Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-servers

Masayuki Abe

NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
E-mail: abe@isl.ntt.co.jp

Abstract. In this paper we construct a universally verifiable Mix-net where the amount of work done by a verifier is independent of the number of mix-servers. Furthermore, the computational task of each mix-server is constant against the number of mix-servers except for some negligible tasks like addition. The scheme is robust, too.

Keywords: Mix-net, Electronic Voting, Universal Verifiability

1 Introduction

Many electronic voting schemes have been introduced in the literature so far and some of them are being implemented. In national-scale elections, electronic voting will strongly reduce the cost of counting vast numbers of votes and also offer a high level of security. Not just for large scale elections, electronic voting can also be a useful tool for quick decision making in several types of cooperative projects on networks. Therefore, electronic voting schemes should comply with a wide variety of voting policies [1].

Many efficient schemes, e.g., [8,9,7,2], compute the final tally without opening each ballot for privacy. As they need to assure the validity of ballots by using zero-knowledge proof, their scheme suits only elementary policies like binary choice where the content of a ballot is limited to one of two fixed options. On the other hand, there are schemes wherein each ballot is opened at the end so that one can easily see if the content conforms to the policy which can be complicated like allocative choice where voters distribute assigned points to several options. Typically, schemes along this line, e.g., [6,10], assume anonymous channels to assure the privacy of voters. As a cryptographic alternative to an anonymous channel, Chaum introduced Mix-net [5] where a series of m entities called Mix-servers sequentially decrypts and permutes encrypted ballots so that no collusion of Mix-servers, except all, can distinguish which vote was from which voter. A problem of Chaum's construction, based on RSA, is that the work needed for each voter is proportional to the number of Mix-servers, i.e., each voter has to repeat encryption and randomization m times. Kurosawa et al., overcame this problem in [16] using ElGamal encryption so that the voter's work became

independent of m. Regarding verifiability in Chaum's construction, although each voter can verify that his or her vote has been correctly counted, no voter can be assured that all ballots have been accounted correctly. Such type of Mix-net is called *locally verifiable*. Sako and Killian proposed a *universally verifiable* scheme in [19], where anybody can verify correctness of the result. Universal verifiability is very important especially for large scale elections as it is impractical to force all voters to check the result. Regarding efficiency, since the scheme in [19] is based on [16], voters' work can be independent of the number of Mix-servers but the verifiers must verify that each server behaved correctly by using cut-and-choose method. Thus, the verifier's work remains proportional to m, more precisely, $\mathcal{O}(m \log \varepsilon)$ where ε is acceptable error probability ($\approx 2^{-80}$). Furthermore, their proof system required side information, and it was not known whether this would leak any information about individual votes. Later, in [13], Michels et al. pointed out that the side information can violate anonymity. Although a plausible fix was shown in [18], it still uses side information and the scheme is not known to be secure.

Another model of Mix-net was introduced by Ogata, et al., in [14] which claims universal verifiability and robustness. As their scheme inherits proof systems from [19], the verification work needed for a verifier is $\mathcal{O}(m \log \varepsilon)$. Furthermore, each mix-server must verify all other servers' behavior one by one. The total work done by m servers will be $\mathcal{O}(m^2 \log \varepsilon)$.

In [11] Jakobsson proposed a novel model of robust Mix-net whose complexity is claimed to be $O(m + \frac{\log \varepsilon}{\log N})$ for N votes. Although the scheme is much more efficient than other schemes (including ours), it lacks one important property, that is, universal verifiability. In their scheme, no one except mix-servers can be assured of correctness of the result. If all servers are corrupt, incorrect result may be published without being noticed by anyone.

In this paper we introduce a universally verifiable Mix-net where the verifier's work is $\mathcal{O}(\log \varepsilon)$, i.e., independent of the number of mix-servers. Furthermore, the computational work done by m servers is $\mathcal{O}(m \log \varepsilon)$ if we only take modular exponentiation, which is the most expensive arithmetic operation in our scheme, into account. Our scheme also enjoys robustness. More precisely, our scheme satisfies the following properties.

Robustness If at least t servers are cooperative, the correct result is obtained. That is, the output of the Mix-net is the decryption of the input.

Privacy Unless t or more servers are corrupt, no poly-bounded entity can associate a particular output to input with probability better than random guessing.

Universal Verifiability Correctness of the result is verifiable for any verifiers.

Efficiency The work done by a verifier is independent of the number of Mix-servers. The computational work done by each server is independent of the number of servers except some negligible ones like addition.

This paper is organized as follows. In section 2, we overview our scheme which consists of several steps. Primitives that correspond to each step are detailed in Section 3. The scheme is analyzed in section 4.

2 Overview

We assume four types of participants: users, a bulletin board, mix-servers, and verifiers. All of them are limited to have polynomial-bounded computational resources. The users post encrypted messages to the bulletin board. The encryption is done by El Gamal encryption with mix-servers' encryption key. After submission is closed, mix-servers start working as follows.

The task of the servers can be divided into two phases; "Randomization and permutation" followed by "decryption". The following summarizes our scheme.

1. Randomization and Permutation Phase

(1) Randomization and Permutation The cascade of Mix-servers work to randomize and permute inputs. Each server keeps his local random factors and random permutation secret.

(2) Joint Proof of Permutation Mix-servers cooperate to execute a protocol to issue `Proof-P` that proves correctness of the output in zero-knowledge, that is, the fact that the servers know random factors and permutations that relate the input to the output. Each server verifies the proof himself. If the proof fails, dishonest servers are identified and removed. The remaining servers restart from the beginning.

2. Decryption Phase

(3) Threshold Decryption A quorum of servers cooperate to decrypt the randomized and permuted messages.

(4) Joint Proof of Correct Decryption The servers cooperate to execute a protocol to issue `Proof-D` that proves correctness of the decryption. If the proof fails, dishonest servers are identified and removed. A new quorum of servers including new ones execute decryption again.

The resulting messages are written on the bulletin board together with `Proof-P` and `Proof-D` which prove correctness of the result and which are verifiable by any verifier. As in former published schemes, users have to trust, for privacy, at least one server because privacy can be violated by even the *passive deviation* of all servers. So it is important for honest servers, in order to maintain users' privacy, to assure themselves that their private random choices in the randomization and permutation phase are not leaked, or canceled by other servers. This is possible by verifying each servers' work one by one as done in former schemes. Such a solution, however, needs each server and verifier to perform work proportional to the number of servers. In section 3.3 and 3.5, we demonstrate efficient protocols wherein the work performed by a verifier is constant against the number of servers. The computational work for each server is also constant except some negligible tasks like addition.

The removal of actively deviating servers is done based on the principle that failure of proof at the end of a protocol identifies the dishonest participants. Mix-servers must verify `Proof-P` before they proceed to the decryption phase. If `Proof-P` fails, servers publish a transcription of all internal computation so that

the dishonest servers are identified. To have this strategy work, each server signs their local outputs and the next server verifies the signature. Since such message authentication can be realized as a function of the underlying network, and we will not mention this point explicitly hereafter. If `Proof-D` fails, all internal computations are published except shared decryption keys. We show how to identify dishonest servers in the decryption phase in section 3.5. Note that once the permutation phase has been done correctly, failure in the decryption phase will not endanger privacy.

3 Protocols

3.1 Preliminaries

Let p and q be large primes such that $p = 2q+1$. By G_q we denote a multiplicative subgroup of order q in Z_p^*. Let g be an element of G_q. We assume all arithmetic is done in modulo p, unless otherwise stated. An ElGamal decryption key x of the Mix-network is shared into $x_1, \ldots, x_m$ by Shamir's threshold scheme so that x equals $\sum_{i \in Q} x_i L_i \pmod q$ where $L_i = \prod_{j \in Q, j \neq i} \frac{j}{j-i}$ for any set $Q \subset \{1, \ldots, m\}$ of size k. Server i possesses x_i and $y_i := g^{x_i}$. The encryption key y is distributed to all participants, and y_is are published among servers.

By E_0, we denote a list of encrypted messages appearing on the bulletin board. The j-th entry of the list is $(M_{0,j}, G_{0,j})$ such that $M_{0,j} = v_j y^{t_j}$ and $G_{0,j} = g^{t_{0,j}}$ where $t_{0,j} \in_R Z_q$ is a random number and $v_j \in G_q$ is a message. To avoid the attack shown in [17], the bulletin board must eliminate all copied or correlated messages by having the user prove his/her knowledge of $t_{0,j}$. This can be done efficiently by using the techniques of [20,4].

In the rest of this paper, we use i as an index for servers and runs from 1 to m. Similarly, j is used as an index for messages and runs from 1 to N. We denote the set of all possible permutations on $\{1, \ldots, N\}$ by Π_N. By $\hat{\pi}_i$ for $i \leq m$, we denote a series of $m - i + 1$ permutations $\pi_i \pi_{i+1} \ldots \pi_m$ where $\pi_i \in \Pi_N$. We say π denotes $\pi_1 \ldots \pi_m$.

3.2 Randomization and Permutation

A list of encrypted messages is randomized and permuted by the cascade of m Mix-servers. Task of each server is as follows. Server i receives a list $E_{i-1} := \{(M_{i-1,1}, G_{i-1,1}), \ldots, (M_{i-1,N}, G_{i-1,N})\}$. He then chooses a permutation $\pi_i \in_R \Pi_N$ and N random factors $t_{i,j} \in_R Z_q$ for all j. Randomization and permutation are done as follows.

$$M_{i,j} := M_{i-1,\pi_i(j)} y^{t_{i,\pi_i(j)}}, \text{and} \tag{1}$$

$$G_{i,j} := G_{i-1,\pi_i(j)} g^{t_{i,\pi_i(j)}} \tag{2}$$

for all j. The resulting list $E_i := \{(M_{i,1}, G_{i,1}), \ldots, (M_{i,N}, G_{i,N})\}$ is sent to the next server. The next server works in the same way, and processing succeeds sequentially up to server m.

The result should be $E_m := \{(M_{m,1}, G_{m,1}), \ldots, (M_{m,N}, G_{m,N})\}$ where

$$M_{m,j} = M_{0,\pi(j)} y^{\hat{t}_{m,j}}, \text{and} \tag{3}$$

$$G_{m,j} = G_{0,\pi(j)} g^{\hat{t}_{m,j}}, \text{where} \tag{4}$$

$$\hat{t}_{m,j} = \sum_{i=1}^{m} t_{i,\hat{\pi}_i(j)} \pmod{q}. \tag{5}$$

The following holds under the intractability assumption of the Decision Diffie-Hellman Problem.

Lemma 1. *Given correct E_{i-1} and E_i, no adversary can determine $\pi_i(j)$ for any j with probability better than $1/N$.*

3.3 Joint Proof of Randomization and Permutation

This section introduces a protocol for servers to jointly prove the correctness of randomization and permutation to external verifiers. The proof also convinces honest servers that they have contributed to the output, i.e., no one has canceled the randomization and permutation performed by the honest servers with probability better than guessing them randomly.

Let $\sigma := -\log_2 \varepsilon$ where ε is acceptable error probability ($\varepsilon \leq 2^{-80}$ would be convincing for most applications). We assume the use of a bit commitment scheme BC whose inputs are from $\{1, \ldots, m\} \times \Pi_N \times \prod_{j=1}^{N} Z_q$. As we assume that all participants are polynomial bounded, such BC will be implemented, in practice, assuming the use of a hash function $\mathcal{H} : \{1, \ldots, m\} \times \Pi_N \times \prod_{j=1}^{N} Z_q \to \{0,1\}^{2\sigma}$.

Mix-servers cooperate to run the following protocol σ times.

A-1 Server i receives $\tilde{E}_{i-1} := \{(\tilde{M}_{i-1,1}, \tilde{G}_{i-1,1}), \ldots, (\tilde{M}_{i-1,N}, \tilde{G}_{i-1,N})\}$. He then selects a random permutation $\lambda_i \in \Pi_N$ and calculates $\tilde{M}_{i,j}$ and $\tilde{G}_{i,j}$ as

$$\tilde{M}_{i,j} = \tilde{M}_{i-1,\lambda_i(j)} y^{\tilde{r}_{i,\lambda_i(j)}}, \text{and}$$
$$\tilde{G}_{i,j} = \tilde{G}_{i-1,\lambda_i(j)} g^{\tilde{r}_{i,\lambda_i(j)}},$$

where $\tilde{r}_{i,j} \in_R Z_q$. Server i then sends $\tilde{E}_i := \{(\tilde{M}_{i,1}, \tilde{G}_{i,1}), \ldots, (\tilde{M}_{i,N}, \tilde{G}_{i,N})\}$ to server $i+1$. The last server publishes $\tilde{E}_m$.

A-2 Verifier publishes $c \in_R \{0,1\}$.

A-3 If $c = 0$, server i computes a commitment $b_i := BC(i, \lambda_i, \tilde{r}_{i,1}, \ldots, \tilde{r}_{i,N})$ and distributes b_i to all other servers. After all commitments are exchanged, server i opens b_i by distributing λ_i and all $\tilde{r}_{i,j}$s. The last server publishes $\lambda := \lambda_1 \ldots \lambda_m$ and $\hat{r}_{m,j} := \sum_{i=1}^{m} r_{i,\hat{\lambda}_i(j)} \bmod q$. Each server, then verifies that all b_is, λ and $\hat{r}_{m,j}$ are correctly made. If this check fails, declare CHEATING and stop.

If $c = 1$, server i calculates $\varphi_i := \pi_i^{-1} \varphi_{i-1} \lambda_i$ and $\hat{w}_{i,j} := \hat{w}_{i-1,\lambda_i(j)} + \tilde{r}_{i,\lambda_i(j)} - t_{i,\varphi_{i-1}\lambda_i(j)} \pmod{q}$ (for server 1, let φ_0 be identity permutation, and $\hat{w}_{0,j} = 0$). The last server then publishes $\varphi := \varphi_m$ and all $\hat{w}_{m,j}$s.

A-4 Each server and verifier verify that if $c = 0$,

$$\tilde{M}_{m,j}/M_{0,\lambda(j)} \stackrel{?}{=} y^{\hat{r}_{m,j}}, \text{ and} \tag{6}$$

$$\tilde{G}_{m,j}/G_{0,\lambda(j)} \stackrel{?}{=} g^{\hat{r}_{m,j}}, \tag{7}$$

if $c = 1$,

$$\tilde{M}_{m,j}/M_{m,\varphi(j)} \stackrel{?}{=} y^{\hat{w}_{m,j}}, \text{ and} \tag{8}$$

$$\tilde{G}_{m,j}/G_{m,\varphi(j)} \stackrel{?}{=} g^{\hat{w}_{m,j}}, \tag{9}$$

The following lemma states the security of this protocol.

Lemma 2. *The above protocol is a honest verifier zero-knowledge proof of knowledge for π and $\hat{t}_{i,j}$s that satisfy Equation 3 and 4. Furthermore, the protocol is honest verifier zero-knowledge proof of knowledge for π_is and $t_{i,j}$s held by honest provers that satisfy Equation 3,4 and 5.*

Proof. Completeness holds as follows. In the case of $c = 1$, the left term of Equation 8 can be transformed to

$$\begin{aligned}\tilde{M}_{m,j}/M_{m,\varphi(j)} &= M_{0,\lambda(j)}y^{\hat{r}_{m,\lambda(j)}}/M_{0,\pi\pi^{-1}\lambda(j)}y^{\hat{t}_{m,\pi^{-1}\lambda(j)}} \\ &= y^{\hat{r}_{m,\lambda(j)}-\hat{t}_{m,\pi^{-1}\lambda(j)}}.\end{aligned}$$

Furthermore, $\hat{w}_{m,j}$ can be transformed to

$$\begin{aligned}\hat{w}_{m,j} &= \hat{w}_{m-1,\lambda_m(j)} + \tilde{r}_{m,\lambda_m(j)} - t_{m,\varphi_{m-1}\lambda_m(j)} \\ &= \sum_{i=1}^{m}(\tilde{r}_{i,\hat{\lambda}_i(j)} - t_{i,\varphi_{i-1}\hat{\lambda}_i(j)}) \\ &= \hat{r}_{m,\lambda(j)} - \hat{t}_{m,\pi^{-1}\lambda(j)}.\end{aligned}$$

Hence, a group of provers who knows $t_{i,j}$, $r_{i,j}$, π_i and λ_i can issue φ and $\hat{w}_{m,j}$ that are acceptable in Equation 8. The same is true for Equation 9. In the case of $c = 0$, it is clear that the provers can open all random choices used to compute $\tilde{E}_m$, and predicates 6 and 7 are satisfied.

To see that soundness holds, assume that the provers are able to answer both $c = 1$ and $c = 0$ cases correctly against the same λ and $\tilde{E}_m$. Then, having φ and λ, one can extract π as $\lambda\varphi^{-1} = \lambda(\pi^{-1}\lambda)^{-1} = \pi$. Similarly, getting both $\hat{w}_{m,j}$ and $\hat{r}_{m,j}$ from each case, one can extract $\hat{t}_{m,j}$ by computing $\hat{r}_{m,\pi(j)} - \hat{w}_{m,\varphi^{-1}(j)}$.

It is more involved to extract the individual knowledge of honest provers. We must first show that a successful protocol run implies that instance E_m is made correctly according to each server's choice of π_i and $t_{i,j}$s. It begins by showing that λ_i owned by an honest prover is independent of λ_k for $k \neq i$. Consider the case of $c = 0$. By lemma 1, λ_i is unknown for anybody except server i before A-3 is executed. In A-3, b_i does not leak information about λ_i by assumption on

BC, and it is infeasible for server k to publish λ_k in a different way afterwards. Furthermore, once the commitments are opened and verified correct, committed server's identity k guarantees that b_k is independent of b_i. So we can conclude that server k could have opened b_k before server i opened b_i, i.e., before λ_i is revealed. Thus, λ_i is independent of λ_k. In the same way, we can see that $r_{i,j}$s are independent of $r_{i,k}$s. As each server confirms that published λ_is and $r_{i,j}$s match the relation between $\tilde{E}_m$ and E_0, the absense of a CHEATING concludes that $\tilde{E}_m$ has made correctly based on λ_i and $r_{i,j}$s. Next, consider the case of $c = 1$ in the same protocol run. Succeeding in verification predicates 8 and 9 with correctly made $\tilde{E}_m$ implies that φ involves λ_i. In this case, however, honest prover does not reveal λ_i but φ_i which equals $\pi_i^{-1}\varphi_{i-1}\lambda_i$. Hence, we see that E_m is based on π_i. In the same way, we can see that E_m is based on $t_{i,j}$s. As wrong $\tilde{E}_m$ is detected with probability larger than $1 - \varepsilon$, a successful protocol run implies that the instance E_m is made correctly according to honest servers' knowledge on π_i and $t_{i,j}$s.

Now we are ready to extract individual witnesses. From step A-3 with $c = 0$, we have λ_i of honest server i. Similarly, from step A-3 with $c = 1$, we have φ_i and φ_{i-1}. Thus, we derive $\varphi_{i-1}\lambda_i\varphi_i^{-1} = \varphi_{i-1}\lambda_i\lambda_i^{-1}\varphi_{i-1}^{-1}\pi_i = \pi_i$. We can extract $t_{i,j}$s in a similar manner. Since λ_is are independent, extracted π_is are independent knowledge of honest server. Note that even if some of λ_is are chosen in a dependent manner (possibly because of dishonest behavior by the servers), and hence some π_is owned by dishonest servers are not correctly extracted, it will not influence the fact that the group of servers jointly know π and $\hat{t}_{m,j}$.

To show that the protocol is zero-knowledge, we construct m simulators that work as follows. First they cooperate to pick up $c \in_R \{0, 1\}$. If $c = 0$, simulate each prover following the steps A-1,2, and 4. It should be successful as predicates 8 and 9 do not require knowledge of π and $\hat{t}_{m,j}$. If $c = 1$, each simulator randomly selects fake permutation $\tilde{\lambda}_i$ and fake exponents $\tilde{r}_{i,j}$. Simulators 1 to $m - 1$, then, follow all steps of the protocol. In step A-1, simulator m computes

$$\tilde{M}_{m,j} = M_{m,\tilde{\lambda}_m(j)} Y^{\tilde{r}_{m,\tilde{\lambda}_m(j)}}, \text{ and}$$
$$\tilde{G}_{m,j} = G_{m,\tilde{\lambda}_m(j)} g^{\tilde{r}_{m,\tilde{\lambda}_m(j)}},$$

and publishes $\tilde{E}_m$. In step A-3, simulator m publishes $\varphi = \tilde{\lambda}_m$ and $\hat{w}_{m,j} = \tilde{r}_{m,j}$, which satisfy predicate 8 as

$$\begin{aligned}\tilde{M}_{m,j}/M_{m,\varphi(j)} &= M_{m,\tilde{\lambda}_m(j)} Y^{\tilde{r}_{m,\tilde{\lambda}_m(j)}}/M_{m,\varphi(j)} \\ &= Y^{\tilde{r}_{m,\tilde{\lambda}_m(j)}} \\ &= Y^{\hat{w}_{m,\varphi(j)}}.\end{aligned}$$

Predicate 9 can be satisfied as well. The outputs of each simulator distribute uniformly over G_q or Z_q and so do the permutations. Thus, the view of the external verifier and each prover produced by the simulators and real protocol run are perfectly indistinguishable. Thus the protocol is zero-knowledge both for the servers and external verifiers. □

The lemma implies that once the protocol succeeds, honest servers are convinced that his contribution to randomization and permutation has not been canceled. If CHEATING is declared or verification in A-4 fails, all servers open $r_{i,j}$s and λ_is. This makes all computation traceable and cheating servers can be identified.

Although the described protocol is interactive, we can derive the non-interactive version of the protocol by using Fiat-Shamir heuristics, in which challenge c is made via a hash function. In such a case, `Proof-P` consists of all outputs of the last server. In order to assure verifiers of the presence of a server that each verifier can trust for his privacy, each server signs `Proof-P`.

3.4 Threshold ElGamal Decryption

A quorum of servers cooperate to decrypt messages in the randomized and permuted list. We assume $Q = \{1, \ldots, t\}$ and server $i \in Q$ works to decrypt message $(M_{m,j}, G_{m,j})$ which is in the randomized and permuted list.

The protocol is as follows. Server $i \in Q$ calculates $W_{i,j} := W_{i-1,j} {G_{m,j}}^{x_i L_i}$ (let $W_{0,j} = 1$ for server 1), and sends all $W_{i,j}$ to the next server. The last server, t, then publishes $W_{t,j}$. Decryption is completed by calculating $M_{m,j}/W_{t,j}$.

If all t servers work correctly, the result, $W_{t,j}$, should equal ${G_{m,j}}^{x}$ as

$$W_{t,j} = \prod_{i=1}^{t} {G_{m,j}}^{x_i L_i} = {G_{m,j}}^{\sum_{i=1}^{t} x_i L_i} = {G_{m,j}}^{x}.$$

3.5 Joint Proof of Correct Decryption

Next we introduce a way to jointly prove decryption correctness, i.e., t servers in Q cooperate to prove $\log_g y = \log_{G_{m,j}} W_{t,j}$ for all j.

B-1 Server i chooses $r_i \in_R Z_q$ and computes

$$U_i := U_{i-1}\, g^{r_i}, \text{ and}$$
$$V_{i,j} := U_{i-1,j}\, G^{r_i}$$

(let $U_0 = V_{0,j} = 1$ for the first server) for all j. Then pass U_i and $V_{i,j}$s over to the next server. The last server publishes U_t and $V_{t,j}$.

B-2 Verifier publishes $c \in_R Z_q$.

B-3 Server i computes $s_i := s_{i-1} + r_i - c x_i L_i \bmod q$ (let $s_0 = 0$ for the first server), then sends s_i to the next server. The last server publishes s_t.

B-4 For all j, verifier verifies that

$$U_t \stackrel{?}{=} g^{s_t} y^c, \text{ and} \tag{10}$$
$$V_{t,j} \stackrel{?}{=} G_{m,j}^{s_t} W_{t,j}^{c}. \tag{11}$$

The following lemma states the security of the above protocol.

Lemma 3. *The protocol for the joint proof of decryption is honest verifier zero-knowledge proof of knowledge for relation* $\log_g y = \log_{G_{m,j}} W_{t,j}$ *for all* j.

Proof. Correctness holds because

$$g^{s_t} y^c = g^{\sum_{i \in Q}(r_i - c x_i L_i)} y^c = g^{\sum_{i \in Q} r_i} = U_t.$$

Similarly,

$$G_{m,j}^{s_t} W_{t,j}^c = G_{m,j}^{\sum_{i \in Q}(r_i - c x_i L_i)} W_{t,j}^c = G_{m,j}^{\sum_{i \in Q} r_i} = V_t.$$

For soundness, assume two protocol runs with the same U_t and $V_{t,j}$ but different challenges, say c' and c''. If s'_t and s''_t are the outputs of step B-3 in each protocol run, witness x can be extracted as $x = (s'_t - s''_t)/(c'' - c')$. The protocol is zero-knowledge because for randomly chosen c and $s_1, \ldots, s_t$, the tuple $(\tilde{U}_1, \tilde{V}_{1,j}, \ldots, \tilde{U}_t, \tilde{V}_{t,j}, \ldots)$ such that $\tilde{U}_i := g^{s_i} y^{c/t}$ and $\tilde{V}_{i,j} := G_{m,j}^{s_i} W_{t,j}^{c/t}$ has uniform distribution and satisfies both predicates in step B-4. □

Note that soundness holds only for witness x. That is, the protocol only guarantees that there is a group of provers, the accumulation of whose knowledge leads to x. However, the result is sufficient for successful cases, because what the verifiers should be assured of through this protocol is the correctness of W.

If the proof fails, cheating server can be identified as follows. Suppose the relation between $G_{m,j}$ and $W_{t,j}$ fails. Then $W_{i,j}, U_i, V_{i,j}, s_i$ for all $i \in Q$ are published. If server i has correctly computed,

$$U_i / U_{i-1} = g^{s_i - s_{i-1}} y_i^{cL(i)}, \text{and} \tag{12}$$

$$V_{i,j} / V_{i-1,j} = G_{m,j}^{s_i - s_{i-1}} (W_{i,j} / W_{i-1,j})^c \tag{13}$$

hold, because

$$g^{s_i - s_{i-1}} y_i^{cL(i)} = g^{r_i - cL(i)x_i} y_i^{cL(i)} = g^{r_i} = U_i / U_{i-1},$$

and, in a similar way,

$$G_{m,j}^{s_i - s_{i-1}} (W_{i,j} / W_{i-1,j})^c = G_{m,j}^{r_i - cL(i)x_i} (G_{m,j}^{L(i)x})^c = G_{m,j}^{r_i} = V_{i,j} / V_{i-1,j}.$$

By continuing the verification from $i = t$ to $i = 1$, we must find i where Equation 12 or 13 does not hold as we assume the relation between $G_{m,j}$ and $W_{t,j}$ fails. We note that $W_{i,j}, U_i, V_{i,j}$ and s_i, do not leak x_i as each server's view is simulatable in the same way as we did in the proof of Lemma 2.

As usual, we can derive a non-interactive version of the above protocol by using Fiat-Shamir heuristics, so dishonest verifiers can be put aside. In that case, `Proof-D` consists of all outputs from server t. Similar non-interactive protocols have been used in the context of multi-signatures [3,12], where the security is proved in the random oracle model in [15].

4 Analysis

We claim that privacy is achieved because the protocol for randomization and permutation leaks no useful information about π_is and $t_{i,j}$, and honest servers can assure themselves that their random π_is were not canceled. Correctness is achieved as `Proof-P` guarantees that there are a permutation and random factors that satisfy Equation 3 and 4. Furthermore, `Proof-D` guarantees that the output is the correct decryption of the randomized messages. Robustness is achieved because of threshold decryption, wherein only a quorum of servers need to work to get the messages decrypted. As any verifier can verify `Proof-P` and `Proof-D`, the scheme is universally verifiable.

We discuss efficiency based on computational cost, throughput, and communication complexity. Computational cost is estimated by the number of exponentiation operations. Assuming the use of the simple pre-computation technique, double-base exponentiation is estimated as 1.2 times more expensive than single base exponentiation. Looking at A-4 and B-3, one can see that total work needed by verifiers is completely independent of the number of servers and the cost is $2\sigma N + 1.2(N + 1)$ exponentiations. For each server, $4\sigma N + 5.2N + 2.2$ exponentiations are computed in the successful cases. For instance, if we take 5 servers and $\sigma = 80$, our scheme costs about $1626N$ exponentiations while [14] takes about $4881N$ exponentiations. Sacrificing universal verifiability, one can get down to approximately $130N$ exponentiations with Jakobsson's solution in [11] for $N \approx 10^6$.

Throughput is not necessarily determined by total computational cost as we can employ precomputation and parallel computation by servers. In fact, all exponentiations in the randomization and permutation protocol and in step A-1, that is $2N + 2\sigma N$ for each server, can be computed beforehand. Although exponentiations in other steps must be done during processing, each server can do it in parallel. As a result, one will get an output after $2\sigma N + 3.2N + 2.2$ exponentiations. Using the same setting as above, approximately $163N$ exponentiations are needed for our scheme while $812N$ and $62N$ are needed for [14] and [11] respectively.

Regarding communication complexity, as we can use the non-interactive version of the protocols, verifiers do not need to talk to each server. Servers need several interactions with other servers or the bulletin board as well as other robust schemes.

Acknowledgments

The author wishes to thank E. Fujisaki, M. Michels and A. Fujioka for valuable comments on the early draft of this paper.

References

1. R. Alton-Scheidl, R. Schmutzer, P.-P. Sint, and G. Tscherteu. Voting and rating. Technical report, Research Unit for Socio-Economics, 1997.

2. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *26th STOC*, pages 544–553, 1994.
3. C. Boyd. Multisignatures based on zero knowledge schemes. *Electronics Letters*, 27(22):2002–2004, 1991.
4. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 318–333. Springer-Verlag, 1997.
5. D. Chaum. Untraceable electronic mail, return address, and digital pseudonyms. In *Communications of the ACM*, volume 24, pages 84–88, 1981.
6. D. Chaum. Elections with unconditionally-secret ballots and disruptions equivalent to breaking RSA. In *EUROCRYPT '88*, volume 330 of *LNCS*, pages 177–182. Springer-Verlag, 1988.
7. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 372–382. IEEE Computer Society, 1985.
8. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 72–83. Springer-Verlag, 1996.
9. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer-Verlag, 1997.
10. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer-Verlag, 1993.
11. Markus Jakobsson. A practical mix. In *EUROCRYPT '98*, 1998.
12. M. Michels and P. Horster. On the risk of disruption in several multiparty signature schemes. In *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 334–345. Springer-Verlag, 1996.
13. M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 125–132. Springer-Verlag, 1996.
14. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *ICICS98*, 1997. To appear.
15. K. Ohta and T. Okamoto. The exact security of multi-signature schemes. Technical Report ISEC97-27, IEICE, July 1997.
16. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT '93*, volume 765 of *LNCS*, pages 248–259. Springer-Verlag, 1994.
17. B. Pfitzmann. Breaking an efficient anonymous channel. In *EUROCRYPT '94*, volume 950 of *LNCS*, pages 339–348. Springer-Verlag, 1995.
18. K. Sako. An improved universally verifiable mix-type voting schemes. Unpublished Manuscript, 1995.
19. K. Sako and J. Kilian. Receipt-free mix-type voting scheme –a practical solution to the implementation of a voting booth–. In *EUROCRYPT '95*, volume 921 of *LNCS*, pages 393–403. Springer-Verlag, 1995.
20. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.