A Small-Scale Voting Protocol Hiding Vote-Counts of All Candidates

Pei-yih Ting and Po-Yueh Hung
Department of Computer Science,
National Taiwan Ocean University
{pyting, m92570002}@mail.ntou.edu.tw

Abstract

In this paper, we focus on the design of the winner-determination procedure of an electronic voting protocol used at critical elections, e.g. at the meeting of the board of a company for critical business decisions or a parliamentary committee for legislation. The number of participating voters is limited to several hundreds but the voting should satisfy a new privacy requirement that the accumulated vote-counts of all candidates should be kept as secret as possible. This additional requirement is significant only for small/medium-scale elections. Traditional electronic voting frameworks simply take the announcement of vote-counts for granted and hope that each individual's actual vote is hidden in the accumulated vote-counts. Therefore, it is not easy to modify an existing scheme to approach this new goal. In the proposed protocol, the homomorphic ElGamal cryptosystem is used. An electronic bulletin board holds public announced values. A ballot consists of separate encrypted 'yes'/'no' vote for each candidate such that the accumulated vote-counts can be calculated from the ciphertexts without any decryption. The correctness of each ballot is guaranteed through ZKPs. The accumulated vote-count ciphertexts are then converted to encrypted unary representation through a mix-and-match sub-protocol such that the vote-counts can be concealed in the winner-determination stage. This protocol is suited for both equal-voting and weighted-voting schemes. Also, the type of voter's selection can be single choice, multiple choices, ranking choice, or the allocative choice.

Keywords: e-voting privacy, ElGamal encryption system, homomorphic encryption, mix-and-match ciphertext conversion

1. Introduction

Consider the following scenarios: In a small meeting room, members of the board of a company are voting for the chairman and the vice chairman of the next term. Each one can vote for two candidates without priority in his ballot. The company organization rule specifies additionally that the winning candidate must have majority supports. After all the anonymous ballots are disclosed and votes tallied, the winners are fairly generated. However, the disclosed ballots leave many intricate traces that might have significant political influences to the management of the company, which is particularly detrimental to the cooperative framework of the company. For example, if the vote count for the chairman is just above the legal margin of majority, some members might challenge the competence of the

winner's leadership. If some candidates who actively campaigned for the position and were promised privately a certain amount of support before the election, but the results turns out to have less supporting votes, suspicion and disgust might arise. In such a situation, a traditional solution is by resorting to a trusted third party who counts all votes secretly and announces only the resulting winners.

A second situation happens in parliamentary elections in which members are representatives from various political parties. There might be some predetermined principles on a certain topic for each member of a political party to obey. Sometimes these principles might be against the individual judgment of a member or against the benefits of most people in the country. However, the public ballot-disclosing vote-counting process forces most participants to obey the principles of their political party. Otherwise, the accumulated vote-counts would reveal some traces about possible traitors. In such a situation, extensive discussions, which usually appear at a meeting before the vote, seem to be superfluous. The seemingly fair public vote-counting process actually prevents voters from voting according to his sober and independent judgment. Somehow, the advantage of voting as a better collective decision method vanishes. Although it is well known that many independent decisions with correctness probability slightly higher than a half will accumulate to a good decision with correctness probability far away from a half and hopefully close to unity. In many such occasions, the interests of the whole company or country could be sacrificed for maintaining the superficially peaceful and cooperative atmosphere. In this paper, we would like to present secret and fair vote-counting and winner-determination procedures that do not disclose the vote-count of any individual candidate. The protocol hides as much information as possible except the names of the winning candidates. At the same time, it is universally verifiable and thus achieves the fairness requirement.

In the past decade, there were extensive researches on electronic voting schemes, [3], [9], [17], [13], [4], [19], [18] to name a few. These schemes make a general electronic election for thousands or millions of voters feasible. The primary achievements include maintaining the privacy of each independent votes, preserving the fairness, preventing vote-buying and coercion, realizing vote-and-go concepts while reducing the computation and communication of the voting centers and the voters. However, while focusing on achieving the above properties, all electronic voting schemes announce the final vote-count of each candidate in order to convince all participants that the winner is determined fairly. For some systems, this announcement is the by-product in the course to achieve effectiveness. For some other schemes, this announcement leaves sufficient auditing traces to ensure the fairness of the trusted authorities. However, this type of vote-counting procedure maintains the privacy of each voter’s independent choice only when the number of voters is sufficiently large. For those schemes[3, 13] based on homomorphic encryption systems, it is theoretically possible to employ the proposed winner-determination procedure to hide the accumulated vote-counts. However, this is completely unnecessary for large-scale elections. For other schemes, it would be hard to accommodate them to achieve our new privacy requirement.

From the aspect of saving computation efforts, there is no obvious reason to automate electronically a voting scheme for small/moderate-scale elections. However, if we rely on a trusted person to compare all votes secretly in the anonymous voting scenario where vote-counts are also required to be secret, this trusted person is very likely to be bribed or

coerced later. Therefore, a good cryptographic vote-counting and winner-determination scheme as proposed in this paper finds itself unique valuation in this application scenario.

In the proposed protocol, the ElGamal cryptosystem[7] is used with its nice homomorphic property. Each ballot consists of separate encrypted 'yes'/'no' vote for each candidate such that the vote-counts can be accumulated without any decryption. The correctness of each encrypted ballot is guaranteed with zero knowledge proofs. The accumulated vote-count ciphertexts are then converted to encrypted unary representations[2] through a 'mix-and-match' sub-protocol[14] such that the vote-counts can be concealed in the secret winner-determination stage. The proposed protocol then compares these encrypted vote-counts to determine the winner or a set of R winners with highest vote-counts. All the public values in the proposed protocol are announced on an electronic bulletin board in such a way that every participant can write in his specified fields along the process of the protocol but cannot modify other's fields. We also formulate the ballot representations for the ranking-choice voting in which each voter gives m candidates a ranking from 1 to m and the allocative-choice voting in which each voter gives m candidates separate satisfaction scores in the range 0-100 with the constraint that all scores sum up to 100.

In section 2, we introduce the underlying cryptographic primitives. In section 3, we present the proposed secure voting protocol and especially the vote-counting and winner-determination procedures. Some variations, security analysis and performance issues are discussed in section 4. Section 5 contains the concluding remark.

2. Cryptographic Primitives

2.1 Secret Sharing Scheme

In our protocol, we use a variation of Pederson's (t, n) threshold secret sharing scheme[16, 12] which features a key sharing protocol without any trusted dealer. Each voter V_i chooses a random degree-(t-1) polynomial $f_i(\theta) = x_i + \sum_{j=1}^{t-1} a_{ij}\theta^j$ and announces $g^{a_{ij}}$ where g is the generator of the subgroup in the ElGamal system. A voter V_h shares his secret x_h with another voter V_i by sending $s_{hi}=f_h(i)$ to V_i. On receiving a share s_{hi}, V_i verifies it with announced public values. After voter V_i receives all correct sub-shares s_{hi}, he applies the secret sharing homomorphism to obtain the t-share of the real secret $X = \sum_{i=1}^{n} x_i = F(0)$ where the degree t-1 super polynomial $F(\theta)$ is defined as $\sum_{i=1}^{n} f_i(\theta)$.

Let $T=\{V_p\}_{p=1,\ldots,k}$, $t \leqq k \leqq n$, be the set of cooperative voters. To jointly decrypt an ElGamal ciphertext (α, β) on the bulletin board, a voter V_p in the set T needs only compute independently $\alpha^{F(p)\prod_{q\neq p, q\in T}(-q)/(p-q)}$ and publishes it.

The decrypted plaintext is $D(\alpha,\beta) = \beta/\alpha^{F(0)} = \beta/\left(\prod_{p\in T}\alpha^{F(p)\prod_{q\neq p, q\in T}(-q)/(p-q)}\right)$. To thwart disruptions by malicious participants, the following ZKP should be supplied by V_p for proving that the F(p) used in $\alpha^{F(p)\prod_{q\neq p, q\in T}(-q)/(p-q)}$ is consistent with all previous published values.

Augmented ZKP for joint decryption in Pederson's threshold scheme:

In the joint decryption stage, each voter needs only calculate $\alpha^{F(p)}$ of the partial decrypted value $\alpha^{F(p)\prod_{q\neq p,\, q\in T}(-q)/(p-q)}$. The value $\prod_{q\neq p,\, q\in T}(-q)/(p-q)$ depends on the identifiers p, q of the actual cooperating set T of voters but is public. To save some efforts in detecting malicious voters who claim to be cooperative but submit invalid partial decrypted values. A ZKP is submitted by each voter taking part in the joint decryption: First, we note that $g^{f_q(p)}$ can be calculated from published values and therefore $g^{F(p)}= g^{\sum_q f_q(p)}$ can be calculated from published values. Second, $\alpha^{F(p)}$ is the primary part of the submitted partial decrypted value. Therefore, a voter V_p can prove that he uses the same exponent F(p) of the submitted value $\alpha^{F(p)}$ as the exponent of $g^{F(p)}$ with an "equal discrete log" ZKP[5].

2.2 The Ballots

In the following, we present the format of the ballot for an equal-voting scheme where the votes of all voters are equally weighted. Each voter V_i encrypts his 'yes'/'no' vote for each candidate as a vector of ElGamal ciphertexts, denoted as C_i:

$C_i = (c_{i1}, ..., c_{ij}, ..., c_{im}) = (E_K(z), E_K(1), ..., E_K(z), E_K(z), ..., E_K(1))$, $1 \leqq i \leqq n$, $1 \leqq j \leqq m$, n is the number of voters and m is the number of candidates. An $E_K(z)$ in the j-th component of C_i represents a 'yes' vote for the j-th candidate, and an $E_K(1)$ represents a 'no' vote. Note that each $E_K(z)$ in the above notation represents a different ciphertext of the plaintext z with different encryption random number. So is $E_K(1)$. There are w_i 'yes' votes in each C_i where w_i is an integer between zero and L, where L is the allowed maximum number of 'yes' votes in each ballot. Note that 'yes' votes for different candidates have equal weight.

Each voter must prove that his encrypted vote vector C_i is valid. It ensures that every voter votes at most L candidates and no candidate obtains multiple 'yes' votes (i.e. $E_K(z^c)$) or multiple 'no' votes (i.e. $E_K(z^{-c})$) such that the following homomorphic counting process can proceed correctly. First, a voter must prove that each element c_{ij} of his ballot corresponds to a plaintext which is either z or 1. Second, using the multiplicative homomorphic property of the ElGamal cryptosystem, the voting center can calculate $\chi_i=\prod_{j=1}^{m} c_{ij}$ which equals $E_K(z^{w_i})$. Each voter proves that χ_i decrypts to one element of a finite set of plaintexts $S=\{z^0, z^1, z^2, ..., z^L\}$ with the following '1-out-of-L' ZKP.

'1-out-of-L' ZKP:

Consider the following public parameters of an ElGamal cryptosystem: p, q, g, and K where p and q are large prime numbers, p=2q+1, g is a generator in the order q quadratic residue subgroup G_q of Z_p^*, $K \equiv g^X \pmod p$ is the public key, X is the private key, X is chosen such that gcd(X, p-1) is 2 and the order of K in G_q is q. The pair ($\alpha \equiv g^r \pmod p$, $\beta \equiv M_i \cdot K^r \pmod p$) is an ElGamal ciphertext, where r is a random number in $Z_{p-1}\backslash\{0\}$. S is a

finite set of specified messages $\{M_1, ..., M_J\}$, $M_i \in G_q$. Peggy wants to prove to Victor that she knows i and r, corresponding to the ciphertext (α, β), without revealing them.

1. Peggy picks randomly J-1 values $\{e_j\}_{j\neq i} \in_R Z_{p-1}$ and J-1 values $\{y_j\}_{j\neq i} \in_R Z_{p-1}$, then she computes $\{a_j \equiv g^{y_j} \alpha^{-e_j} (\text{mod } p)\}_{j\neq i}$, $\{b_j \equiv K^{y_j} (M_j/\beta)^{e_j} (\text{mod } p)\}_{j\neq i}$, chooses a random $w \in_R Z_{p-1}$ and calculates $a_i \equiv g^w (\text{mod } p)$, $b_i \equiv K^w (\text{mod } p)$. Finally, she commits $\{a_j, b_j\}_{j=1,...,J}$ to Victor.
2. Victor chooses a random challenge $e \in_R Z_{p-1}$ and sends it to Peggy.
3. Peggy computes $e_i \equiv e - \sum_{j\neq i} e_j (\text{mod } p-1)$, $y_i \equiv w + r \cdot e_i (\text{mod } p-1)$, and sends the response $\{e_j, y_j\}_{j=1,...,J}$ to Victor.
4. Victor checks that $e \equiv \sum_j e_j (\text{mod } p-1)$ and that $g^{y_j} \equiv a_j \cdot \alpha^{e_j} (\text{mod } p)$, $K^{y_j} \equiv b_j \cdot (\beta/M_j)^{e_j} (\text{mod } p)$ for j=1,...,J.

A formal proof of this ZKP can be found in [20].

2.3 Verifiable vector mix-net

A mix-net[1, 15] is a cryptographic device that takes a list of m encrypted data items and outputs a list of m completely different ciphertexts that correspond to an undisclosed permutation of the original data items. The origin of each data item should be hided by the mix-net, therefore, most implementations use a series of independently operated mix-servers to perform secret permutation and re-encryption operations. The mix-server in Jakobsson's Millimix[15] proves its private permutation through a "Disjunctive Schnorr identification protocol" NIZKP, which is a variation of the "1-out-of-L re-encryption" NIZKP[6, 19], and an "equal-discrete log" NIZKP[5] on the product of input and output ciphertexts. The mix-servers in Abe's mix-net[1] jointly prove their concatenated private permutations more efficiently with a graph isomorphism style of NIZKP. Both mix-net schemes use ElGamal ciphertexts as the input data items such that each mix-server can re-encrypt an input ciphertext to a different one without decrypting the input ciphertext and prove the correctness of re-encryption through "equal-discrete log" NIZKPs.

In some applications, the data element to be mixed might be larger than a single block of ElGamal ciphertexts, e.g. e-mails or the ballot vectors C_i in Section 2.2. Common practices divide the data into a sequence of ciphertexts and form a ciphertext vector. If a list of ciphertext vectors are required to be mixed (i.e. permuted and re-encrypted), the previous scheme[15] need to be modified. Note that it is demanded that the order of the components inside a ciphertext vector remains unchanged during the mixing. The modified vector mix-net protocol for each server is as follows:

Step 1. A mix-server S_f receives m n-dimensional ciphertext vectors from its previous server:
$\{ (d^{f-1}_{1,1}, d^{f-1}_{1,2}, d^{f-1}_{1,3}, ..., d^{f-1}_{1,n-1}, d^{f-1}_{1,n}), (d^{f-1}_{2,1}, d^{f-1}_{2,2}, d^{f-1}_{2,3}, ..., d^{f-1}_{2,n-1}, d^{f-1}_{2,n}),$
$..., (d^{f-1}_{m,1}, d^{f-1}_{m,2}, d^{f-1}_{m,3}, ..., d^{f-1}_{m,n-1}, d^{f-1}_{m,n})\}$. The server permutes these vectors, re-encrypts all m×n component ciphertexts $\{d^{f-1}_{i,j}\}$ independently, and outputs the resulting m new ciphertext vectors. As shown in figure 1, each row represents an output vector of the mix-server S_f and is published. Every server and verifier can

calculate homomorphically the product of all ciphertexts of a row to obtain μ_i^f, and the product of all ciphertexts of a column to obtain ω_j^f.

Step 2. The mix-server S_f is then required to provide n NIZKPs that the ciphertext set of the j-th column $\{d^f_{1,j}, d^f_{2,j}, ..., d^f_{m,j}\}$ is permuted and re-encrypted from the previous ciphertext set $\{d^{f-1}_{1,j}, d^{f-1}_{2,j}, ..., d^{f-1}_{m,j}\}$ and that the product ciphertext set $\{\mu^f_{1,j}, \mu^f_{2,j}, ..., \mu^f_{m,j}\}$ is permuted and re-encrypted from the previous product ciphertext set $\{\mu^{f-1}_{1,j}, \mu^{f-1}_{2,j}, ..., \mu^{f-1}_{m,j}\}$. This can be proved in two steps: First, mix-server S_f proves that each element in the input set is re-encrypted from one element of the output set using the "1-out-of-L re-encryption" NIZKP [6, 19], which is basically a generalization of the "equal discrete log" NIZKP. Note that the above proof is in the reverse direction from the input to the output to prevent duplicated output provided that all inputs correspond to different plaintexts. The re-encryption random number used in the proof is the negative (mod p-1) of the original random number. Second, mix-server S_f proves that the product ciphertext ω_j^f is re-encrypted from the product ciphertext ω_j^{f-1}. This can be done using the "equal discrete log" NIZKP since the server S_f knows all the re-encryption random numbers to form ciphertexts ω_j^f.

$d^f_{1,1}$,	$d^f_{1,2}$,	$d^f_{1,3}$,	...,	$d^f_{1,n}$	$\prod_{j=1}^{n} d^f_{1,j} = \mu_1^f$
$d^f_{2,1}$,	$d^f_{2,2}$,	$d^f_{2,3}$,	...,	$d^f_{2,n}$	$\prod_{j=1}^{n} d^f_{2,j} = \mu_2^f$
		⋮			⋮
$d^f_{m,1}$,	$d^f_{m,2}$,	$d^f_{m,3}$,	...,	$d^f_{m,n}$	$\prod_{j=1}^{n} d^f_{m,j} = \mu_m^f$
$\prod_{i=1}^{m} d^f_{i,1} = \omega_1^f$	$\prod_{i=1}^{m} d^f_{i,2} = \omega_2^f$	$\prod_{i=1}^{m} d^f_{i,3} = \omega_3^f$	⋯	$\prod_{i=1}^{m} d^f_{i,n} = \omega_n^f$	

Figure 1. The published output ciphertexts of the server S_f and all homomorphically computed values

The above protocol prevents each mix-server from sabotaging a ciphertext vector or replacing any components of a ciphertext vector received from previous mix-server. One important pre-requisite for this protocol to work is either keeping all plaintexts random and unknown to each mix-server or keeping the plaintexts specifically restricted, e.g. in our voting protocol the plaintexts allowed are z and 1 only. To avoid imposing the above constraints on the plaintext, another type of NIZKP based on graph isomorphism ZKP [1] can be modified to replace step 2. Due to the space limitation, we leave the security proofs and several other variations of this vector mix-net to the full paper.

2.4 The 'match' sub-protocol[14]:

A 'match' sub-protocol takes a ciphertext v and a set of ciphertexts $U = \{u_1, u_2, ..., u_n\}$ to determine the index i* such that $D_K(v) = D_K(u_{i^*})$ without revealing $D_K(v)$ or $\{D_K(u_i)\}_{i=1,...,n}$ or

the relations between $D_K(v)$ and arbitrary $D_K(u_i)$. This sub-protocol will be used in both the 'conversion of vote-count' and the 'winner-determination' stages.

Step 1. For each i, divide u_i by v.

Step 2. For each i, raise (u_i/v) to a secret random power λ sequentially by all participants. Every participant chooses a secret random number λ_k , calculates and publishes $(u_i/v)^{\lambda_k}$. Each participant is required to provide a "knowing discrete logarithm" NIZKP[19] for the correctness of his operation. In this way, no participant knows the overall random exponent $\lambda = \lambda_1 + \lambda_2 + ... + \lambda_n$.

Step 3. For each i, perform jointly a threshold decryption on $(u_i/v)^{\lambda_k}$. If the result is unity for a particular i*, then u_{i*} equals v. Output i* as the result.

3. Secure Voting Protocol

The secure voting protocol is divided into initialization, voting, counting, vote-count conversion, and winner-determination stages. These stages are presented as follows:

3.1 Initialization stage

Enroll, choose the secret, and decide the public key: Each participating voter is required to present a valid certificate signed by a specific certificate authority (CA) at this stage in order to register at the vote. Each voter is then given a one-time password for the electronic bulletin board. This allows a voter to write in his specific field exactly once.

Step 1. A voter V_i chooses a secret x_i, computes g^{x_i}, and publishes g^{x_i}.

Step 2. Each voter computes the public key $K \equiv g^X \equiv g^{x_1+x_2+.....+x_n} \equiv g^{x_1} \cdot g^{x_2} \cdot ... \cdot g^{x_n} \pmod p$.

Share the secret key:

Step 3. V_i chooses privately a degree t-1 polynomial $f_i(\theta) = x_i + \sum_{j=1}^{t-1} a_{ij}\theta^j$ which hides the secret x_i as the constant term, and publishes the exponentials $g^{a_{ij}}$ of all coefficients.

Step 4. V_i sends $f_i(h)$, which is the h-th share of x_i, to the voter V_h.

Step 5. V_i cross verifies all the shares he received, $\{s_{hi}\}_{h=1,...,n}$, against published values at step 3. through the equation $g^{s_{hi}} \equiv g^{f_h(i)} \equiv (g^{x_n})(g^{a_{b_1}})^i (g^{a_{b_2}})^{i^2} \cdots (g^{a_{b_{t-1}}})^{i^{t-1}}$

Step 6. V_i calculates from $\{s_{hi}\}_{h=1,...,n}$ the share $F(i) = \sum_{h=1}^{n} f_h(i) = \sum_{h=1}^{n} s_{hi}$ of the actual decryption key X=F(0).

Prepare ciphertext conversion vector sets:

Let S be a set consisting of n+1 n-dimensional ciphertext vectors: $\{(E_K(1), E_K(1), ..., E_K(1)), (E_K(z), E_K(1), ..., E_K(1)), ..., (E_K(z), E_K(z), ..., E_K(z), E_K(1)), (E_K(z), E_K(z), ..., E_K(z))\}$ where $E_K(1)$ and $E_K(z)$ are two common ciphertexts with their encryption random numbers

known to everyone. Each ciphertext vector is the unary representation corresponding to an integer in the set {0, 1, ..., n}. Each voter performs as a mix-server in the vector mix-net described in Section 2.3. The above set S of ciphertext vectors is fed n times into the vector mix-net to create n independently permuted vector set $S_1^{(enc)}$, ..., $S_n^{(enc)}$. All voters are assured that each permuted set $S_k^{(enc)}$ contains all ciphertexts re-encrypted and permuted from S. No voter can determine the correspondence between elements of $S_k^{(enc)}$ and S. As shown in figure 1 above, the product of ciphertext components of the i-th row is denoted as μ_i^f. Every ciphertext set $S_k^{(enc)}$ will be used only once for converting the representation of an accumulated vote-count ciphertext through the 'match' sub-protocol later.

3.2 Voting stage

Prepare the ballot:

Step 1. A voter V_i decides 'yes'/'no' for each candidate. He prepares a ballot C_i = ($E_K(z)$, $E_K(1)$, $E_K(z)$, ..., $E_K(1)$). An $E_K(z)$ represents a 'yes' vote and an $E_K(1)$ represents a 'no' vote. There are w_i 'yes' vote in the ballot C_i where w_i is no more than a constant L.

Prove the validity of a ballot: There are several illegal cases the protocol would like to avoid. First, a ballot contains an element $E_K(z^k)$. It is effectively k 'yes' votes to the same candidate. Second, a ballot contains more than L 'yes'-votes. Third, a ballot contains illegal encrypted elements other than $E_K(z)$ or $E_K(1)$ that would disrupt the homomorphic counting procedure. Fourth, a ballot is copied and re-encrypted from another ballot.

Step 2. A voter proves that each ciphertext element of the ballot is the encryption of either z or 1 with the '1-out-of-L' NIZKP described in section 2.2. Note that in this proof, a voter needs to know the encryption random number for each ciphertext. Therefore, this prevents a dishonest voting center from copying and re-encrypting some other voter's ballot.

Step 3. A voter also proves that $\chi_i \equiv \prod_{j=1}^{m} c_{ij} \pmod p$ is the encryption of an element in the set {1, z^1, z^2, ..., z^L} with the '1-out-of-L' NIZKP described in section 2.2.

Publish the ballot:

Step 4. A voter logs on the electronic bulletin board with the one-time password received earlier.

Step 5. A voter publishes the ballot and the NIZKPs on his designated fields as shown in Figure 2.

Candidates / Voters	1	2	3	...	m-2	m-1	m	
V_1	c_{11}	c_{12}	c_{13}	...	$c_{1(m-2)}$	$c_{1(m-1)}$	c_{1m}	NIZKP for $\{c_{1j}\}$ NIZKP of χ_1

V_2	c_{21}	c_{22}	c_{23}	…	$c_{2(m-2)}$	$c_{2(m-1)}$	c_{2m}	NIZKP for $\{c_{2j}\}$ NIZKP of χ_2
…	…	…	…		…	…	…	…
V_{n-1}	$c_{(n-1)1}$	$c_{(n-1)2}$	$c_{(n-1)3}$	…	$c_{(n-1)(m-2)}$	$c_{(n-1)(m-1)}$	$c_{(n-1)m}$	NIZKP for $\{c_{(n-1)j}\}$ NIZKP of χ_{n-1}
V_n	c_{n1}	c_{n2}	c_{n3}	…	$c_{n(m-2)}$	$c_{n(m-1)}$	c_{nm}	NIZKP for $\{c_{nj}\}$ NIZKP of χ_n
Tallies	τ_1	τ_2	τ_3		τ_{m-2}	τ_{m-1}	τ_m	

Figure 2. Ballots published on the electronic bulletin board

Verify the validity of NIZKPs:

Step 6. In order to prevent malicious subversion in the remaining protocol, the voting center has the responsibility to verify the NIZKPs corresponding to each encrypted vote vectors.

Step 7. Because none of the published c_{ij} will be decrypted in the protocol, any suspicious participant can also verify these proofs to establish his confidence on the ongoing protocol.

3.3 Counting and tally-conversion stage

Tally the votes homomorphically:

As shown in Figure 2, the ciphertexts of the j-th column on the electronic bulletin board are multiplied together as the encrypted tally of the j-th candidate, i.e. $\tau_j = \prod_{i=1}^{n} c_{ij}$.

Convert each candidate's vote-count to an encrypted unary vector:

In this stage, each voter jointly converts τ_j into its unary ciphertext vector representation through the 'match' sub-protocol without revealing the corresponding vote-count. First, the encrypted tally τ_j is matched against the set $S_j^{(enc)}$, which is prepared in the initialization stage. During the 'match' sub-protocol, all voters jointly compute random exponentials to blind the conversion process and jointly decrypt the result to find the matched item. The converted unary ciphertext vector $(\sigma_{1j}, \sigma_{2j}, ..., \sigma_{nj})$ for the j-th candidate is then published on the bulletin board as a column vector as shown in Figure 3.

Candidates / # Votes	τ_1	...	τ_j	...	τ_{m-1}	τ_m	
1	σ_{11}	...	σ_{1j}	...	$\sigma_{1(m-1)}$	σ_{1m}	ρ_1
⋮	⋮		⋮		⋮	⋮	⋮
i	σ_{i1}	...	σ_{ij}	...	$\sigma_{i(m-1)}$	σ_{im}	ρ_i
⋮	⋮		⋮		⋮	⋮	⋮
n	σ_{n1}	...	σ_{nj}	...	$\sigma_{n(m-1)}$	σ_{nm}	ρ_n

Figure 3. Unary ciphertext vector representations of vote-counts

Determine R winners:

As shown in Figure 3, ρ_i can be calculated as the product of a row of ciphertexts homomorphically, i.e. $\rho_i \equiv \prod_{j=1}^{m} \sigma_{ij} \pmod p$. The plaintexts corresponding to $(\rho_1, \rho_2, ..., \rho_n)$ must be decreasing exponentials of z, ex. $(z^m, z^m, z^{m-1}, z^{m-1}, ..., z^3, z^3, z^3, z^3, z^2, z^2, z^2, z^2, z^1, z^1, z^0, z^0, z^0)$. In this example, ρ_{n-5} being z^2 means that there are two candidates whose votes are no less than n-5. In order to find the R winners without decrypting all candidates' votes, the ciphertext $E_K(z^R)$ is matched against the vector $(\rho_1, \rho_2, ..., \rho_n)$ with the 'match' sub-protocol. After the match, if ρ_{i^*} is the first entry that decrypts to z^R, $\{\sigma_{i^* j}\}_{j=1, ..., m}$ in the i*-th row of Figure 3 are decrypted. If $D_K(\sigma_{i^* j^*})$ is z, then the j*-th candidate is declared a winner. There are R winners with vote-counts no less than i*. Note that the joint decryptions in the 'match' sub-protocol on $\{ (\rho_i / E_K(z^R))^{\lambda} \}_{i=1, ..., n}$ should be performed in increasing order of i. The decryption should stop at the i*-th element so that no voter can know the right boundary of the sequence of $E_K(z^R)$ in $(\rho_1, \rho_2, ..., \rho_n)$.

4. Security Analysis and Discussion

Consider the following cases where a malicious mix-server might try to disrupt the vector mix protocol of Section 2.3 when it is used to prepare $S_1^{(enc)}, ..., S_n^{(enc)}$ in Section 3.1.

Case 1: A server S_f removes a particular input row $(d^{f-1}_{i,1}, d^{f-1}_{i,2}, ..., d^{f-1}_{i,n-1}, d^{f-1}_{i,n})$ and replaces it with a row vector re-encrypted from another row $(d^{f-1}_{i^*,1}, d^{f-1}_{i^*,2}, ..., d^{f-1}_{i^*,n-1}, d^{f-1}_{i^*,n})$. This behavior is thwarted by the n re-encryption NIZKPs on the pairs $\{(\omega_j^{f-1}, \omega_j^f)\}_{j=1, ...,n}$ since there is at least one different element between the i-th row and the i*-th row in our protocol.

Case 2: A server S_f permutes each column independently. This will be detected by the m '1-out-of-L re-encryption' NIZKPs from every element in the set $\{\mu^{f-1}_{1,j}, \mu^{f-1}_{2,j}, ..., \mu^{f-1}_{m,j}\}$ to the set $\{\mu^{f}_{1,j}, \mu^{f}_{2,j}, ..., \mu^{f}_{m,j}\}$ provided that independent permutations would cause different row products. The next case will discuss some special malicious operations that do not satisfy the above condition.

Case 3: A server S_f chooses two rows i_1 and i_2, two columns j_1 and j_2, multiplies $E(z^{-1})$ to $d_{i_1j_1}$ and $d_{i_2j_2}$ and multiplies $E(z)$ to $d_{i_2j_1}$ and $d_{i_1j_2}$. In this way, the row products $\mu^{f}_{i_1}$ and $\mu^{f}_{i_2}$ remain unchanged, the column products $\omega^{f}_{j_1}$ and $\omega^{f}_{j_2}$ also remain unchanged. For example:

	i_1	i_2
j_1	$E_K(1)$	$E_K(1)$
j_2	$E_K(1)$	$E_K(1)$

$\Rightarrow$

	i_1	i_2
j_1	$E_K(z^{-1})$	$E_K(z)$
j_2	$E_K(z)$	$E_K(z^{-1})$

However, the modified element $E_K(z^{-1})$ is not a valid representation, and will be detected by the m "1-out-of-L re-encryption" NIZKPs from every element in the set $\{d^{f}_{1,j}, d^{f}_{2,j}, ..., d^{f}_{m,j}\}$ to the set $\{d^{f-1}_{1,j}, d^{f-1}_{2,j}, ..., d^{f-1}_{m,j}\}$. In the following table, we list the other eight possible cases for the above example. Note that each output row has at least one invalid element, which is either $E_K(z^{-1})$ or $E_K(z^2)$.

$d^{f-1}_{i_1j_1}$	$d^{f-1}_{i_1j_2}$	$d^{f-1}_{i_2j_1}$	$d^{f-1}_{i_2j_2}$		$d^{f}_{i_1j_1}$	$d^{f}_{i_1j_2}$	$d^{f}_{i_2j_1}$	$d^{f}_{i_2j_2}$
$E_K(1)$	$E_K(1)$	$E_K(z)$	$E_K(1)$	$\Rightarrow$	**$E_K(z^{-1})$**	$E_K(z)$	**$E_K(z^2)$**	**$E_K(z^{-1})$**
$E_K(1)$	$E_K(1)$	$E_K(z)$	$E_K(z)$	$\Rightarrow$	**$E_K(z^{-1})$**	$E_K(z)$	**$E_K(z^2)$**	$E_K(1)$
$E_K(z)$	$E_K(1)$	$E_K(1)$	$E_K(1)$	$\Rightarrow$	$E_K(1)$	$E_K(z)$	$E_K(z)$	**$E_K(z^{-1})$**
$E_K(z)$	$E_K(1)$	$E_K(z)$	$E_K(1)$	$\Rightarrow$	$E_K(1)$	$E_K(z)$	**$E_K(z^2)$**	**$E_K(z^{-1})$**
$E_K(z)$	$E_K(1)$	$E_K(z)$	$E_K(z)$	$\Rightarrow$	$E_K(1)$	$E_K(z)$	**$E_K(z^2)$**	$E_K(1)$
$E_K(z)$	$E_K(z)$	$E_K(1)$	$E_K(1)$	$\Rightarrow$	$E_K(1)$	**$E_K(z^2)$**	$E_K(z)$	**$E_K(z^{-1})$**
$E_K(z)$	$E_K(z)$	$E_K(z)$	$E_K(1)$	$\Rightarrow$	$E_K(1)$	**$E_K(z^2)$**	**$E_K(z^2)$**	**$E_K(z^{-1})$**
$E_K(z)$	$E_K(z)$	$E_K(z)$	$E_K(z)$	$\Rightarrow$	$E_K(1)$	**$E_K(z^2)$**	**$E_K(z^2)$**	$E_K(1)$

Case 3a: Similar to case 3, a server S_f chooses two rows i_1 and i_2, two columns j_1 and j_2, swaps two pairs of ciphertexts, i.e. $\{d^{f-1}_{i_1j_1}, d^{f-1}_{i_1j_2}\}$ and $\{d^{f-1}_{i_2j_1}, d^{f-1}_{i_2j_2}\}$. For certain situations, the row products $\mu^{f}_{i_1}$ and $\mu^{f}_{i_2}$, the column products $\omega^{f}_{j_1}$ and $\omega^{f}_{j_2}$ preserve their values under the swap operation. For instance,

	i_1	i_2
j_1	$E_K(z)$	$E_K(1)$
j_2	**$E_K(1)$**	**$E_K(z)$**

$\Rightarrow$

	i_1	i_2
j_1	**$E_K(1)$**	**$E_K(z)$**
j_2	$E_K(z)$	$E_K(1)$

However, the input $E_K(1)$, ..., $E_K(z)$ is not a valid representation in our protocol.

Because the ElGamal encryption system is used, the overall security of the proposed voting protocol is built upon the common intractability assumptions, namely, the DDH

assumption. Because the Fiat-Shamir heuristic[8] of turning a ZKP into an NIZKP is used throughout the protocol, we also rely on the random oracle model assumption. Besides, we assume a broadcast channel in the form of a bulletin board.

At the key sharing stage, the correctness of the shared key can be verified as in Gennaro's framework[12]. In the protocol, it requires at least t corrupted voters to perform an unauthorized decryption. The encrypted ballot is guaranteed nonmalleable through the "1-out-of-L" NIZKP in Section 2.2. This thwarts the phantom votes. All homomorphic computations in the counting and winner-determination stages are publicly repeatable. Every voter can verify them to ensure his vote is properly counted. The vector mix-net introduced is completely public verifiable.

At the vote-count conversion stage, the joint decryption in the 'match' sub-protocol does not reveal the actual vote-count because of the previous 'mix' operation on the conversion sets $S_i^{(enc)}$. The relations between a vote-count and other members in the conversion set $S_i^{(enc)}$ are hided by the random exponent λ provided jointly by all participants. Each random conversion set $S_i^{(enc)}$ is used only once to eliminate the chance of collision, i.e. the vote-counts of two candidates equal and match to the same element of a set $S_i^{(enc)}$.

At the winner-determination stage, the homomorphic multiplication is publicly repeatable. The 'match' operation of $E_K(z^R)$ and the decryption of ρ_i till the first appeared '1' ($=\rho_{i*}$) reveals only the fact that there are R candidates having votes no less than the constant i*. One special problem to be noticed is the case that the R-th and (R+1)-th candidates have equal votes. In this case, $E_K(z^R)$ would match to nothing in the list $\{ \rho_i \}_{i=1, ..., n}$. What happened is that ..., z^{R+1}, z^{R-1}, ... is a partial sequence in the list $\{ \rho_i \}_{i=1, ..., n}$. This problem can be solved by matching $E_K(z^{R+1})$ or $E_K(z^{R-1})$ against the list $\{ \rho_i \}_{i=1, ..., n}$ depending on pre-established election rules about the equal-vote situations. This procedure can be easily generalized when three or more candidates have equal votes

Along the execution of this voting protocol, all uncooperative participants can be identified through the NIZKP at the spot and banned from further participation. Therefore, this protocol is robust against active adversaries. To further investigate the security features of the entire protocol, it is suggested[11] to evaluate this protocol in the universal composability framework.

In a small-scale voting scheme as addressed in this paper, the amount of computation and communication is not that critical as a large-scale general e-voting scheme. It is the fulfillment of the newly established privacy requirement that justifies the usage of an electronic voting scheme in such a small-scale critical election. The fairness benefits obtained soon pay for the costs of deployment and the operational inconvenience.

The proposed secure winner-determination procedure can be used in many voting schemes with variations, e.g. a ranking-choice voting or an allocative-choice voting. To accommodate the current scheme to both types of voting schemes, the primary part to be modified is the ballot preparation and the corresponding NIZKPs provided by a voter. The remaining tallying, vote-count conversion, and winner-determination stages work in its current form.

In an allocative-choice voting scheme, each voter gives m candidates separate satisfaction scores in the range 0-100. It is required that these m scores sum up to 100. In the ballot preparation stage, each voter V_i prepares a ballot $C_i = (E_K(z^{a_1}), E_K(z^{a_2}), E_K(z^{a_3}), ..., E_K(z^{a_m}))$ where a_j is an integer lies within 0 and 100 and satisfies $\sum_{j=1}^{m} a_j = 100$. The voter V_i is required to submit m '1-out-of-L' NIZKPs to prove that each ciphertext $E_K(z^{a_j})$ decrypts to one element of the set $\{ z^0, z^1, ..., z^{100} \}$. The voter also needs to prove that the plaintext of the homomorphically calculated ciphertext $E_K(z^{a_1 + a_2 + ... + a_m})$ is z^{100} by an 'equal discrete log' NIZKP.

In a ranking-choice voting scheme, each voter gives m candidates a ranking from 1 to m. It is illegal to give two candidates the same rank. Again in the ballot preparation stage, each voter V_i prepares a ballot $C_i = (E_K(z^{a_1}), E_K(z^{a_2}), E_K(z^{a_3}), ..., E_K(z^{a_m}))$ where $\{a_1, a_2, ..., a_m\}$ is a permutation of the set $\{1, 2, ..., m\}$. The voter V_i is required to submit m '1-out-of-L re-encryption' NIZKPs to prove, for each j from 1 to m, that the ciphertext $E_K(z^j)$ is re-encrypted from one element of the set $\{ E_K(z^{a_1}), E_K(z^{a_2}), E_K(z^{a_3}), ..., E_K(z^{a_m}) \}$.

5. Concluding Remarks

In this paper, we identified an important requirement on a small-scale electronic voting scheme, namely, to keep all vote-counts secret and the voting process publicly verifiable. There are a bunch of application scenarios that require electronic voting schemes to eliminate trusted third parties in the election process. However, the voting schemes proposed in the past focused mainly on problems associated with a large-scale generic election and are not applicable to the scenario we noticed. We constructed our voting scheme based on popular homomorphic schemes and focused on the design of the winner-determination procedure to keep all vote-counts secret. This task is a specialization of a general multi-party protocol[10]. We achieved the goal using a ciphertext format conversion procedure, which was based on the 'mix-and-match' sub-protocol and many NIZKPs. In the future, we are looking for some light-weight version of protocols that can fulfill the same requirements identified here.

6. References

[1] M. Abe, “Universally verifiable MIX with verification work independent of the number of MIX servers”, Advanced in Cryptology – Eurocrypt 1998, LNCS 1403, 1998.
[2] M. Abe and K. Suzuki, “M+1-st Price Auction Using Homomorphic Encryption”, PKC 2002, LNCS 2274, pp. 115-124, 2002.
[3] O. Baudron, P. Fouque, D. Pointcheval, J. Stern, and G. Poupard, “Practical Multi-Candidate Election System”, ACM 20-th Symposium on Principle of Distributed Computing, PODC’01, 2001.
[4] J. C. Benaloh, “Verifiable Secret Ballot Elections”, Ph.D. thesis, Yale University, 1987.
[5] D. Chaum and T. Pedersen, “Wallet Databases with Observers”, Advanced in Cryptology – Crypto 1992, pp. 89-105, 1992.
[6] R. Cramer, R. Gennaro, and B. Schoenmakers, “A Secure and Optimally Efficient Multi-Authority Election Scheme”, Advanced in Cryptology – Eurocrypt 1997, LNCS 1233, pp. 119-136, 1997.
[7] T. ElGamal, “A Public-key Cryptosystem and Signature Scheme Based on Discrete Logarithms”, IEEE Trans. on Information Theory, Vol. IT-31, pp. 469-472, 1985.
[8] A. Fiat and A. Shamir, “How To Prove Yourself: Practical Solutions to Identification and Signature Problems”, Advances in Cryptology: Proc. Crypto’86, pp.186-194, 1986.
[9] A. Fujioka, T. Okamoto, and K. Ohta, “A practical secret voting scheme for large scale elections”, Advanced in Cryptology – AUSCRYPT’92, 1992.
[10] O. Goldreich, S. Micali, and A. Wigderson, “How to Play Any Mental Game”, ACM STOC’87, 1987
[11] J. Groth, “Evaluating Security of Voting Schemes in the Universal Composability Frmework”, Proc. ACNS’04, 2004.
[12] R. Gennaro et al. “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”, Advances in Cryptology: Proc. Eurocrypt’99, pp. 293-310, 1999.
[13] M. Hirt and K. Sako, “Efficient receipt-free voting based on homomorphic encryption”, Advanced in Cryptology – Eurocrypt’00, 2000.
[14] M. Jakobsson and A. Juels, “Mix and Match: Secure Function Evaluation via Ciphertexts”, Advanced in Cryptology – Asiacrypt’00, pp. 162-177, 2000.
[15] M. Jakobsson and A. Juels, “Millimix: Mixing in Small Batches”, Tech. Rep. 99-33, DIMACS, 1999.
[16] T. P. Pedersen, “A Threshold Cryptosystem without a Trusted Party”, Advanced in Cryptology – Eurocrypt 1991, pp. 522-526, 1991.
[17] M. J. Radwin, “An untraceable, universally verifiable voting scheme”, 1995, available at http://www.radwin.org/michael/projects/voting.html.
[18] R. Rivest, “Electronic Voting”, Financial Cryptography’91, 1991.
[19] Z. Rjaskova, “Electronic Voting Schemes”, Ms Thesis, Comenius University, Bratislava, 2002.
[20] P.-Y. Ting, Y.-T. Lee, C.-Y. Chen “On The Public Verifiability of an M+1-st Price Auction Using Homomorphic Encryption”, Technical Report, National Taiwan Ocean University, Dec. 2003.