Crypto'86, Springer-Verlag, pp. 186-199, 1986.

[12] Fujioka, A., Okamoto, T., Ohta, K., "A practical secret voting scheme for large scale elections", Advances in Cryptology - Auscrypt'92, pp.244-251, 1992.

[13] Iversen, K. R., "A cryptographic scheme for computerized general elections", Advances in Cryptology - Crypto'91, Springer-Verlag, pp. 405-419, 1991.

[14] Juang, W., Lei, C. L., and Fan, C. I, "A Secure and Practical Electronic Voting Scheme for Real World Environments", Proceedings of National Information Security Conference 1996, R. O. C.

[15] Kurosawa, K., Tsujii, S., "Multi-language zero knowledge interactive proof systems", Advances in Cryptology - Crypto'90, pp.339-352, 1991.

[16] Niemi, V., Renvall, A., "How to prevent buying of votes in computer elections", Asiacrypt'94, pp.141-148, 1994.

[17] Park, C., Itoh, K., Kurosawa, K., "All/Nothing election scheme and anonymous channel", Advances in Cryptology - Eurocrypt'93, pp.248-259, 1993.

[18] Pfitzmann, B.,"Breaking an Efficient Anonymous Channel", Advances in Cryptology - Eurocrypt'94, pp. 339-348, 1994.

[19] Sako, K., Kilian, J., "Secure voting using partially compatible homomorphisms", Advances in Cryptology - Crypto'94, Springer-Verlag, pp. 411-424, 1994.

[20] Sako, K., Kilian, J., "Receipt-free mix-type voting scheme ", Advances in Cryptology - Eurocrypt'95, Springer-Verlag, pp. 393-403, 1995.

[21] Slessenger, P. H., "Socially Secure Cryptographic Election Scheme", Electronics Letters, Vol. 27, No. 11, pp. 955-957, May, 1991.