Mix-Networks on Permutation Networks

Masayuki ABE

NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
abe@isl.ntt.co.jp

Abstract. Two universally verifiable mix-net schemes that eliminate the cumbersome Cut-and-Choose method are presented. The construction is based on a permutation network composed of a network of 'switches' that transposes two inputs. For N inputs and t tolerable corrupt mix-servers, the schemes achieve $\mathcal{O}(tN \log N)$ efficiency in computation and communication while previous schemes require $\mathcal{O}(\kappa m N)$ for error probability $2^{-\kappa}$ and m mix-servers. The schemes suit small to middle-scale secret-ballot electronic elections. Moreover, one of the schemes enjoys less round complexity so that servers do not need to talk to other servers except their neighbors unless disruption occurs.

1 Introduction

Mix-net is a cryptographic primitive that consists of a series of mix-servers that decrypt input ciphertexts and output the results in random order; thus it cuts the link between inputs and outputs. It has been mainly used to construct electronic secret ballot voting schemes where privacy should be protected.

Early constructions e.g., [5] and [13] are weak against the active deviation of mix-servers. For instance, any one server can control the output by stealthily replacing messages unless every user makes sure that his/her message appears among the output. In [15], Sako and Killian introduced a universally verifiable scheme by which the correctness of the entire output could be verified by anybody [15]. Robustness was first added in [12]. These later schemes are based on the cumbersome Cut-and-Choose method where each server and verifier suffers $\mathcal{O}(\kappa m N)$ computation and communication cost for m servers, N inputs, and error probability $1/2^{\kappa}$. In [1], the cost for a verifier was reduced to $\mathcal{O}(\kappa N)$ at the cost of an increase in communication between servers.

In [10,11], Jakobsson introduced robust schemes that are efficient in computation for large N. However, unlike the previous universally verifiable schemes, a verifier need to trust at least one server to convince himself of the correctness. Such a model would be sufficient when managers of mix-servers are selected from parties which can discredit each other. Thus, for instance, such schemes suit large-scale election where no conspiracy is expected. On the contrary, this paper addresses universally verifiable schemes where the correctness can be verified without trusting any server. Such a property prevents managers of mix-servers

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 258–273, 1999.

from being blackmailed or committing bribery because the final output is out of their control.

Another drawback of previously known robust schemes is that the servers must interact each other frequently to verify intermediate results. If such schemes are implemented on a real network such as the Internet where the cost is excessive in terms of overhead, round complexity may dominate the total efficiency.

Our contribution: This paper presents two efficient robust and universally verifiable mix-nets, referred as MiP-1 and MiP-2, that are the first to eliminate Cut-and-Choose. The heart of our construction is the use of *Permutation Networks* where inputs can be arbitrarily permuted according to a control signal. Both of our schemes offer $\mathcal{O}(t\,N\,\log N)$ computation/communication efficiency where t is the number of tolerable corrupt servers. As complexity grows faster in N than the previous schemes, our solution suits small to moderate scale applications (but covers many practical cases). For instance, for $N < 2^{25}$, both schemes require less computation than the scheme in [12] at $\kappa = 80$ and $m = 5$. Furthermore, MiP-1 has an attractive property in that servers only need to send data to the next server unless disruption happens.

2 Model

Participants: Users $\{\mathsf{U}_i\}_{i\in\{1,\dots,N\}}$, mix-servers $\{\mathsf{M}_j\}_{j\in\{1,\dots,m\}}$, and a verifier V. All of them are polynomially bounded.

Communication Channel: We assume the use of a bulletin board BB where participants read and write in authenticated manner. No one can cancel any information once written to the BB.

Adversary: It is assumed that there exists a polynomially bounded adversary A who can corrupt up to some fixed number of mix-servers and users. We denote the set of corrupt users and servers by $\mathcal{U}_{\mathsf{A}}$ and $\mathcal{M}_{\mathsf{A}}$ respectively. Similarly, $\mathcal{U}_H$ and $\mathcal{M}_H$ denote honest users and servers respectively. A controls $\mathcal{U}_{\mathsf{A}}$ and $\mathcal{M}_{\mathsf{A}}$ in an arbitrary way to break anonymity or to create incorrect output.

Security: Here we present a sketch of the definitions of security properties. Let $E_y(msg, r)$ denote the probabilistic encryption of message msg with random factor r and encryption key y. Let $D_x()$ be the corresponding decryption function. The space for plain messages defined by the encryption function is denoted by MSG_E. An application may define its own message space $\Sigma \subseteq \mathrm{MSG}_E$. Let $(E_1, \dots, E_N)$ denote an input to a mix-net, and $(\upsilon_1, \dots, \upsilon_N)$ denote the output obtained as a result of invocation of Mix-net protocols with respect to the input.

Definition 1 (Correctness). *The output $(\upsilon_1, \dots, \upsilon_N)$ is correct if there exists a permutation between $(\upsilon_1, \dots, \upsilon_N)$ and $(D_x(E_1), \dots, D_x(E_N))$.*

Definition 2 (Universal Verifiability). *Let $view_{pub}$ be all the information written in the BB during an execution of the mix-net protocols. A mix-net is verifiable if there exists a (probabilistic) polynomial-time algorithm V that, on* *input* $view_{pub}$*, outputs* accept *if* $(\upsilon_1, \ldots, \upsilon_N)$ *is correct, otherwise outputs* reject *with overwhelming probability.*

Let $\mu_i \in \Sigma$ be the plaintext chosen by user U_i. Σ_H denotes the choice of honest users, i.e., $\Sigma_H = \{\mu_i \mid \mathsf{U}_i \in \mathcal{U}_H\}$. Let $\mathcal{M}_I$ be an ideal-model mix (oracle) that takes $(\mu_1, \ldots, \mu_N)$ as inputs from the users via authenticated untappable channels and outputs $(\upsilon_1, \ldots, \upsilon_N)$ that satisfies $\upsilon_i = \mu_{\pi(i)}$ for all i and some permutation π on $\{1, \ldots, N\}$. Let AI be an ideal-model adversary such that given $\mathcal{M}_I$ and $\mathcal{U}_H$ as black boxes, outputs (i, μ) that satisfies $\mathsf{U}_i \in \mathcal{U}_H$ and $\mu \in \Sigma_H$. Similarly, let A be a real-life adversary such that given $\mathcal{M}_H$ and $\mathcal{U}_H$ as black boxes, outputs (i, μ) that satisfies $\mathsf{U}_i \in \mathcal{U}_H$ and $\mu \in \Sigma_H$.

Definition 3 (Anonymity). *A mix-net is* (t_u, t_m)*-anonymous if, for all polynomially bound ideal-model adversaries* AI *that can corrupt at most* t_u *users and for all polynomially bound real-life adversaries* A *that can corrupt at most* t_u *users and* t_m *mix-servers,*

$$\left| Pr[(i, \mu) \leftarrow \mathsf{A}^{\mathcal{M}_H, \mathcal{U}_H} \,:\, \mu = D_x(E_i)] - Pr[(i, \mu) \leftarrow \mathsf{AI}^{\mathcal{M}_I, \mathcal{U}_H} \,:\, \mu = \mu_i] \right|$$

is negligible.

Remark: The above definition does not cover the case where all users are in $\mathcal{U}_{\mathsf{A}}$ because, then, there is no anonymity to maintain.

Definition 4 (Robustness). *A mix-net is* (t_u, t_m)*-robust if, for any adversary* A *that controls* $\mathcal{U}_{\mathsf{A}}$ *and* $\mathcal{M}_{\mathsf{A}}$ *that satisfy* $|\mathcal{U}_{\mathsf{A}}| \leq t_u$ *and* $|\mathcal{M}_{\mathsf{A}}| \leq t_m$*,* $\mathcal{M}_H$ *stops in polynomial-time with correct output with overwhelming probability.*

3 Overview

The clear structural difference between our two proposals is whether they use two phases (permutation phase, decryption phase) or one phase (randomized decryption phase). The two-phase scheme, referred to as MiP-2, bears some similarity to the schemes in [12,1] in that at the end of the first phase all servers must agree on whether they proceed to the second phase or not. This contrasts with the one-phase scheme, referred to as MiP-1, where no such synchronization is needed as long as no disruption occurs. Let us begin with MiP-2 which is rather easy to comprehend.

3.1 Sketch of MiP-2

MiP-2 consists of two phases: (1) the permutation phase in which inputs are randomized and randomly permuted, and (2) the decryption phase in which the result of the first phase is decrypted.

Let us first review the permutation network which is the heart of the permutation phase. A *permutation network* is a circuit which, on input $(1, \ldots, N)$ and an

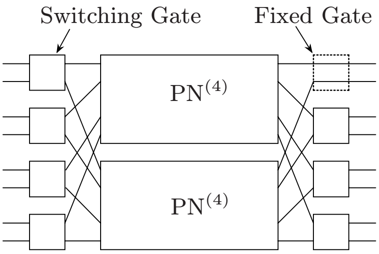

Fig. 1. PN$^{(8)}$ with PN$^{(4)}$

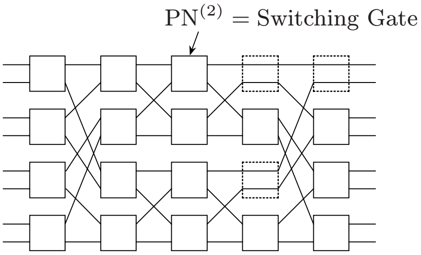

Fig. 2. PN$^{(8)}$ after decomposing PN$^{(4)}$

arbitrary permutation $\Pi : \{1,\dots,N\} \to \{1,\dots,N\}$, outputs $(\Pi(1),\dots,\Pi(N))$. Let PN$^{(N)}$ denote a permutation network with N inputs hereafter. In this paper, we consider N to be a power of 2. A *switching gate* is a permutation network for two inputs, that is PN$^{(2)}$. It has two pairs of input/output terminals labeled by I_0, I_1, O_0 and O_1. It also accepts control signal $b \in \{0,1\}$. According to the control signal, it outputs either $(O_0, O_1) = (I_0, I_1)$ or $(O_0, O_1) = (I_1, I_0)$. Suppose a switching gate provides delay of 1. The following theorem holds.

Theorem 5. *For any N of power of 2, there exists PN$^{(N)}$ that consists of $N \log_2 N - N + 1$ switching gates and provides $2 \log_2 N - 1$ delay.*

Please refer [17] for proof and detailed construction. Here we use intuitive figures to illustrate the recursive construction of PN$^{(8)}$. Dotted boxes indicate *fixed gates* that simply output the inputs. We refer to the gates in the column nearest to the output terminal as *output gates*.

Suppose that a switching gate randomizes inputs. It outputs the results in random order to conceal the correspondence between the inputs and outputs. Let $E_y(\cdot)$ and $R_y(\cdot)$ be a probabilistic encryption function and a randomization function with respect to encryption key y that satisfy $R_y(E_y(msg, s), r) = E_y(msg, s + r)$ for any message $msg \in \mathrm{MSG}_E$ for random factors s, r and for some binary operation '+'. Let $I_0 := E_y(msg_0, s_0), I_1 := E_y(msg_1, s_1)$ be inputs to a switching gate. The switching gate selects random factor r_0, r_1 and outputs $O_b := R(I_0, r_0) = E_y(msg_0, s_0 + r_0)$ and $O_{\bar{b}} := R_y(I_1, r_1) = E_y(msg_1, s_1 + r_1)$ according to $b \in_R \{0,1\}$. If E_y provides indistinguishability, that is, it is infeasible to determine $\beta \in_R \{0,1\}$ given $(msg_\beta, msg_{\bar{\beta}})$ and (I_0, I_1), then it is also infeasible to determine b when (I_0, I_1) and (O_0, O_1) are given. Accordingly, the switching gate conceals the correspondence between its inputs and outputs. By using such switching gates for PN$^{(N)}$, N randomized ciphertexts are output in random order.

To provide verifiability, the switching gate executes a zero-knowledge proof that guarantees correct application of R_y without revealing r_0, r_1, b. Such a proof can be done by combining the Chaum-Pedersen protocol [6], which proves the equality of two discrete logarithms, with the protocol used in [7], which proves two statements connected by OR. This costs about four times as much computation as the Chaum-Pedersen protocol (see section 5.2).

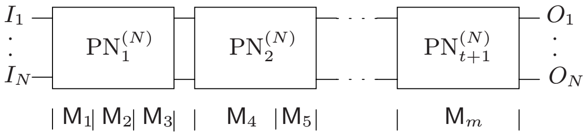

Fig. 3. Series of permutation networks. Each server is assigned some gates within a network.

To tolerate at most t honest but curious servers (i.e., passive adversaries), we construct the permutation phase with $t+1$ permutation networks connected in series as shown in Figure 3. As illustrated, each server is assigned some gates within a network and never works for two or more networks. With this structure, we have at least one adversary-free permutation network among them. Accordingly, those networks can yield any permutation of N inputs unknown to at most t passive adversaries.

When more than t servers agree that the permutation phase is done correctly, they proceed to the decryption phase where they decrypt the result of the first phase in collaboration.

3.2 Sketch of MiP-1

MiP-1 consists of only one phase, the randomized decryption phase, where a series of servers perform decryption and random permutation at the same time.

Similar to MiP-2, a series of $t+1$ permutation networks are used for constructing MiP-1. This time, however, each server is assigned to one or more column of gates within a permutation network, in contrast to MiP-2 where servers can be assigned to any gate.

Let d be the total number of columns of $t+1$ permutation networks. A server assigned to the gates in j-th column has an encryption key y_j and decryption key x_j. Let x be $x = x_1 \oplus \cdots \oplus x_d$ and $y = y_1 \otimes \cdots \otimes y_d$ for some commutative binary operations '$\oplus$' and '$\otimes$'. Let $\hat{y}_j$ denote $y_j \otimes \cdots \otimes y_d$ (so $y = \hat{y}_1$). Suppose that decryption function $D_{x_j}(\cdot)$ satisfies $D_{x_j}(E_{\hat{y}_j}(msg, s)) = E_{\hat{y}_{j+1}}(msg, s)$ for any appropriate message msg and random factor s. Namely, $D_{x_j}(\cdot)$ is a function that transforms a ciphertext with $\hat{y}_j$ into another with $\hat{y}_{j+1}$ associated with the same plain message. (For $j = d$, let $D_{x_d}(E_{y_d}(msg, s)) = msg$.)

Let $I_0 := E_{\hat{y}_j}(msg_0, s_0), I_1 := E_{\hat{y}_j}(msg_1, s_1)$ be inputs to a switching gate in the j-th column. The switching gate randomly chooses r_0, r_1 and applies $D_{x_j}(\cdot)$ and $R_{\hat{y}_{j+1}}(\cdot)$ to I_0 and I_1 as $O_b := R_{\hat{y}_{j+1}}(D_{x_j}(I_0), r_0)$, $O_{\bar{b}} := R_{\hat{y}_{j+1}}(D_{x_j}(I_1), r_1)$ for $b \in_R \{0,1\}$. It then outputs O_0, O_1. Observe that $O_b := R_{\hat{y}_{j+1}}(D_{x_j}(I_0), r_0) = R_{\hat{y}_{j+1}}(E_{\hat{y}_{j+1}}(msg_0, s_0), r_0) = E_{\hat{y}_{j+1}}(msg_0, s_0 + r_0)$. As a result, the output of a switching gate in the $(j+1)$-th column is a message encrypted with key $\hat{y}_{j+1}$. Therefore, by applying this procedure to the last column of the series of permutation networks (the output gates only apply $D_{x_d}(\cdot)$), we obtain the plaintext as $D_{x_d}(R_{\hat{y}_d}(D_{x_{d-1}} \cdots R_{\hat{y}_2}(D_{x_1}(I_0), r_0) \cdots)) = D_{x_d \oplus \cdots \oplus x_1}(I_0) = msg_0$.

For verifiability, each gate proves in zero-knowledge that $D_{x_j}(\cdot)$ and $R_{\hat{y}_{j+1}}(\cdot)$ have been applied correctly without revealing x_j, r_0, r_1, b. It is about 6 times more costly than the Chaum-Pedersen protocol (see Section 6.2). Fixed gates in the j-th column only apply $D_{x_j}()$ to inputs. In this case, the proof costs about half of that of the switching gates.

4 Encryption Scheme

To achieve anonymity, inputs to a mix-net must be encrypted in a non-malleable way. Although the robust threshold Cramer-Shoup cryptosystems [4,2], which are provably non-malleable, can be used for our purpose we employ the following scheme which combines El Gamal encryption and Schnorr signature for the sake of efficiency. For a rigorous discussion of the security of the scheme, please refer to [16].

Let $\mathcal{G}$ be a discrete logarithm instance generator, which, on input security parameter 1^n, outputs (p, q, g) where p, q are large primes that satisfy $p = 2q+1$ and g is an element of subgroup G_q of order q of multiplicative group Z_p^*. We assume the intractability of solving the discrete logarithm in G_q. Hereafter, all arithmetic operations are done in mod p unless otherwise stated.

Let $(x, y) \in Z_q \times G_q$ be a decryption and encryption key pair that satisfy $y = g^x$. A ciphertext of message $msg \in G_q$ is (M, G, c, z) where $M = msg\ y^s$, $G = g^s$, $c = \mathcal{H}(M, G, g^w)$, $z = w - c\,s \bmod q$ for a hash function $\mathcal{H} : \{0,1\}^* \rightarrow 2^{|q|}$ and random number $s, w \in_U Z_q$. Decryption is done by first verifying whether $c = \mathcal{H}(M, G, g^z G^c)$, and $M, G \in G_q$. If it is successful, the output is $msg := M/G^x$.

The input to a mix-net is a list of (M, G)-s that successfully passes the above verifications. The list is assumed not to contain duplicated (M, G)-s.

5 Details of MiP-2

5.1 Preliminaries

Common parameters p, q, g are generated by a generator and are published to all participants in an authentic way. Threshold t is selected from 0 to $\lfloor \frac{m-1}{2} \rfloor$. As described in the overview, each server is assigned some switching gates within the same $\mathrm{PN}^{(N)}$ in a series of $t+1$ $\mathrm{PN}^{(N)}$. Though each server can be assigned to any (not necessarily adjacent) gates, we assume, hereafter, that all the gates of M_k are connected to the gates of M_{k+1} just for efficiency and simplicity. This limitation does not affect the descriptions given in Section 5.2. Obvious modification of descriptions in Section 5.3 will yield general description for MiP-2.

All servers cooperate to execute the key generation protocol [14,9]. As a result, decryption key x is shared into $x_1, \ldots, x_m$ by using the $(t+1, m)$-threshold scheme so that server M_k has x_k privately. Corresponding public keys $y := g^x$ and all $y_k := g^{x_k}$ are published.

5.2 Task of Switching Gates

The task of a switching gate is twofold: the main operation and the proof of correctness of the operation. Server M_k executes the following for each switching gate in its charge. Let $I_i = (M_i, G_i)$ for $i = 0, 1$ be inputs to a switching gate.

[Main Operation]
Choose $r_0, r_1 \in_R Z_q$, and $b \in_R \{0, 1\}$. Compute

$$O_b := (\tilde{M}_b, \tilde{G}_b) = (M_0 y^{r_0}, G_0 g^{r_0}), \tag{1}$$

$$O_{\bar{b}} := (\tilde{M}_{\bar{b}}, \tilde{G}_{\bar{b}}) = (M_1 y^{r_1}, G_1 g^{r_1}). \tag{2}$$

[End]

M_k then proves in zero-knowledge that

$$\text{(St-1):} \quad \log_y \tilde{M}_0/M_0 = \log_g \tilde{G}_0/G_0 \quad \wedge \quad \log_y \tilde{M}_1/M_1 = \log_g \tilde{G}_1/G_1$$

or

$$\text{(St-2):} \quad \log_y \tilde{M}_0/M_1 = \log_g \tilde{G}_0/G_1 \quad \wedge \quad \log_y \tilde{M}_1/M_0 = \log_g \tilde{G}_1/G_0$$

is satisfied. Observe that (St-1) resp. (St-2) should hold for $b = 0$ resp. 1.

[Proving (St-1) $\vee$ (St-2)]

SP$_1$-1: M_k selects $w_0, w_1, z_{\bar{b},0}, z_{\bar{b},1}, e_{\bar{b}} \in_R Z_q$, and computes, for $i = 0, 1$,

$$\begin{aligned} T_{b,i} &:= y^{w_i}, \\ W_{b,i} &:= g^{w_i}, \\ T_{\bar{b},i} &:= y^{z_{\bar{b},i}} (\tilde{M}_{\bar{b}\oplus i}/M_i)^{e_{\bar{b}}}, \quad \text{and} \\ W_{\bar{b},i} &:= g^{z_{\bar{b},i}} (\tilde{G}_{\bar{b}\oplus i}/G_i)^{e_{\bar{b}}} \quad (\oplus \text{ means XOR hereafter}). \end{aligned}$$

It then sends $(T_{0,0}, W_{0,0}, T_{0,1}, W_{0,1}, T_{1,0}, W_{1,0}, T_{1,1}, W_{1,1})$ to V.

SP$_1$-2: V sends $c \in_R Z_q$ to M_k.

SP$_1$-3: M_k computes $e_b := c - e_{\bar{b}} \bmod q$ and $z_{b,i} := w_i - e_b\, r_i \bmod q$ for $i = 0, 1$. M_k then sends $(e_0, e_1, z_{0,0}, z_{0,1}, z_{1,0}, z_{1,1})$ to V.

SP$_1$-4: V first checks $c \overset{?}{=} e_0 + e_1 \bmod q$ and verifies that, for $b = 0, 1$ and $i = 0, 1$,

$$\begin{aligned} T_{b,i} &\overset{?}{=} y^{z_{b,i}} (\tilde{M}_{b\oplus i}/M_i)^{e_b}, \text{ and} \\ W_{b,i} &\overset{?}{=} g^{z_{b,i}} (\tilde{G}_{b\oplus i}/G_i)^{e_b}. \end{aligned}$$

If any check fails, V outputs False. True, otherwise.

[End]

Though SP$_1$ is described in interactive manner, a non-interactive protocol is obtained by employing Fiat-Shamir heuristics [8].

Lemma 6. *SP_1 is an honest verifier zero-knowledge proof for (St-1) $\vee$ (St-2).*

Proof. Correctness and zero-konwledgeness can be shown in a straightforward way. To prove soundness, we execute SP_1 twice with different challenge c_1, c_2 in step SP_1-2 by rewinding M_k. Let the answers obtained in SP_1-3 be $(e_0, e_1, z_{0,0}, z_{0,1}, z_{1,0}, z_{1,1})$ and $(e'_0, e'_1, z'_{0,0}, z'_{0,1}, z'_{1,0}, z'_{1,1})$. Since $e_0 + e_1 \neq e'_0 + e'_1$, at least either $e_0 \neq e'_0$ or $e_1 \neq e'_1$ holds. Suppose $e_0 \neq e'_0$. Then, since $y^{z_{0,0}}(\tilde{M}_0/M_0)^{e_0} = y^{z'_{0,0}}(\tilde{M}_0/M_0)^{e'_0}$ holds, $r_0 := \log_y(\tilde{M}_0/M_0) = (z_{0,0} - z'_{0,0})/(e'_0 - e_0)$ is obtained. Similarly, from $g^{z_{0,0}}(\tilde{G}_0/G_0)^{e_0} = g^{z'_{0,0}}(\tilde{G}_0/G_0)^{e'_0}$, we have $\log_g(\tilde{G}_0/G_0) = (z_{0,0} - z'_{0,0})/(e'_0 - e_0) = r_0$. Hence $r_0 = \log_y(\tilde{M}_0/M_0) = \log_g(\tilde{G}_0/G_0)$ is obtained. From $z_{0,1}, z'_{0,1}, e_0, e'_0$, we can compute r_1 that satisfies $r_1 = \log_y(\tilde{M}_1/M_1) = \log_g(\tilde{G}_1/G_1)$ in the same way. Thus, (St-1) is satisfied.

In the case of $e_1 \neq e'_1$, we can show that (St-2) holds in the same way. □

Corollary 7. *If honest V outputs True at the end of SP_1, then O_0, O_1 are ciphertexts whose plaintexts are the same as those of I_0, I_1 regardless of the order.*

Proof. Consider the case where (St-1) holds. Decryption of the outputs yields

$$\begin{aligned}\tilde{M}_0/\tilde{G}_0^x &= M_0\, y^{r_0}/(G_0\, g^{r_0})^x = M_0/{G_0}^x, \text{ and}\\ \tilde{M}_1/\tilde{G}_1^x &= M_1\, y^{r_1}/(G_1\, g^{r_1})^x = M_1/{G_1}^x.\end{aligned}$$

Thus, the results of decrypting inputs and outputs are identical. Similarly, if (St-2) holds, we have

$$\begin{aligned}\tilde{M}_0/\tilde{G}_0^x &= M_1\, y^{r_1}/(G_1\, g^{r_1})^x = M_1/{G_1}^x, \text{ and}\\ \tilde{M}_1/\tilde{G}_1^x &= M_0\, y^{r_0}/(G_0\, g^{r_0})^x = M_0/{G_0}^x.\end{aligned}$$

Thus, the results of decryption are identical except for the order.

If (St-1) and (St-2) hold at the same time then, from above equations, we have $M_0/{G_0}^x = M_1/{G_1}^x$. Therefore, such a case happens only if two inputs are made from the same plaintext. □

Lemma 8. *If the Decision Diffie-Hellman Problem is intractable, it is infeasible to determine b with probability non-negligibly better than $1/2$.*

Proof. Suppose that there exists a polynomial-time algorithm A_3 that, given $(p, g, y, I_0, I_1, O_0, O_1)$, outputs b with probability non-negligibly better than 1/2. Then, there exists a polynomial-time algorithm A_4 that, given (msg_0, msg_1) and their encryption $(I'_0, I'_1) = (E_y(msg_\beta), E_y(msg_{\bar{\beta}}))$ where $\beta \in_R \{0, 1\}$, outputs β with the same success probability as that of A_3.

Algorithm A_4 encrypts msg_0, msg_1 as $(O'_0, O'_1) = (E_y(msg_0), E_y(msg_1))$, and then it inputs $(p, g, y, I'_0, I'_1, O'_0, O'_1)$ to A_3. Finally A_4 outputs what A_3 outputs.

Observe that O'_0, O'_1 produced by A_4 has the same distribution as O_0, O_1 according to Corollary 7 and the fact that β is randomly chosen. Hence A_3 outputs the correspondence between (I'_0, I'_1) and (O'_0, O'_1) which is the same as the correspondence between (I'_0, I'_1) and (msg_0, msg_1). Thus, if A_3 succeeds, so does A_4. However, if the Decision Diffie-Hellman Problem is intractable, algorithm A_4 does not exist [16]. Thus A_3 does not exist either.

Furthermore, Lemma 6 guarantees that SP_1 does not leak anything about b. Hence it is infeasible to determine b. □

5.3 MiP-2 Protocol Description

[Permutation Phase]
The following steps are repeated for $k = 1, \ldots, m$.

$\mathbf{P}_1$-1: M_k reads the output of M_{k-1} from BB. (For $k = 0$, M_k reads input messages.) It then performs the task of the switching gates in its charge according to the description in Section 5.2, and posts the results to BB.

$\mathbf{P}_1$-2: Every server verifies the output of M_k. If more than $t + 1$ servers declare that it is faulty, M_k is disqualified. Then, the faulty result is just abandoned and the output from M_{k-1} is used for subsequent operation instead.

[End]

As a result, a list of El Gamal ciphertexts, say $\{(\tilde{M}_\ell, \tilde{G}_\ell) \mid \ell = 1, \ldots, N\}$, appears on BB. Let $\mathcal{Q}_1$ be the indexes of servers not disqualified in P_1-2. The remaining servers proceed to the decryption phase, which consists of distributed decryption and zero-knowledge proofs for correctness of the decyption. The following protocol reduces computation complexity for the verifier by using the probabilistic verification [15,3] at the cost of communication complexity.

[Decryption Phase]

$\mathbf{P}_2$-1: Each $\mathsf{M}_{k \in \mathcal{Q}_1}$ chooses $w \in_R Z_q$ and computes $T_{k0} := g^w$ and $D_{k\ell} := \tilde{G}_\ell^{x_k}$, $T_{k\ell} := \tilde{G}_\ell^w$ for $\ell = 1, \ldots, N$. It then computes $e_k := \mathcal{H}(y_k, T_{k0}, D_{k1}, T_{k1}, \ldots, D_{kN}, T_{kN}) \bmod q$ and $s_k := w - e_k x_k \bmod q$. It finally sends s_k and all $D_{k\ell}$, $T_{k\ell}$ to BB.

$\mathbf{P}_2$-2: For each $k \in \mathcal{Q}_1$, V computes e_k in the same way as above and checks if

$$1 = (g^{s_k} y_k^{e_k} T_{k0}^{-1})^{\gamma_0} \prod_{\ell=1}^{N} (\tilde{G}_\ell^{s_k} D_{k\ell}^{e_k} T_{k\ell}^{-1})^{\gamma_\ell}$$

holds for randomly picked $\gamma_1, \ldots, \gamma_N \in Z_q^*$.

[End]

The list of plaintexts can be obtained by computing

$$v_\ell := \tilde{M}_\ell / \prod_{k \in \mathcal{Q}_2} {D_{k\ell}}^{\lambda_{\mathcal{Q}_2,k}}$$

for all $\ell = 1, \ldots, N$ where $\mathcal{Q}_2$ is the indices of servers whose output passes verification in P_2-2 and $\lambda_{\mathcal{Q}_2,k} = \sum_{\xi \in \mathcal{Q}_2 \setminus \{k\}} \frac{\xi}{\xi - k} \bmod q$.

5.4 Security

Theorem 9. *MiP-2 is* $(N-1, \lfloor \frac{m-1}{2} \rfloor)$*-anonymous.*

Proof. We prove this theorem only against static adversaries who determine corrput servers and users in advance. First, when $t \leq \lfloor \frac{m-1}{2} \rfloor$, x is unconditionally secure against A because it is shared by $(t+1, m)$-VSS. Furthermore, the input ciphertexts cast by honest users give only negligibly better advantage to A in distinguishing corresponding plaintexts because the encryption scheme is semantically secure. Next, since there are $t+1$ permutation networks and only t adversarial servers, at least one permutation network consists of honest servers. According to Lemma 8, one can conclude that it is hard for A to determine the permutation applied by the honest permutation network. Furthermore, because the encryption scheme is non-malleabile, the plaintexts chosen by honest users are independent of those of corrupt users with overwhelming probability. So the plaintexts chosen by corrupt users have only negligibly small chance of containing some information about the plaintexts chosen by honest users. Therefore, all information except the resulting plaintexts gives A only a negligible advantage. Thus, the success probability of A differs negligibly from that of AI. This argument obviously holds for arbitrary numbers of corrupt users up to $N-1$. □

Theorem 10. *MiP-2 is* $(N, \lfloor \frac{m-1}{2} \rfloor)$*-robust.*

Proof. According to Lemma 6, any incorrect operation in a gate is detected with overwhelming probability, so actively deviating servers are disqualified at P_1-2 because they are in the minority if $|\mathcal{M}_{\mathsf{A}}| \leq t \leq \lfloor \frac{m-1}{2} \rfloor$. On the other hand, honest servers will never be disqualified because they are in the majority. Similarly, any incorrect computation in the decryption phase will be detected with overwhelming probability because every server must prove correctness in zero-knowledge. From Corollary 7, one can conclude that if all proofs in the permutation phase are correct, then the output of that phase is a list of El Gamal ciphertexts that yields the same set of plaintexts as do the input ciphertexts. Since there are at most t corrupt servers, $\mathcal{Q}_2 \geq m - t \geq t + 1$ must hold. Thus M_k, $k \in \mathcal{Q}_2$ are sufficient to produce correct plaintexts because the decryption key is shared by using the $(t+1, m)$-threshold scheme.

The above argument holds for any input message. Furthermore, all computation in the protocols can be finished in time polynomial against input message length. Since users are polynomial bounded, their input message length is also polynomial bounded. That is, the protocol can be completed in polynomial-time regardless of the inputs. Thus, no deviation of any number of users will affect either the correctness or running time. □

Theorem 11. *MiP-2 is Universally Verifiable.*

Proof. According to Corollary 7, if all proofs in the permutation phase are correct, then the output of that phase is a list of El Gamal ciphertexts that yield the same set of plaintexts as do the input ciphertexts. It is also true that if the proof in the decryption phase is correct, the output of that phase is a result of decrypting the ciphertexts produced by the permutation phase. Furthermore, it is clear that all verification equations use just public variables. □

6 Details of MiP-1

6.1 Preliminaries

Similar to MiP-2, $t+1$ permutation networks are connected in sequence and every server is assigned some gates within a network. In this case, however, a server must be assigned all gates in the same column. Let d be the total number of columns of $t+1$ permutation networks. That is, $d = (2\log N - 1)(t+1)$. Each server will be in charge of d/m columns on average. Suppose that M_k is in charge of the j-th column. M_k generates $x_j \in Z_q$ and $y_j := g^{x_j}$. It then distributes x_j to other servers with $(t+1, m)$-VSS in case M_k should malfunction during subsequent protocols. Let $\hat{y}_j$ denote $y_j y_{j+1} \cdots y_d$. Encryption key y is $y := \hat{y}_1 = y_1 \cdots y_d$, which is published together with all y_j-s.

6.2 Task of Switching Gates

The task of a switching gate in the j-th column is to decrypt the inputs with x_j, randomize the result, and output them in random order. Such a technique was first used in [13].

[Main Operation]
Select $r_0, r_1 \in_R Z_q$, $b \in_R \{0,1\}$ and compute

$$O_b := (\tilde{M_b}, \tilde{G_b}) = (M_0\, {G_0}^{-x_j}\, \hat{y}_{j+1}^{r_0},\ G_0\, g^{r_0}), \text{ and} \tag{3}$$

$$O_{\bar{b}} := (\tilde{M_{\bar{b}}}, \tilde{G_{\bar{b}}}) = (M_1\, {G_1}^{-x_j}\, \hat{y}_{j+1}^{r_1},\ G_1 g^{r_1}). \tag{4}$$

[End]

To guarantee that the main operation step is done correctly, it proves in zero-knowledge that the input and output satisfy

$$\text{(St-3):}\quad \begin{array}{llll} \tilde{M_0}/M_0 = {G_0}^{-x_j}\, \hat{y}_{j+1}^{r_0} & \wedge & \tilde{G_0}/G_0 = g^{r_0} & \wedge \\ \tilde{M_1}/M_1 = {G_1}^{-x_j}\, \hat{y}_{j+1}^{r_1} & \wedge & \tilde{G_1}/G_1 = g^{r_1} & \end{array}$$

or

$$\text{(St-4):}\quad \begin{array}{llll} \tilde{M_1}/M_0 = {G_0}^{-x_j}\, \hat{y}_{j+1}^{r_0} & \wedge & \tilde{G_1}/G_0 = g^{r_0} & \wedge \\ \tilde{M_0}/M_1 = {G_1}^{-x_j}\, \hat{y}_{j+1}^{r_1} & \wedge & \tilde{G_0}/G_1 = g^{r_1} & \end{array}$$

for some r_0, r_1 and for x_j that satisfies $y_j = g^{x_j}$.

[Proof of (St-3) $\vee$ (St-4)]

SP$_2$-1: M_k randomly selects $w_0, w_1, v, z_{\bar{b},0}, z_{\bar{b},1}, s_{\bar{b}}, e_{\bar{b}}$ from Z_q. It then computes, for $i = 0, 1$,

$$\begin{aligned} T_{b,i} &:= G_i^{-v}\, \hat{y}_{j+1}^{w_i}, \\ W_{b,i} &:= g^{w_i}, \\ V_b &:= g^v, \\ T_{\bar{b},i} &:= G_i^{-s_{\bar{b}}}\, \hat{y}_{j+1}^{z_{\bar{b},i}} (\tilde{M}_{\bar{b}\oplus i}/M_i)^{e_{\bar{b}}}, \\ W_{\bar{b},i} &:= g^{z_{\bar{b},i}} (\tilde{G}_{\bar{b}\oplus i}/G_i)^{e_{\bar{b}}} \\ V_{\bar{b}} &:= g^{s_{\bar{b}}}\, {y_j}^{e_{\bar{b}}}, \end{aligned}$$

and sends $(T_{0,0}, W_{0,0}, T_{0,1}, W_{0,1}, V_0, T_{1,0}, W_{1,0}, T_{1,1}, W_{1,1}, V_1)$ to V.

SP$_2$-2: V sends $c \in_R Z_q$ to M_k.

SP$_2$-3: M_k computes

$$\begin{aligned} e_b &:= c - e_{\bar{b}} \bmod q, \\ s_b &:= v - e_b x_j \bmod q, \text{ and} \\ z_{b,i} &:= w_i - e_b\, r_i \bmod q \text{ for } i = 0, 1. \end{aligned}$$

It then sends $(e_0, e_1, z_{0,0}, z_{0,1}, s_0, z_{1,0}, z_{1,1}, s_1)$ to V.

SP$_2$-4: V first checks $e_0 + e_1 \overset{?}{=} c \pmod q$, then verifies that

$$\begin{aligned} T_{b,i} &\overset{?}{=} G_i^{-s_b} \hat{y}_{j+1}^{z_{b,i}} (\tilde{M}_{b\oplus i}/M_i)^{e_b}, \\ W_{b,i} &\overset{?}{=} g^{z_{b,i}} (\tilde{G}_{b\oplus i}/G_i)^{e_b} \\ V_b &\overset{?}{=} g^{s_b} {y_j}^{e_b} \end{aligned}$$

for $b = 0, 1$ and $i = 0, 1$. If it holds, V outputs True, or False otherwise.

[End]

Lemma 12. *SP$_2$ is an honest verifier zero-knowledge proof for (St-3) $\vee$ (St-4).*

Corollary 13. *If honest V outputs True at the end of SP$_2$, then decryption of inputs with key $x_j + \cdots + x_d \bmod q$ and decryption of outputs with key $x_{j+1} + \cdots + x_d \bmod q$ results in the same list of plaintexts without regard to the order.*

The proof follows that for Collorary 7.

Lemma 14. *If the Decision Diffie-Hellman Problem is intractable, it is infeasible to determine b with probability non-negligibly better than $1/2$.*

Proof. (Sketch) Suppose that there exists a polynomial-time algorithm A_5 that, given $(\tilde{M}_0, \tilde{G}_0)$, $(\tilde{M}_1, \tilde{G}_1)$, (M_0, G_0), (M_1, G_1) and $y_j, \ldots, y_d$, $x_{j+1}, \ldots, x_d$ that satisfy Relation 3 and 4, outputs correct b with probability non-legligibly better than 1/2. We can then show that there exists algorithm A_6 that distinguishes the correspondence between a pair of El Gamal ciphertexts for encryption key y_j and a pair of messages with probability non-neglegibly better than 1/2. Since such A_6 does not exist if the Decision Diffie-Hellman Problem is intractable, neither does A_5. Moreover, Lemma 12 states that the proof is zero-knowledge. Thus, it is hard to guess b with probability non-negligibly better than 1/2. □

6.3 Task of Fixed Gates

The fixed gates in the j-th column simply perform decryption as follows.

[Main Operation]

$$(\tilde{M_0}, \tilde{G_0}) := (M_0\, {G_0}^{-x_j},\ G_0)$$
$$(\tilde{M_1}, \tilde{G_1}) := (M_1\, {G_1}^{-x_j},\ G_1)$$

[End]

To guarantee that the above process is done correctly, a server proves in zero-knowledge that the inputs and outputs satisfy

$$\log_{G_0} \tilde{M_0}/M_0 = \log_{G_1} \tilde{M_1}/M_1 = -\log_g y_j.$$

The precise protocol can be easily derived from the Chaum-Pedersen protocol and so is omitted.

6.4 Task of Output Gates

Each output gate performs decryption and random permutation (without randomization) as follows.
[Main Operation]
For $b \in_R \{0, 1\}$, compute

$$\tilde{M_b} := M_0\, {G_0}^{-x_d}, \text{ and}$$
$$\tilde{M_{\bar{b}}} := M_1\, {G_1}^{-x_d}.$$

[End]

The proof for the above process involves showing that the inputs and outputs satisfy

$$\text{(St-5):} \quad \log_{G_0} \tilde{M_0}/M_0 = \log_{G_1} \tilde{M_1}/M_1 = -\log_g y_d,$$

or

$$\text{(St-6):} \quad \log_{G_1} \tilde{M_0}/M_1 = \log_{G_0} \tilde{M_1}/M_0 = -\log_g y_d.$$

The precise protocol can be obtained from SP_2 by removing the factors related to r_0 and r_1.

6.5 MiP-1 Protocol Description

[Randomized Decryption Phase]

P$_3$-1: The following is repeated for $k = 1, \ldots, m$: M_k verifies all proofs issued by previous servers. If successful, M_k takes the output of M_{k-1} as its input and performs the task of each gate in its charge. If it finds M_{bad} faulty, it declares so and goes into the disqualification procedure (see below). Then M_k performs his task by taking the result of public decryption done in the disqualification procedure as its input.

Disqualification Procedure: Every server verifies the proofs issued by M_{bad}. If more than $t+1$ servers conclude that M_{bad} is faulty, all servers from M_{bad} to M_{k-1} are disqualified and all partial decryption keys belonging to those servers are published by VSS reconstruction. Partial decryption with those keys is then done in public in the place of the disqualified servers.

P$_3$-2: V verifies all outputs appearing on the BB. If V finds wrong intermediate results, it invokes the disqualification procedure to obtain the decryption keys of the faulty servers.

[End]

Alhough servers use BB in above description, M_k can directly send all results to M_{k+1} together with all inputs received from M_{k-1} in an authentic and non-repudiable way.

Similar to MiP-2, the following theorem holds.

Theorem 15. *MiP-1 is $(N-1, \lfloor\frac{m-1}{2}\rfloor)$-anonymous, $(N, \lfloor\frac{m-1}{2}\rfloor)$-robust and Universally Verifiable.*

7 Efficiency

Computational efficiency is estimated based on the number of modular exponentiations in $\bmod p$. Since the servers have different amounts of tasks in both of our schemes, we discuss average cases. In the following, we count the cost of computing a single-base modular exponentiation as 1. By assuming the use of the simple table lookup method, double, triple, and quadruple-base modular exponentiation are counted as 1.2, 1.25, and 1.31 respectively. Table 1 shows the computational cost for the main operations, proofs, and verification at each gate.

MiP-2 needs further costs in the decryption phase: each server pays $1+N$ for decryption, N for proving correct decryption, and $(m-1)(1+1.2N)$ for verifying the proof issued by other servers. The number of switching gates in MiP-2 is $(N\log_2 N - N + 1)(t+1)$. MiP-1 uses $(N\log_2 N - N + 1)(t+1) - (N/2-1)$ switching gates, $N/2-1$ fixed gates, and $N/2-1$ output gates.

According to our estimation, MiP-1 is faster than the DL-based scheme of [12] for $N < 2^{40}$ with $\kappa = 80$ and $m = 5$, which is the same setting as used in [1].

Table 1. Computational cost for a gate.

		main operation	proof	verify
MiP-2	switching gate	4	9.2	10.4
MiP-1	switching gate	4.4	9.52	10.24
	fixed gate	2	2	2.5
	output gate	2	4.5	5

With the same parameters, MiP-2 is faster for $N < 2^{25}$. For instance, with $N = 2^{12}$ and $N = 2^7$, MiP-1 is 3.6 and 6.7 times faster than [12] respectively. In the same setting, MiP-2 is 2.2 and 4.0 times faster than [12].

Efficiency can be increased by employing pre-computation for all fixed-base modular exponentiation with random exponents. Furthermore, the probabilistic verification used in P_2-2 improves efficiency if it is used in PS_1-4 and PS_2-4.

8 Concluding Remarks

The major difference between the constructions, MiP-1 and MiP-2, is that each server in MiP-2 verifies all results by itself while servers in MiP-1 do not unless triggered by others. Accordingly, MiP-2 will be used in applications where the servers eventually use the output for further computation. On the other hand, MiP-1 suits applications where mix servers work as a gateway and the resulting plaintexts are used by other entities.

Another considerable difference is their recovery from disruption. If a deviating server is detected in MiP-2, other severs simply skip his results while servers in MiP-1 must cooperate to recover decryption keys and perform decryption in public each time a deviating server is detected. Accordingly, MiP-1 suffers more round/communication/computation complexity for recovery than MiP-2. However, this disadvantage can be eliminated as follows: Suppose a server, which hosts gates in the j-th column, finds faults among the output of gates in the $(j-1)$-th column. The server then takes the output from the $(j-2)$-th column as its input and uses $y_{j-1}\hat{y}_{j+1}$ for randomization (in Equation 3 and 4) instead of $\hat{y}_{j+1}$. This, intuitively, corresponds to putting off the decryption process for j-th column to the end of the Mix-net. Corresponding proofs can be generated appropriately. After M_m outputs the result, V requests all servers to reveal pieces of x_{j-1} so that the verifier can execute the skipped decryption process by reconstructing x_{j-1}.

Acknowledgements

The author wishes to thank Kazue Sako and Tatusaki Okamoto for their helpful suggestions on the definitions in Section 2. Constructive discussions with A. Fujioka, F. Hoshino, and M. Michels are also acknowledged. Finally the author

thank the anonymous refrees for comments which improved presentation of the paper.

References

1. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *Advances in Cryptology — EUROCRYPT '98*, LNCS 1403, pages 437–447. Springer-Verlag, 1998. 258, 260, 271
2. M. Abe. Robust threshold Cramer-Shoup cryptosystem. *Proc. of the 1999 Symposium on Cryptography and Information Security*, T1-1.3, 1999. (in Japanese). 263
3. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology — EUROCRYPT '98*, LNCS 1403, pages 236–250. Springer-Verlag, 1998. 266
4. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — EUROCRYPT '99*, LNCS 1592, pages 90–106. Springer-Verlag, 1999. 263
5. D. L. Chaum. Untraceable electronic mail, return address, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981. 258
6. D. L. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92*, LNCS 740, pages 89–105. Springer-Verlag, 1993. 261
7. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, LNCS 839, pages 174–187. Springer-Verlag, 1994. 261
8. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, LNCS 263, pages 186–199. Springer-Verlag, 1986. 264
9. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology — EUROCRYPT '99*, LNCS 1592, pages 259–310. Springer-Verlag, 1999. 263
10. M. Jakobsson. A practical mix. In *Advances in Cryptology — EUROCRYPT '98*, LNCS 1403, pages 448–461. Springer-Verlag, 1998. 258
11. M. Jakobsson. Flash mixing. *PODC99*, 1999. 258
12. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *ICICS98*, LNCS 1334, pages 440–444. Springer-Verlag, 1998. 258, 259, 260, 271, 272
13. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology — EUROCRYPT '93*, LNCS 765, pages 248–259. Springer-Verlag, 1994. 258, 268
14. T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology — EUROCRYPT '91*, pages 522–526. Springer-Verlag, 1991. 263
15. K. Sako and J. Kilian. Receipt-free mix-type voting scheme — a practical solution to the implementation of a voting booth —. In *Advances in Cryptology — EUROCRYPT '95*, LNCS 921, pages 393–403. Springer-Verlag, 1995. 258, 266
16. Y. Tsiounis and M. Yung. On the security of El Gamal based encryption. In *First International Workshop on Practice and Theory in Public Key Cryptography – PKC '98*, LNCS 1431, pages 117–134. Springer-Verlag, 1998. 263, 266
17. A. Waksman. A permutation network. *Journal of the Association for Computing Machinery*, 15(1):159–163, January 1968. 261