An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling

Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, Kazue Sako

Internet Systems Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae Kawasaki 216-8555 Japan
j-furukawa@ay.jp.nec.com, ke-mori@bx.jp.nec.com, obana@bx.jp.nec.com,
h-miyauchi@bc.jp.nec.com, k-sako@ab.jp.nec.com

Abstract. This paper discusses the implementation of the voting scheme based on mix-net technology. The advantages of employing this technology are that voters can vote-and-go, and that it is flexible enough to be used for variety of vote-expression methods, while ensuring the privacy of votes and the elimination of faulty players. The most attractive security feature of this scheme is its universal verifiability; anyone can confirm the correctness of the result.
Such verifiability is achieved by providing proofs on correct shuffling and decryption. The paper presents a new scheme for generating a single proof for shuffle-and-decrypt process. Compared to the combination of two separate proofs on shuffle and decryption, the new scheme is 150% faster with only 80% of the length. As a result, the system was able to produce results that were verified correct within *twenty* minutes following a vote participated in by ten thousand voters, with three shuffling centers being used.
We believe this is the first implementation report of a voting scheme with universal verifiability.

Key words: voting, election, privacy, shuffle, zero-knowledge proof, mix-net

1 Introduction

1.1 Voting schemes

The three basic requirements on a secure voting system are detection of faulty voters; detection of faulty centers; and vote secrecy. The result of research in cryptographic protocols gave us several variations of voting protocols achieving the above requirements, which can be grouped under the following three categories:

- Blind signature based schemes:[FOO92,Sak94,OMAFO]
- Homomorphic encryption based schemes:[CY85,SK94,CFSY96,CGS97,DJ01]
- Shuffling based schemes:[PIK93,SK95,Abe99,FS01,Ne01]

Blind signature based protocol is the most computationally efficient one, and have been implemented in [SENSUS,EVOX]. However, this scheme has only individual verifiability; a voter can confirm that his own vote is accepted by the authority, but no outside 'observer' can verify the whole procedure. Secondly, this schemes requires a channel that is untraceable, which is not easy to achieve in a real life. Furthermore, the voter must engage in all phases of voting, and the existence of any voter who fails in completing a process threats a sound execution of the voting result.

Homomorphic encryption based schemes is an elegant and efficient scheme when used for yes/no-vote case. However, they suffer largely from inflexibility in representing a vote. The extension the scheme to 1-out-of L voting will sacrifice the efficiency with large L and large number of voters,and the size of L that is allowable is unknown.

On the other hand, the shuffling based schemes, also known as mix-net, bears many desirable features; it enjoys flexibility in representing any vote, the voters can simply vote-and-go and requires only a small computational ability. Moreover, by having authorities prove the correctness of their procedures, it achieves public verifiability. With this capability, we can employ 'observers', third parties to ensure the correctness of procedures. In a paper-based voting scheme, the existence of the observers is often required by law.

The cost is paid for this property. Proofs for assuring correctness of shuffling and decryption are not for free. Since proving the correctness of shuffling is more expensive than that of decryption, many schemes were proposed to improve efficiency[SK95,Abe99,FS01,Ne01]. Most of the efficiency arguments made in these documents were based on theoretical evaluations, like number of modular exponentiations. To the authors' knowledge, no voting system which uses these scheme was reported, which indicates the scheme is yet to be believed practical.

1.2 Our contribution

We report here the first implementation of universally verifiable voting scheme based on shuffling. We present a new scheme to prove correctness of shuffle-and-decrypt process. It is, in a way, a merge of an efficient proof on shuffling proposed in [FS01] and an efficient proof on decryption using the techniques from [Br93]. Experimental results show that it is 150% faster than performing each proof separately, and a proof is 20% shorter. The proposed scheme can be proved to be correct and sound.

However, we cannot prove any zero-knowledgeness of the resulted scheme. Moreover, we found that the protocol of [FS01] which we based our protocol does not satisfy the definition of computational zero-knowledge. On the other hand, we considered security requirements regarding which information must be kept hidden in the protocol. As a result, we can prove that no polynomially bounded adversary can compute any partial information of the permutation from the protocol.

Using the newly proposed scheme, we have implemented a voting system which is publicly verifiable. Our system can output a certified tally for 10,000

voters within 20 minutes, with three shuffling centers, each on a personal computer being used.

We note that receipt-freeness is not the aim of our system, although we are aware of potential threats of vote buying and coercing due to lack of the receipt-free property.

1.3 Organization of the Paper

In Section 2, we describe a new scheme to prove correctness of shuffle-and-decrypt procedure. In Section 3 we provide implementation details of our voting system. We also provide experimental results comparing different proof scheme we have implemented.

2 The New Scheme for Proving Shuffle-and-Decrypt Procedure

In this section, we provide details of the new scheme. First, we briefly describe the shuffling based voting scheme, and discuss why we need shuffle-and-decrypt proofs.

2.1 An Overview of the Shuffling Based Voting

The shuffling based schemes, also known as mix-net, achieve anonymity of votes by distributing the decryption key among multiple authorities called shuffling centers. Voters send signed but encrypted votes. These votes will be processed by the shuffling centers, who shuffles the votes and decrypt them. After all the shuffling centers have done their work, the encrypted vote will be completely decrypted, but its original owner can not be identified due to the shuffles. By providing shuffle-and-decrypt proofs, anyone can verify the output is a result of correct decryptions of the shuffled valid votes. The proofs should be made in a way it will not reveal the shuffle nor the decryption key, so it would not infringe the vote anonymity.

2.2 How votes are generated and processed

The votes are encrypted using ElGamal cryptosystems[E85], with public keys (p, q, g, Y) and a secret key $\bar{x} \in \mathbf{Z_q}$ s.t. $Y = g^{\bar{x}} \bmod p$. Here, p and q are two primes s.t. $p = kq + 1$, where k is an integer, and g is an element that generates a subgroup $\mathbf{G_q}$ of order q in $\mathbf{Z}/p\mathbf{Z}$.

The vote m_v is encrypted to a ciphertext (G, M) where $G = g^{\bar{r}} \bmod p$ and $M = m_v \cdot Y^{\bar{r}}$, and $\bar{r}$ is a random element in $\mathbf{Z}/q\mathbf{Z}$, chosen by the voter.

Such ciphertext can be decrypted by a person knowing the secret key $\bar{x}$. The vote is recovered by: $M/G^{\bar{x}} \bmod p$. However, in the voting scheme no single entity knows $\bar{x}$. The secret key $\bar{x}$ is distributed among m shuffling centers. Each shuffling center SC_i has a secret key $x_i \in \mathbf{Z}/q\mathbf{Z}$ and corresponding public key $y_i = g^{x_i}$.

The sum of their secret keys gives $\bar{x}$. Given a ciphertext $(G, M) = (G^{(1)}, M^{(1)})$, when all the shuffling centers computes $(G^{(i+1)}, M^{(i+1)}) = (G^{(i)}, M^{(i)}/(G^{(i)})^{x_i} \bmod p)$ sequentially, then the second component of the output from the last shuffling center gives the decrypted result,$m_v = M/G^{\bar{x}} \bmod p$.

A ciphertext (G, M) can be randomly transformed to another ciphertext which decrypts to a same message. Using a randomly generated $s \in \mathbf{Z}/q\mathbf{Z}$, such transformed ciphertext, $(G \cdot g^s, M \cdot Y^s)$, is unlinkable to original ciphertext (G, M). This will be useful for shuffling ciphertext, which will be described in the next subsection.

2.3 Shuffle-and-Decrypt

For multiple votes, each shuffling center shuffles the input votes before decrypting each of them by his own secret key. We call this procedure 'shuffle-and-decrypt'. For simplicity, we concentrate on one shuffling center and denote his secret key as x. We represent by $\bar{y}$ the product of the public keys of subsequent centers.

Given n ciphertexts $\{(G_i, M_i)\}$, where all $\{G_i\}$ and $\{M_i\}$ have the order q, the shuffling center randomly chooses a permutation π and a random element $s_i \in_U \mathbf{Z}/q\mathbf{Z}$ to obtain shuffle-and-decrypt result as follows:

$$(G'_i, M'_i) = \left(g^{s_i} G_{\pi(i)}, \bar{y}^{s_i} M_{\pi(i)} / G_i'^{x}\right) \bmod p (i = 1, \ldots, n)$$

Details of this procedure is given in Subsection 3.2.

2.4 Generation of the proof

We now provide the new scheme to generate a proof that the shuffling center (which will be denoted as the prover in the sequel) indeed shuffled and decrypted honestly. The proof adds proof of decryption on the top of proof of shuffle proposed in [FS01]. The newly added proving procedures are the equations labeled (2) to (5).

We describe the scheme in a non-interactive way, where a challenge from a verifier is given as an output of some universal one-way hash functions. We assume here the order of elements of input ciphertexts (G_i, M_i) and output ciphertexts (G'_i, M'_i) are all q. [1]

To prove (G'_i, M'_i) are generated correctly from (G_i, M_i), the prover computes the following equations for randomly chosen $z, z_i, \rho, \sigma, \tau$, λ and $\lambda_i, z' \in_U \mathbf{Z}/q\mathbf{Z}$ $(i = 1, \ldots, n)$: We use $\mathcal{H}$ and $\tilde{\mathcal{H}}$ to denote universal one-way hash functions which output an element of $\mathbf{Z}/q\mathbf{Z}$ and $\mathbf{G_q}$, respectively.

[1] We note here that it is important for both shuffling centers and verifiers to check that the order are indeed identically q. If one of the input ciphertexts are not of order q, then the resulting output may reveal the permutation used in the protocol. Or, the shuffling center may intentionally present wrong ordered ciphertexts which pass the verification formula but is not a correct shuffle.

$$
\begin{aligned}
&\tilde{g} = \tilde{\mathcal{H}}(p,q,g,Y,0), \quad \tilde{g}_i = \tilde{\mathcal{H}}(p,q,g,Y,i) \\
&v = g^{\rho}, \quad w = g^{\sigma}, \quad t = g^{\tau}, \quad u = g^{\lambda}, \quad u_i = g^{\lambda_i} \bmod p \\
&\tilde{g}'_i = \tilde{g}^{s_i}\tilde{g}_{\pi(i)}, \qquad \tilde{g}' = \tilde{g}^{z}\prod_{j=1}^{n}\tilde{g}_j{}^{z_j} \bmod p \\
&g' = g^{z}\prod_{j=1}^{n}G_j{}^{z_j}, \qquad m' = \bar{y}^{z}\prod_{j=1}^{n}M_j{}^{z_j} \bmod p \\
&\dot{t}_i = g^{3z_{\pi(i)}+\tau\lambda_i}, \qquad \dot{v}_i = g^{3z_{\pi(i)}{}^2+\rho s_i} \bmod p \\
&\dot{v} = g^{\sum_{j=1}^{n} z_j{}^3+\tau\lambda+\rho z} \bmod p \\
&\dot{w}_i = g^{2z_{\pi(i)}+\sigma s_i}, \qquad \dot{w} = g^{\sum_{j=1}^{n} z_j{}^2+\sigma z} \bmod p \\
&c_i = \mathcal{H}(p,q,g,\bar{y},\tilde{g},\{\tilde{g}_j\},\{(G_j,M_j)\},\{(G'_j,M'_j)\}, \\
&\qquad \tilde{g}',\{\tilde{g}'_j\},g',m',v,w,t,u,\{u_j\}, \\
&\qquad\quad \{\dot{t}_j\},\dot{v},\{\dot{v}_j\},\dot{w},\{\dot{w}_j\},i:(j=1,\ldots,n)) \qquad (1) \\
&r_i = c_{\pi^{-1}(i)} + z_i, \qquad r = \sum_{j=1}^{n} s_j c_j + z \bmod q \\
&\lambda' = \sum_{j=1}^{n}\lambda_j c_j{}^2 + \lambda \bmod q \\
&\zeta = \prod_{j=1}^{n} G'_j{}^{c_j}, \qquad \eta = \zeta^{x} \bmod p \qquad (2) \\
&y' = g^{z'}, \qquad \eta' = \zeta^{z'} \bmod p \qquad (3) \\
&c' = \mathcal{H}(p,q,g,y,\zeta,\eta,y',\eta') \qquad (4) \\
&r' = c'x + z' \bmod q \qquad (5)
\end{aligned}
$$

The prover send the proof $g', m', \tilde{g}', \tilde{g}'_i, v, w, t, u, u_i, \dot{t}_i, \dot{v}_i, \dot{v}, \dot{w}_i, \dot{w}, r, r_i, \lambda', \eta, \eta', y', r'$ $(i = 1, \ldots, n)$ to the verifier along with $\{(G'_i, M'_i)\}$.

2.5 Verifications of the proof

The verifier first computes $(c_i)_{(i=1,\ldots,n)}$ according to eq.(1). Next, the verifier compute

$$\zeta = \prod_{j=1}^{n} G'_j{}^{c_j} \bmod p$$

and generate c' according to eq.(4).

The verifier accepts the proof if all of the following equations hold.

$$v^q = t^q = w^q = 1 \bmod p$$

$$
\begin{aligned}
g^r \prod_{j=1}^{n} {G_j}^{r_j} &= g'\zeta \bmod p \\
\bar{y}^r \prod_{j=1}^{n} {M_j}^{r_j} &= \eta\, m' \prod_{j=1}^{n} {M'_j}^{c_j} \bmod p \\
\tilde{g}^r \prod_{j=1}^{n} {\tilde{g}_j}^{r_j} &= \tilde{g}' \prod_{j=1}^{n} {\tilde{g}'_j}^{c_j} \bmod p \\
g^{\lambda'} &= u \prod_{j=1}^{n} {u_j}^{c_j^2} \bmod p \\
t^{\lambda'} v^r g^{\sum_{j=1}^{n} ({r_j}^3 - {c_j}^3)} &= \dot{v} \prod_{j=1}^{n} {\dot{t}_j}^{{c_j}^2} \prod_{j=1}^{n} {\dot{v}_j}^{c_j} \bmod p \\
w^r g^{\sum_{j=1}^{n} ({r_j}^2 - {c_j}^2)} &= \dot{w} \prod_{j=1}^{n} {\dot{w}_j}^{c_j} \bmod p \\
g^{r'} = y^{c'} y' \ &, \ \zeta^{r'} = \eta^{c'} \eta' \bmod p.
\end{aligned}
$$

2.6 Structure of the Scheme

We briefly sketch the structure of our scheme. Our scheme is constructed by merging shuffle-proof proposed in [FS01] and decryption-proof. For the detail description of shuffle-proof, please refer to [FS01]. The decryption-proof can be constructed by well known technique to prove the equality of discrete logarithm as follows.

The prover wants to prove that M'_i is the valid decryption of $(G'_i, \bar{M}_i)$ with a private key x, that is, a relation $M'_i/\bar{M}_i = {G'_i}^x \bmod p$ holds for all i. The prover computes the followings:

$$
\begin{aligned}
c_i &= \mathcal{H}(p, q, g, y, G'_i, \bar{M}_i, M'_i : (i = 1, \ldots, n)) \\
\zeta &= \prod_{j=1}^{n} {G'_j}^{c_j} \ , \ \eta = \zeta^x \bmod p \\
y' &= g^{z'} \ , \ \eta' = \zeta^{z'} \bmod p \\
c' &= \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta') \\
r' &= c'x + z' \bmod q
\end{aligned}
$$

Then, the prover sends to the verifier, y', η', r' as a proof.

The verifier computes

$$
\begin{aligned}
c_i &= \mathcal{H}(p, q, g, y, G'_i, \bar{M}_i, M'_i : (i = 1, \ldots, n)) \\
\zeta &= \prod_{j=1}^{n} {G'_j}^{c_j} \ , \ \eta = \prod_{j=1}^{n} (M'_j/\bar{M}_j)^{c_j} \bmod p \\
c' &= \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta').
\end{aligned}
$$

Then, verifier accepts the proof if the following equations hold.

$$g^{r'} = y^{c'} y' \, , \zeta^{r'} = \eta^{c'} \eta' \bmod p.$$

We have noticed that some of the computations done in the shuffle-proof [FS01] and in the decryption-proof are similar. For example, even though c_i is generated in a slightly different way, $\zeta = \prod_{j=1}^{n} G_j'^{c_j} \bmod p$ is also computed in the shuffle-proof, $\eta = \prod_{j=1}^{n} (M_j'/\bar{M}_j)^{c_j} = \prod_{j=1}^{n} M_j'^{c_j} / \prod_{j=1}^{n} \bar{M}_j{}^{c_j} \bmod p$ in decryption-proof and $\prod_{j=1}^{n} \bar{M}_j{}^{c_j} \bmod p$ in shuffle-proof have same factor.

We merged the two protocols in a following way: Two proof share the same c_i so that both prover and verifier need to computes above mentioned value only once. As a result of merging, experimental results show that it is 150% faster than performing each proof separately, and a proof is 20% shorter.

2.7 Properties of the Scheme

The proposed scheme is complete. Assuming that the prover is computationally bounded, i.e. solving discrete logarithm problem is hard for the prover, the scheme is sound. We claim that the protocol does not reveal the permutation; if an adversary can compute any part of the permutation from the proof, then he can solve the decision Diffie-Hellman problem.

The shuffle-proof presented in [FS01] is claimed to be computational zero-knowledge. However it is not correct. [2] However, it does not straightforwardly mean that the protocol of [FS01] leaks some information. Therefore, we considered other means to ensure the security of the protocol. What we need to ensure is that the protocol does not leak any information regarding the permutation. Therefore we formalized the situation that the protocol does not leak such information and proved that the protocol of [FS01] satisfies the definition.

Definition : Let X be a set of input and output ciphertexts, $\Pi(X)$ be a set of corresponding valid permutations, $View_{P,V}$ be a view of a verifier, that is, a random tape of the verifier and messages the verifier receives from a prover. We say the protocol (P, V) does not leak any information about the permutation if and only if:
for every polynomially bounded adversary E, there exists a polynomially bounded algorithm M, such that for every polynomial $p(\cdot)$, the following inequality holds

$$\Pr[E(View_{P,V}) \in \Pi(X)] < \Pr[M(X_n) \in \Pi(X)] + \frac{1}{p(|X|)}.$$

[2] In the proof of Theorem 10 in [FS01], it is claimed that the probability of the distinguisher who can distinguish between the real transcript and a simulated transcript is equal to distinguishing two decision-Diffie Hellman type distributions, when we take distribution over random input. However, in the definition of computational zero-knowledge, it should be argued in the case where the input is fixed. Apparently, if the input is fixed, no similar reduction will be established.

Claim x: The protocol of [FS01] does not leak any information about the permutation.

Claim y: The proposed protocol does not leak any information about the permutation.

Proofs of claims will be presented in the final version.

3 Implementation and its Results

We built a voting system based on the above technology. We first explain the model we employed in the system. The clarification of the responsibilities of the entities involved was quite indispensable for designing the system. It enabled us eliminate redundant computation. For example, theoretically frauds can be detected if all the centers performed verification process at every occasion. By employing a center who is responsible for verification, we can reduce the number of verifications and still have a secure system.

Then we give the procedures each entities follow. We tried to explain them in detail, including side trivial issues which is often omitted from theoretical papers whose purpose is mainly to describe new ideas.

3.1 Model

We involve five kinds of players, which are

1. Election policy committee
2. Voting center
3. Shuffling management center
4. Shuffling center
5. Voters

The election policy committee will be responsible for any fraud caused by the voting center, the shuffling management center, and the shuffling centers. The election policy committee does not engage in actual run of the electronic voting. It takes part in determining election and security policies, and assignment of the centers. The committee authorizes the output computed by the other centers, such as the parameters determined in a set-up phase and the final tally.

The voting center is in charge of handling transactions with the voters. It will announce the voting procedures, collects pro forma votes from the authorized voters, issues receipts of the collected votes, and announces the result of the tally. The voting center will receive the result of the tally by sending the list of collected votes to the shuffling management center.

The shuffling management center is responsible for decrypting and tallying the list sent from the voting center, in collaboration with the shuffling centers. The shuffling management center passes the list to the first shuffling center, and collects his answer which will be sent to of the next shuffling center, and repeats the process until all the assigned shuffling centers shuffle and decrypt the list. The shuffled result of decryption will be sent back to the voting center. The

shuffling management center is also responsible for composing a public key in the set-up phase, again in collaboration with the shuffling centers.

The shuffling center is responsible for secure management of the secret key generated in the set-up phase, and conducting decryption using the key. He is also responsible for randomly shuffling the list and keeping the shuffle confidential.

We assume all entities has his public key and corresponding secret key assured by some public key infrastructure, and the secret key is securely managed by the individual.

We require for the universal verifiability, that any party can verify that all centers conducted correctly based on the policy approved by the election policy committee. Our goal in vote privacy is that it will not be infringed as long as at least one shuffling center remains honest.

Figure 1 illustrates how these players constitute an voting system. We note that the roles of the voting center and the shuffling management center can be played by one entity.

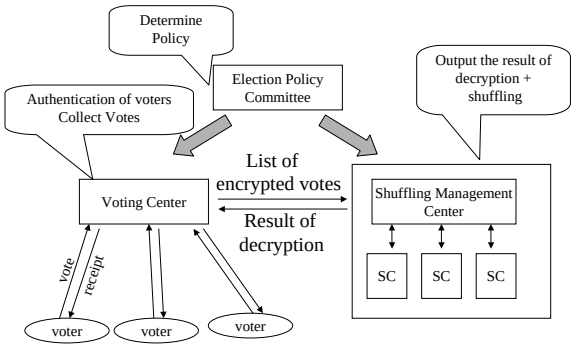

Fig. 1. System Configuration

3.2 Protocol

In this subsection we describe the procedure to set-up, the algorithm to encrypt votes, procedures to tally the votes and the algorithm to prove the correctness of the output. The protocol follows the lines of [SK95] with some minor changes for additional security. This protocol resolves the flaw in [SK95] pointed out by [MH96].

In the sequel, we assume there are m shuffling centers and n voters. All the communication between the entities are digitally signed based on a public key infrastructure.

Set-Up

1. The election policy committee will determine the parameters (p, q, g) which will be used in ElGamal cryptosystem. The numbers p and q are prime numbers satisfying $p = kq + 1$ for some integer k, and g is a generator of a group of order q on mod p.
   The shuffling management center announces the authorized parameters (p, q, g) to all the shuffling centers. The j-th shuffling center, SC_j, will randomly choose x_j mod q as his secret key, and report his public key $y_j = g^{x_j} \bmod p$ to the shuffling management center. The report will be accompanied by the proof y'_j, r_j which ensures that SC_j indeed knows the secret x_j corresponding to y_j. The proof will be generated by SC_j as follows.

$$\begin{aligned} y'_j &= g^{\beta_j} \bmod p \\ c_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ r_j &= c_j x_j + \beta_j \bmod q \end{aligned}$$

   with a randomly generated $\beta_j \in \mathbf{Z}/q\mathbf{Z}$.
2. The shuffling management center will verify the proof y'_j, r_j for each public key $y_j (j = 1, \cdots, m)$ as follows.

$$\begin{aligned} c_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ g^{r_j} {y_j}^{-c_j} &= y'_j \bmod p \\ {y_j}^q = 1 \bmod p \ &, \ y_j \neq 1 \bmod p \end{aligned}$$

   The verified public keys are combined to compose the common public key Y.

$$Y = \prod_{j=1}^{m} y_j \bmod p$$

   The proof for each public key is necessary to ensure that the common public key Y is not generated under a control of some shuffling centers.
3. The election policy committee will certify the public keys y_j and Y properly generated as above.

Encryption of Votes

The $Voter_i$ will use the parameters Y and (p, q, g) certified by the election policy committee and encrypt his vote m_i as follows. (We assume here that m_i is selected to be an element of order q.)

$$(G_i, M_i) = (g^{\bar{r}_i}, m_i Y^{\bar{r}_i}) \bmod p$$

where $\bar{r}_i$ is an element randomly chosen by the $Voter_i$, and ID_i is information that identifies the voter. He then proves the knowledge of m_i by generating the

proof α_i, t_i by

$$\begin{aligned} \alpha_i &= g^{\gamma_i} \bmod p \\ c_i &= \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i) \\ t_i &= c_i \bar{r}_i + \gamma_i \bmod q \end{aligned}$$

with a randomly generated γ_i. This proofs ensures that the voter who knows the content of the vote has generated the vote; a vote duplication attack by copying someone else's encrypted vote will be thwarted here.

The voting center will verify the signature and make sure that the sender is a registered voter and has not voted before. Then it will verify that the proof satisfies

$$\begin{aligned} c_i &= \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i) \\ g^{t_i} {G_i}^{-c_i} &= \alpha_i \bmod p \end{aligned}$$

and that the elements G_i and M_i are both of order q. If everything is verified, then it can optionally send back a receipt of acceptance. Such a receipt cuts in two ways: it will add confidence to the voter that the center indeed accepted his vote and will be an evidence for any disputes on vote delivery. On the other hand, it will serve as a receipt in vote-buying or coercing scenario.

Tallying

The voting center will send the list $\{(G_i, M_i)\}_{(i=1,\cdots,n)}$ to the shuffling management center. The shuffling management center will verify that all of each component is of order q, and rename them to be $(G_i, M_i) = (G_i^{(1)}, M_i^{(1)})$ for all i, which will be the input to the first shuffling center $SC1$.

The list $\{(G_i^{(j)}, M_i^{(j)})\}_i$ will be sent to SC_j. His response will be verified by the shuffling management center and will be renamed to $\{(G_i^{(j+1)}, M_i^{(j+1)})\}_i$ and sent to the next shuffling center. The response from the last shuffling center, SC_m will be verified and sent back to the voting center .

Below, we describe the procedures of each shuffling center.

1. SC_j will receive the list $\{(G_i^{(j)}, M_i^{(j)})\}_i$. He will choose a random permutation $\pi^{(j)}$ and permute the input list $\{(G_i^{(j)}, M_i^{(j)})\}_i$ and achieve the list $\{(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})\}_i$ as follows:

$$\{(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})\} = \{(G^{(j)}_{\pi^{(j)}(i)}, M^{(j)}_{\pi^{(j)}(i)})\}$$

2. The above permutation only changes the order of the ciphertexts, so it is easy to trace the permutation. In order to hide the permutation, we need to change the *look* of the ciphertext. The following procedure changes the look without changing the message hidden in the ciphertext.

First, SC_j combines the public keys of the subsequent shuffling centers as

$$Y_j = \prod_{\ell=j}^{m} y_\ell \bmod p.$$

For each of $(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})$, he chooses a random element $s_i^{(j)} \bmod q$ and obtains $\{(G'^{(j)}_i, M'^{(j)}_i)\}_i$ by

$$G'^{(j)}_i = \bar{G}_i^{(j)} \cdot g^{s_i^{(j)}} \bmod p$$
$$M'^{(j)}_i = \bar{M}_i^{(j)} \cdot Y_j^{s_i^{(j)}} \bmod p.$$

3. SC_j will decrypt each of $(G'^{(j)}_i, M'^{(j)}_i)$ using his secret key x_j as follows:

$$M''^{(j)}_i = M'^{(j)}_i / (G'^{(j)}_i)^{x_j} \bmod p \quad G''^{(j)}_i = G'^{(j)}_i$$

The list $(G''^{(j)}_i, M''^{(j)}_i)_i$ will be returned to the shuffling management center.

3.3 Proving Correctness

We employ the scheme we depicted in Section 2 to prove the correctness of the procedures performed by the shuffling centers. This proof makes the scheme universally verifiable.

In order to save time, when a shuffling center sends back the shuffled and decrypted list with the proofs, the shuffling management center first verifies the signature of the shuffling center and then pass the list to the next shuffling center. While the next shuffling center is executing shuffle, decrypt and prove procedures, the management center verifies the proof of the previous shuffling center. If the management center finds inconsistency in the proof, the procedure stops there and the faulty shuffling center is blamed. How to resume the tally will be of a human decision, taking into the account the cause of the error.

3.4 Computational Tricks

The most costly computation in the protocol is modular exponentiation. Although we have employed a quite efficient proof algorithm, it still requires the number of modular exponentiations which is in linear to the number of voters. However, looking closely at the algorithm, the most exponentiation are computed on the same base g or multiple exponentiation. Thus, We took advantage of this situation and succeeded in speeding up the system. The methods we chose are a variant of "Fixed-base comb method" and a variant of "Simultaneous multiple exponentiation method" exposed in [MOV].

3.5 Result of Implementation

We have evaluated the system under the following conditions:

- Key Size $(|p|, |q|) = (1024, 160)$.
- Security parameter k=160
- The number of shuffling centers $m = 3$.
- CPU:Athlon 1GHz, memory 256Mbyte for each of shuffling centers and shuffling management center.
- Communication Line 100baseTX

As a result, with a ten thousand voters the system can output a certified tally in 20 minutes and with a hundred thousand voters within 224 minutes, including data transmission time. Two-fifths of the time are required for computing the tally, and the rest of the time is devoted to proving and verifying the proof.

	Total Time		Length of Proof	
number of voters	10,000	100,000	10,000	100,000
Only Tallying (No proof)	8 min	1 hrs 10 min	–	–
New Scheme	20 min	3 hrs 44 min	12.6Mbyte	126Mbyte
two proofs with [FS01]	27 min	4 hrs 40 min	15Mbyte	150Mbyte
two proofs with [SK95]	5 hrs 20 min	(53 hrs)	865Mbyte	(8.7 Gbyte)
two proofs with [Abe99]	(9 hrs)	(120 hrs)	(545 Mbyte)	(5.2 Gbyte)
two proofs with [Ne01]	(1 hrs 20 min)	(15 hrs)	(58 Mbyte)	(580 Mbyte)

Table 1. Processing time and proof size

For comparison, we give in Table 1 our result on the alternative implementations, where we used two separate proofs for shuffling and decryption. For shuffling, we give two types of proofs. One is matrix based shuffling proof in [FS01], and the other cut&choose based shuffle proof described in [SK95]. We also provide estimations of the shuffle proof described in [Abe99] and [Ne01]. Those which is shown in parenthesis are based on estimation.

The author of [Ne01] claims that the scheme of [Ne01] requires only $8n$ modular exponentiations compared to $18n$ of [FS01] for the shuffle of n ciphertexts. However, in the latter it includes the cost of the both proving and verifying of shuffle, whereas the former includes only the cost for proving a general k-shuffle. We estimate that the total cost of protocol of [Ne01], including the verification, would be $47n$.

As can be seen from the Table 1, the extra time needed for providing and verifying proof of the new scheme is 12 minutes. Since the same time using two separate proofs of shuffling [FS01] and decryption takes 19 minutes, we can see the speed up of 150%. As for the size of proof, the merged new proof requires only 12.6 Mbytes whereas the two separate proofs requires 15Mbytes.

It is worth noting that although the number of exponentiations required in the scheme of [Abe99] is smaller than that of [SK95], actual time necessary to

complete the protocol does not compliant. This is because the scheme of [SK95] uses a large number of fix-based computation, and merits largely from the table lookup method.

4 Concluding Remarks

In this paper, we discussed on the implementation of shuffling based voting scheme, which realizes many plausible features. We also proposed a proof that proves correctness of shuffle-and-decrypt, which is effectively faster and shorter than proving them separately. We demonstrated that an election with 100,000 voters can output the certificated result within 4 hours.

Although we currently believe that shuffling based voting scheme has many advantages over other schemes, we do not believe that this is the only solution. The privacy provided by the scheme (and also by homomorphic encryption based scheme) is conditional, that it is based on a model that the shuffling centers would not collude to intrude privacy. On the other hand, blind signature based schemes provides perfect secrecy if a voter can have an anonymous channel. Better solutions are always in search for.

References

[Abe99] M. Abe: "Mix-networks on permutation networks," *Advances in Cryptology — ASIACRYPT '99*, pp. 258–273, Springer-Verlag, 1999.

[Br93] S. Brands, *An Efficient Off-line Electronic Cash System Based On The Representation Problem*, CWI Technical Report CS-R9323, (1993)

[Cha81] D. Chaum: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, pp. 84–88, ACM, 1981.

[CY85] J. D. Cohen and M. Yung: "Distributing the power of a government to enhance the privacy of voters," *Annual Symposium on Principles of Distributed Computing*, pp. 52–62, 1985.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers: "A secure and optimally efficient multi-authority election scheme," *European Transactions on Telecommunications*, 8:481–489, 1997. Preliminary version in *Advances in Cryptology — EUROCRYPT '97*.

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers and M. Yung: "Secure Secret Ballot Election Schemes with Linear Work," in *Advances in Cryptology — EUROCRYPT '96*.

[DJ01] I. Damgård and M. Jurik: "A Generalization, a Simplification and some Applications of Paillier's Probabilistic Public-Key system," in *Proc. of Public Key Cryptography 2001*.

[E85] T. ElGamal: "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithm," *IEEE Transactions on Information Theory,* Vol. IT-31, pp. 469–472, 1985.

[EVOX] http://theory.lcs.mit.edu/~cis/voting/voting.html

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta: "A Practical Secret Voting Scheme for Large Scale Elections," *Advances in Cryptology – AUSCRYPT '92*, pp. 244–251, Springer-Verlag, 1992.

[FS86] A. Fiat and A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.

[FS01] J. Furukawa and K. Sako: "An Efficient Scheme for Proving an Shuffle" *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, 2001.

[MOV] Menezes, van Oorschot, Vanstone: "Handbook of Applied Cryptography" CRC Press.

[MH96] M. Michels and P. Horster: "Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme," *Advances in Cryptology – ASIACRYPT '96*, pp. 125–132, Springer-Verlag, 1996.

[Ne01] C.A. Neff, "A Verifiable Secret Shuffle and its Application to E-Voting", ACM-CCS 01 pp. 116-125 2001.

[OMAFO] M. Ookubo, F. Miura, M. Abe A. Fujioka and T. Okamoto: "An improvement of a practical secret voting scheme" Information Security Workshop 1999.

[PIK93] C. Park, K. Itoh, and K. Kurosawa: "Efficient Anonymous Channel and All/Nothing Election Scheme," *Advances in Cryptology – EUROCRYPT '93,* pp. 248–259, Springer-Verlag, 1993.

[Sak94] K. Sako: "Electronic voting schemes allowing open objection to the tally," *Transactions of IEICE,* vol. E77-A No.1, Jan. 1994.

[SK94] K. Sako, J. Kilian: "Secure Voting using Partially Compatible Homomorphisms," *Advances in Cryptology – CRYPTO '94*, pp. 411–424, Springer-Verlag, 1994.

[SK95] K. Sako, J. Kilian: "Receipt-Free Mix-Type Voting Scheme," *Advances in Cryptology – EUROCRYPT'95*, pp. 393–403, Springer-Verlag, 1995.

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

[SENSUS] http://www.ccrc.wustl.edu/~lorracks/sensus/