Attack for Flash MIX

Masashi Mitomo* and Kaoru Kurosawa

Tokyo Institute of Technology,
2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan
mitomo@flab.fujitsu.co.jp, kurosawa@ss.titech.ac.jp

Abstract. A MIX net takes a list of ciphertexts $(c_1, \cdots, c_N)$ and outputs a permuted list of the plaintexts $(m_1, \cdots, m_N)$ without revealing the relationship between $(c_1, \cdots, c_N)$ and $(m_1, \cdots, m_N)$. This paper shows that the Jakobsson's flash MIX of PODC'99, which was believed to be the most efficient robust MIX net, is broken. The first MIX server can prevent computing the correct output with probability 1 in our attack. We also present a countermeasure for our attack.

1 Introduction

A MIX net takes a list of ciphertexts $(c_1, \cdots, c_N)$ of users $1, \cdots, N$ and outputs a permuted list of the plaintexts $(m_1, \cdots, m_N)$ without revealing the relationship between $(c_1, \cdots, c_N)$ and $(m_1, \cdots, m_N)$. MIX nets have found many applications in anonymous communication [4], election schemes [4, 7, 13, 15] and payment systems [9].

The original MIX net was proposed by Chaum [4]. B.Pfitzmann and A.Pfitzmann, however, showed an attack by a sender, which is more complicated than a simple repeated ciphertext attack [14].

Another problem of Chaum's MIX net, based on RSA, is that the size of each ciphertext c_i is very long proportionally to the number of MIX servers v. Park et al. overcame this problem by using ElGamal encryption scheme so that the size of each c_i became independent of v [13]. Almost all MIX nets proposed after this paper are based on ElGamal encryption scheme.

A general method to achieve verifiability is to have each MIX server to prove that he behaved correctly in zero knowledge. Sako and Kilian [15] showed such an efficient proof system for Park et al.'s MIX net. This scheme is the first *universally verifiable* MIX net.

On the other hand, Ogata et al. showed the first *robust* MIX net which is also universally verifiable [12]. In this scheme, the computational cost of each MIX server is $O(\kappa t N)$ and the external verifier's cost is also $O(\kappa t N)$, where κ is the security parameter and t denotes the number of malicious MIX servers.

At Eurocrypt'98, Abe showed a robust MIX net in which the external verifier's cost is reduced to $O(\kappa N)$ [1]. At the same time, Jakobsson showed a more efficient robust MIX net, called practical MIX [8] (but not universally verifiable).

* He is currently working for Fujitsu Laboratories Ltd.

Instead of cut and choose methods, he introduced a method of so called repetition robustness. However, this scheme was recently broken by Desmedt and Kurosawa (DK attack) [5].

At PODC'99, Jakobsson proposed his second robust MIX net, called flash MIX [10]. This scheme is the most efficient robust MIX net known so far which satisfies $v = O(t)$, where v is the number of MIX servers. (The MIX net recently proposed by [5] requires $v = O(t^2)$.) In flash MIX, the computational cost of each MIX server is only $O(tN)$. The DK attack [5] for practical MIX [8] does not work for flash MIX directly because two dummy elements are inserted into the input list at the beginning of the protocol in flash MIX. Actually, Jakobsson *proved* the security of flash MIX in [10, Theorem 1 and Theorem 2].

In this paper, however, we show that flash MIX is broken. In our attack, the first MIX server can prevent computing the correct output with probability 1. This means that his security proof is wrong. Our attack is a variant of the DK attack for practical MIX [5]. We also present a countermeasure for our attack. It will be a further work to study about the security of our countermeasure.

Flash MIX consists of the first re-encryption phase, the second re-encryption phase and the unblinding protocol in which each MIX server proves that he behaved correctly in the first and the second re-encryption phases. Now our malicious first MIX server executes the first re-encryption phase honestly, but cheats in the second re-encryption phase. He computes his invalid output lists from not only his input lists of the second re-encryption phase but also the input to the flash MIX itself so that no cheating is detected in the unblinding protocol.

Other related works. Abe showed MIX nets which are efficient for small N [2, 3]. In Abe's MIX nets, the cost of each MIX server is $O(tN \log N)$. Jakobsson and Juels showed a MIX net which has the same advantage in [11]. In their MIX net, the cost of each MIX server is $O(tN \log^2 N)$. Since these complexities grow faster in N than the other schemes, these schemes suit small N.

On the other hand, Desmedt and Kurosawa showed an MIX net in which the cost of each MIX server is only $O(N)$ while $v = O(t^2)$ [5].

2 Model of MIX net

2.1 Model and definitions

In the model of MIX nets, there exist three types of participants: *users*, a *bulletin board*, and the *MIX servers*.

1. The *users* post encrypted messages $(c_1, \cdots, c_N)$ to the *bulletin board*.
2. After the bulletin board fills up, or after some other triggering event occurs, the *mix servers* compute a randomly permuted list of decryptions $(m_1, \cdots, m_N)$ of all valid encryptions posted on the bulletin board.

MIX nets must satisfy privacy, verifiability and robustness. Suppose that at most t among v MIX servers and at most $N-2$ among N senders are malicious. Then we say that a MIX net satisfies :

- t-*privacy* if the relationship between $(c_1, \cdots, c_N)$ and $(m_1, \cdots, m_N)$ is kept secret.
- t-*verifiability* if an incorrect output of the MIX net is detected with overwhelming probability.
- t-*robustness* if it can output $(m_1, \cdots, m_N)$ correctly with overwhelming probability.

We say that a MIX net is t-*resilient* if it satisfies t-privacy, t-verifiability and t-robustness.

2.2 ElGamal based encryption scheme for users

Let p be a safe prime, i.e., p, q be primes such that $p = 2q + 1$, and g be a generator of G_q. Let $y = g^x \bmod p$, where x is a secret key. The public key of ElGamal encryption scheme is (p, q, g, y).

To encrypt a value $m \in G_q$, a random number $r \in_u Z_q$ is chosen and the ciphertext $(a, b) = (g^r, my^r)$ is calculated. For decryption, $m = b/a^x$ is calculated. (To guarantee that $m \in G_q$, we should let $m = (M \mid p)M$ for an original message $M \in [1 \ldots (p-1)/2]$, where $(M \mid p)$ is the Jacobi symbol of M.)

The MIX servers share a secret key x using a $(t+1, v)$ threshold scheme [16], where v denotes the number of MIX servers.

3 Flash MIX

Jakobsson proposed his second t-resilient MIX net, called flash MIX, at PODC'99 [10]. (His first t-resilient MIX net [8] was broken [5].) This scheme is the most efficient robust MIX net known so far which satisfies $v = O(t)$. (The MIX net recently proposed by [5] requires $v = O(t^2)$.) In flash MIX, the computational cost of each MIX server is only $O(tN)$.

For (a, b), let

$$(c, d) = (ag^\beta, by^\beta).$$

We say that (c, d) is a re-encryption of (a, b) and β is the re-encryption exponent. For (a_1, b_1) and (a_2, b_2), we say that $(a_1 a_2, b_1 b_2)$ is the product of (a_1, b_1) and (a_2, b_2).

3.1 Functionality

The input to flash MIX is a list of ciphertexts

$$((a_1, b_1), \cdots, (a_N, b_N)),$$

where (a_i, b_i) is an ElGamal encryption of a message m_i with respect to the public key (p, q, g, y). The output is a random permutation of

$$((a'_1, b'_1), \cdots, (a'_N, b'_N)),$$

where

$$(a_i^{'}, b_i^{'}) = (a_i g^{r_i}, b_i y^{r_i})$$

is a random re-encryption of (a_i, b_i). $(a_i^{'}, b_i^{'})$ can later be decrypted by a $(t+1, v)$-threshold decryption scheme.

Flash MIX starts with $t+1$ MIX servers, say, MIX servers $1, \cdots t+1$. If cheating is detected during any step of the protocol, then a cheater detection phase commences. In the cheater detection phase, a cheater is detected and replaced. Afterwards, the protocol is restarted.

It consists of two subprotocols, the blinding protocol and the unblinding protocol.

3.2 Blinding protocol

Flash MIX first executes the blinding protocol as follows.

(1) Generation and insertion of dummies.
Two dummies (a_{N+1}, b_{N+1}) and (a_{N+2}, b_{N+2}) are constructed collectively by all MIX servers such that $a_{N+1}, b_{N+1}, a_{N+2}$ and b_{N+2} are random elements of G_q. Let

$$L_0 = ((a_1, b_1), \cdots, (a_{N+2}, b_{N+2})). \tag{1}$$

(2) Duplication.
$\tau \geq 2$ copies of L_0 are created, where

$$\tau = 1 - \frac{\log_2 \epsilon}{2\log_2 N} \tag{2}$$

for ϵ which denotes the maximum failure probability. They are denoted by $L_{1,0}, L_{2,0}, \cdots, L_{\tau,0}$.

(3) First re-encryption.
For $j = 1, 2, \cdots, t+1$, MIX server j takes as input the lists

$$L_{1,(j-1)}, L_{2,(j-1)}, \cdots, L_{\tau,(j-1)}.$$

He re-encrypts each element of each lists given to him, and forwards random permutations of the resulting lists to the next server. His output lists are denoted by

$$L_{1,j}, \cdots, L_{\tau,j}.$$

The final result of this step is denoted by

$$L_{1,0}^{'} \triangleq L_{1,t+1}, \cdots, L_{\tau,0}^{'} \triangleq L_{\tau,t+1}.$$

Since at least one of the $t+1$ MIX servers is assumed to be honest, they are randomly re-encrypted and permuted lists of L_0.

(4) Second re-encryption.
The $t+1$ MIX servers execute similar re-encryption on input $L_{1,0}^{'}, \cdots, L_{\tau,0}^{'}$. The input lists of MIX server 1 are $L_{1,0}^{'}, \cdots, L_{\tau,0}^{'}$ and the output lists are denoted by $L_{1,1}^{'}, \cdots, L_{\tau,1}^{'}$. The input lists of MIX server $j \geq 2$ are denoted by $L_{1,(j-1)}^{'}, \cdots, L_{\tau,(j-1)}^{'}$ and the output lists are denoted by $L_{1,j}^{'}, \cdots, L_{\tau,j}^{'}$.

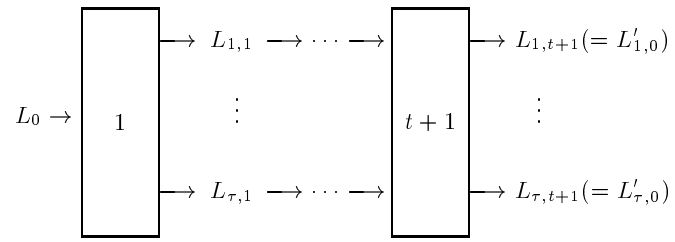

Fig. 1. First re-encryption

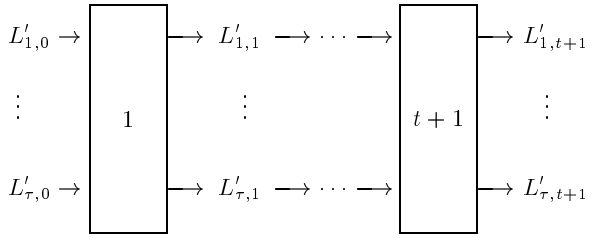

Fig. 2. Second re-encryption

3.3 Unblinding protocol

After the blinding protocol, the unblinding protocol is executed in which each MIX server proves that he behaved honestly in the blinding protocol.

(1) Verifying the first re-encryption.
Each MIX server reveals the re-encryption exponents and the random permutation which he used in the first re-encryption. They are checked by the other MIX servers.

(2) Aggregation.
After the above step, everyone can compute the aggregate permutations $\Pi_1, \cdots, \Pi_\tau$ and the aggregate re-encryption exponents $\beta_{i,j}$ of the first re-encryption such that

$$\begin{aligned} L'_{1,0} &= \Pi_1 \left((a_1 g^{\beta_{1,1}}, b_1 y^{\beta_{1,1}}), \cdots, (a_{N+2} g^{\beta_{1,N+2}}, b_{N+2} y^{\beta_{1,N+2}}) \right), \\ &\vdots \\ L'_{\tau,0} &= \Pi_\tau \left((a_1 g^{\beta_{\tau,1}}, b_1 y^{\beta_{\tau,1}}), \cdots, (a_{N+2} g^{\beta_{\tau,N+2}}, b_{N+2} y^{\beta_{\tau,N+2}}) \right) \end{aligned} \tag{3}$$

(3) Verification of dummy values.
In this phase, each MIX server proves that he behaved honestly about the two dummies in the second re-encryption.

(3.1) MIX server 1 publishes how he permuted the two dummies in $L'_{1,1}, \cdots, L'_{\tau,1}$. (Note that after the verifying the first re-encryption phase, he knows the positions of the two dummies in $L'_{1,0}, \cdots, L'_{\tau,0}$.)

Next, he reveals the re-encryption exponent he used for the second dummy. He also proves that he knows the re-encryption exponent he used for the first dummy in zero-knowledge.
They are checked by the other MIX servers.

(3.2) MIX server 2 behaves similarly to MIX server 1. (Note that from Step 3.1, he knows the positions of the two dummies in $L'_{1,1}, \cdots, L'_{\tau,1}$.)

(3.3) MIX server $3, \cdots, t+1$ behave similarly.

(4) Verification of products.
In this phase, each MIX server proves that he behaved honestly about the product of all elements except the second dummy of each list in the second re-encryption.

(4.1) MIX server 1 behaves as follows. For $i = 1, 2, \cdots, \tau$, let

$$(A_i, B_i) \stackrel{\triangle}{=} \text{the product of all elements of } L'_{i,0} \text{ except the second dummy.}$$
$$(C_i, D_i) \stackrel{\triangle}{=} \text{the product of all elements of } L'_{i,1} \text{ except the second dummy.}$$

Then it holds that

$$C_i = A_i g^{\mu_i} \text{ and } D_i = B_i y^{\mu_i} \tag{4}$$

for some μ_i. MIX server 1 publishes such μ_i, for $1 \leq i \leq \tau$. The other MIX servers verify that eq.(4) holds for $1 \leq i \leq \tau$.

(4.2) MIX servers $2, 3, \cdots, t+1$ behave similarly.

(5) Verification of relative sorting.
Each MIX server j proves that $L'_{1,j}$ is a permuted and re-encrypted version of $L'_{i,j}$ for $2 \leq i \leq \tau$ in the second re-encryption.
Let f be a keyed function that can be modelled by a random oracle. For simplicity, we assume that the range and the domain of f are equal but for a negligible fraction of values.

(5.1) MIX server 1 behaves as follows. Let

$$L'_{1,1} = ((a'_1, b'_1), \cdots, (a'_{N+2}, b'_{N+2})),$$
$$L'_{i,1} = ((c'_1, d'_1), \cdots, (c'_{N+2}, d'_{N+2})),$$

Then $L'_{1,1}$ is a permuted and re-encrypted version of $L'_{i,1}$ for $2 \leq i \leq \tau$. That is,

$$L'_{1,1} = \Phi_i \left((c'_1 g^{\gamma_{i,1}}, d'_1 y^{\gamma_{i,1}}), \cdots, (c'_{N+2} g^{\gamma_{i,N+2}}, d'_{N+2} y^{\gamma_{i,N+2}}) \right) \tag{5}$$

for some Φ_i and $\{\gamma_{i,j}\}$. Note that MIX server 1 can compute such Φ_i and $\{\gamma_{i,j}\}$ from Π_i, $\{\beta_{i,j}\}$ of eq.(3) and the random numbers he used in the second re-encryption.

Now MIX server 1 proves that eq.(5) holds by revealing the so called tag lists $T_{1,1}, \cdots, T_{\tau,1}$ and the so called offset lists $E_{2,1}, \cdots, E_{\tau,1}$ such that

$$
\begin{aligned}
T_{1,1} &= (R_1, \cdots, R_{N+2}), \\
T_{i,1} &= \Phi_i^{-1}(R_1, \cdots, R_{N+2}) \text{ for } 2 \leq i \leq \tau, \\
E_{i,1} &= \Phi_i^{-1}(\gamma_{i,1}, \cdots, \gamma_{i,N+2}) \text{ for } 2 \leq i \leq \tau,
\end{aligned}
$$

where $R_1, \cdots, R_{N+2}$ are unique elements in the domain of f. (Revealing the tag lists and the offset lists is almost equivalent to revealing Φ_i and $\{\gamma_{i,j}\}$).
The other MIX servers verify that eq.(5) holds by using the above tag lists and offset lists.

(5.2) Each MIX server $i(\geq 2)$ applies the function f, keyed with a secret and random key, to all the elements of his input tag lists. His tag lists are obtained by applying the permutation he used in the second re-encryption to the above updated lists. He also generates his offset lists by using his input tag lists and input offset lists. He then reveals his tag lists and offset lists. The other MIX servers verify them.

(6) Output of flash MIX.
If no cheater was found, then dummies are removed from the re-encrypted and permuted first list copy, and the resulting list is output.

3.4 Security

The DK attack [5] for practical MIX [8] does not work for flash MIX directly because two dummy elements are inserted into the input list at the beginning of the protocol in flash MIX. Actually, Jakobsson argued the security of flash MIX as follows [10, Proof of Theorem 1].

In order to successfully alter some elements in the final output, the adversary has to alter at least two elements of the re-encrypted and permuted first list copy, none of which are the second dummy. (See Step (4) and (6) of the previous subsection.) In order for this not to be noticed in the step where lists are relatively sorted and compared, the adversary has to select the same two elements from the remaining $\tau - 1$ list copies.

He claimed that this probability was smaller than ϵ. (See eq.(2) for ϵ.) In the next section, however, we show that this claim is not true.

4 Attack for flash MIX

In this section, we show that MIX server 1 can prevent computing the correct output.

4.1 Attack for the Blinding Protocol

In the blinding protocol of flash MIX, our malicious MIX server 1 executes the first re-encryption honestly, but cheats in the second re-encryption. He computes his invalid output lists $L'_{1,1}, \cdots, L'_{\tau,1}$ from not only his input lists $L'_{1,0}, \cdots, L'_{\tau,0}$ of the second re-encryption phase but also the input to the flash MIX itself (L_0 of eq.(1)). See Fig.3 and Fig.4.

Now our malicious MIX server 1 executes the second re-encryption phase as follows.

(1) MIX server 1 first chooses random numbers $\alpha_1, \cdots, \alpha_N$ such that

$$\alpha_1 + \cdots + \alpha_N = 1 \bmod q. \tag{6}$$

(2) For $1 \leq i \leq \tau$, $L'_{i,0}$ is written as follows.

$$L'_{i,0} = \Pi_i \left((a'_{i,1}, b'_{i,1}), \ldots, (a'_{i,N+2}, b'_{i,N+2}) \right)$$

where Π_i is the aggregate permutation,

$$a'_{i,k} \triangleq a_k g^{\beta_{i,k}} \quad , \quad b'_{i,k} \triangleq b_k y^{\beta_{i,k}}. \tag{7}$$

and $\beta_{i,k}$ is the aggregate re-encryption exponent. (See eq.(3).)

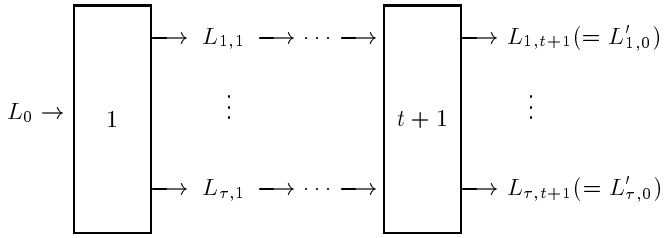

Fig. 3. First re-encryption of our attack

MIX server 1 does not know Π_i. However, note that he can compute the products $a'_{i,1} \cdots a'_{i,N+2}$ and $b'_{i,1} \cdots b'_{i,N+2}$. Now MIX server 1 computes

$$\widetilde{A_i} \triangleq a'_{i,1} \cdots a'_{i,N+2} / a_{N+1} a_{N+2} \quad \text{and} \quad \widetilde{B_i} \triangleq b'_{i,1} \cdots b'_{i,N+2} / b_{N+1} b_{N+2}, \tag{8}$$

where (a_{N+1}, b_{N+1}) and (a_{N+2}, b_{N+2}) are the two dummy elements which are inserted into the input list at the beginning of the protocol (see eq.(1)).
(3) Next for $i = 1, \ldots, \tau$, MIX server 1 publishes

$$\begin{aligned} L'_{i,1} \triangleq \theta_i \Big(& (\widetilde{A_i}^{\alpha_1} g^{t_{i,1}}, \widetilde{B_i}^{\alpha_1} y^{t_{i,1}}), \cdots, (\widetilde{A_i}^{\alpha_N} g^{t_{i,N}}, \widetilde{B_i}^{\alpha_N} y^{t_{i,N}}), \\ & (a_{N+1} g^{t_{i,N+1}}, b_{N+1} y^{t_{i,N+1}}), (a_{N+2} g^{t_{i,N+2}}, b_{N+2} y^{t_{i,N+2}}) \Big) \end{aligned} \tag{9}$$

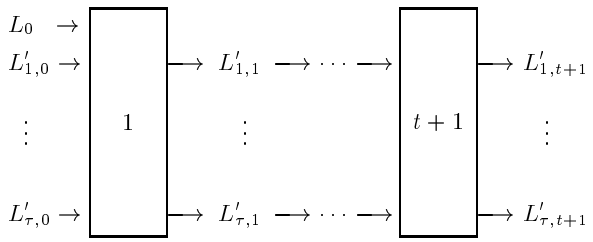

Fig. 4. Second re-encryption of our attack

where θ_i and $t_{i,1}, \cdots, t_{i,N+2}$ are randomly chosen by MIX sever 1.

Let

$$\widetilde{A} \stackrel{\triangle}{=} a_1 \cdots a_N, \quad \widetilde{B} \stackrel{\triangle}{=} b_1 \cdots b_N. \tag{10}$$

Then note that for $i = 1, \ldots, \tau$, $L'_{i,1}$ is a randomly re-encrypted and permuted list of $(\widetilde{a}_1, \widetilde{b}_1), \cdots, (\widetilde{a}_N, \widetilde{b}_N), (a_{N+1}, b_{N+1}), (a_{N+2}, b_{N+2})$, where

$$\begin{aligned}(\widetilde{a}_1, \widetilde{b}_1) &= (\widetilde{A}^{\alpha_1}, \widetilde{B}^{\alpha_1}) \\ &\vdots \\ (\widetilde{a}_N, \widetilde{b}_N) &= (\widetilde{A}^{\alpha_N}, \widetilde{B}^{\alpha_N}).\end{aligned}$$

4.2 Attack for the Unblinding protocol

We next show that MIX server 1 can behave properly in each phase of the unblinding protocol so that no cheating is detected.

Theorem 1. *MIX server* 1 *can behave properly so that the verifying the first re-encryption phase ends successfully.*

Proof. Our MIX server 1 executed the first re-encryption phase honestly. Therefore, he can execute the *verifying* the first re-encryption phase correctly. □

Theorem 2. *MIX server* 1 *can behave properly so that the verification of dummy values phase ends successfully.*

Proof. Everyone knows Π_i of eq.(3). MIX server 1 knows θ_i of eq.(9). Therefore, MIX server 1 knows how the two dummies are permuted from $L'_{i,0}$ to $L'_{i,1}$. Hence, MIX server 1 can publish a description of how the two dummies are permuted from $L'_{i,0}$ to $L'_{i,1}$ for $1 \leq i \leq \tau$.

Next let

$$\begin{aligned}z_{i,N+1} &= t_{i,N+1} - \beta_{i,N+1} \\ z_{i,N+2} &= t_{i,N+2} - \beta_{i,N+2}\end{aligned}$$

where $\beta_{i,j}$ are defined in eq.(3) and $t_{i,j}$ are defined in eq.(9). Then $z_{i,N+1}$ and $z_{i,N+2}$ are the re-encryption exponents of the two dummies from $L'_{i,0}$ to $L'_{i,1}$.

MIX server 1 can compute $z_{i,N+1}$ and $z_{i,N+2}$ because he know $\{t_{i,j}\}$ and $\{\beta_{i,j}\}$. Therefore, he can reveal $z_{i,N+2}$. He also proves that he knows $z_{i,N+1}$ in zero-knowledge.

Then the verification of dummy values phase ends successfully. □

Theorem 3. *MIX server* 1 *can behave properly so that the verification of products phase ends successfully.*

Proof. From eq.(3), it holds that

$$\begin{aligned} A_i &= a_1 \cdots \ a_{N+1} \cdot g^{\beta_{i,1}+\cdots+\beta_{i,N+1}} \\ B_i &= b_1 \cdots \ b_{N+1} \cdot y^{\beta_{i,1}+\cdots+\beta_{i,N+1}} \end{aligned}$$

On the other hand, from eq.(9), it holds that

$$\begin{aligned} C_i &= (\widetilde{A_i}^{\alpha_1} g^{t_{i,1}}) \cdots (\widetilde{A_i}^{\alpha_N} g^{t_{i,N}}) \cdot (a_{N+1} g^{t_{i,N+1}}) \\ &= \widetilde{A_i}^{\alpha_1+\cdots+\alpha_N} \cdot a_{N+1} \cdot g^{t_{i,1}+\cdots+t_{i,N+1}} \\ &= \widetilde{A_i} \cdot a_{N+1} \cdot g^{t_{i,1}+\cdots+t_{i,N+1}} \end{aligned} \tag{11}$$

from eq.(6). Substitute eq.(7) into eq.(8). Then we have

$$\begin{aligned} \widetilde{A_i} &= a'_{i,1} \cdots a'_{i,N+2} / a_{N+1} a_{N+2} \\ &= a_1 g^{\beta_{i,1}} \cdots a_{N+2} g^{\beta_{i,N+2}} / a_{N+1} a_{N+2} \\ &= a_1 \cdots a_N \cdot g^{\beta_{i,1}+\cdots+\beta_{i,N+2}} \end{aligned} \tag{12}$$

Further, substitute eq.(12) into eq.(11). Then we have

$$\begin{aligned} C_i &= a_1 \cdots a_N \cdot g^{\beta_{i,1}+\cdots+\beta_{i,N+2}} \cdot a_{N+1} \cdot g^{t_{i,1}+\cdots+t_{i,N+1}} \\ &= a_1 \cdots a_{N+1} \cdot g^{(\beta_{i,1}+\cdots+\beta_{i,N+2})+(t_{i,1}+\cdots+t_{i,N+1})} \end{aligned}$$

Similarly, we have

$$D_i = b_1 \cdots b_{N+1} \cdot y^{(\beta_{i,1}+\cdots+\beta_{i,N+2})+(t_{i,1}+\cdots+t_{i,N+1})}$$

Now let

$$\mu_i = t_{i,1} + \cdots + t_{i,N+1} + \beta_{i,N+2}. \tag{13}$$

Then it is clear that eq.(4) is satisfied.

MIX server 1 can compute the above μ_i because everyone knows $\beta_{i,N+2}$ and $\{t_{i,j}\}$ is chosen by MIX server 1. Note that $\beta_{i,N+2}$ is computable for everyone by aggregating the re-encryption exponents of the second dummy, which are published in the verification of dummy values. He reveals this μ_i. Then the verification of products phase ends successfully. □

Theorem 4. *MIX server* 1 *can behave properly so that the verification of relative sorting phase ends successfully.*

Proof. Substitute eq.(12) into eq.(9). Then we have

$$L'_{i,1} \stackrel{\triangle}{=} \theta_i \Big((\widetilde{A}^{\alpha_1} g^{t_{i,1}+\alpha_1\cdot(\beta_{i,1}+\cdots+\beta_{i,N+2})}, \widetilde{B}^{\alpha_1} y^{t_{i,1}+\alpha_1\cdot(\beta_{1,1}+\cdots+\beta_{1,N+2})}),$$
$$\vdots$$
$$(\widetilde{A}^{\alpha_N} g^{t_{i,1}+\alpha_N\cdot(\beta_{1,1}+\cdots+\beta_{1,N+2})}, \widetilde{B}^{\alpha_N} y^{t_{i,1}+\alpha_N\cdot(\beta_{i,1}+\cdots+\beta_{i,N+2})}),$$
$$(a_{N+1} g^{t_{i,N+1}}, b_{N+1} y^{t_{i,N+1}}),$$
$$(a_{N+2} g^{t_{i,N+2}}, b_{N+2} y^{t_{i,N+2}}) \Big)$$

where $\widetilde{A}$ and $\widetilde{B}$ is defined in eq.(10). Let

$$\gamma_{i,1} \stackrel{\triangle}{=} t_{1,1} - t_{i,1} + \alpha_1 \cdot (\beta_{1,1} + \cdots + \beta_{1,N+2}) - \alpha_1 \cdot (\beta_{i,1} + \cdots + \beta_{i,N+2}),$$
$$\vdots$$
$$\gamma_{i,N} \stackrel{\triangle}{=} t_{1,N} - t_{i,N} + \alpha_N \cdot (\beta_{1,1} + \cdots + \beta_{1,N+2}) - \alpha_N \cdot (\beta_{i,1} + \cdots + \beta_{i,N+2}),$$

where $t_{i,j}$ is defined in eq.(9). Note that MIX server 1 can compute $\gamma_{i,1}, \cdots, \gamma_{i,N}$ for $2 \le i \le \tau$.

Now MIX server 1 reveals the tag lists $T_{1,1}, \cdots, T_{\tau,1}$ and the offset lists $E_{2,1}, \cdots, E_{\tau,1}$ such that

$$T_{i,1} = \theta_i(R_1, \cdots, R_{N+2}) \text{ for } 1 \le i \le \tau,$$
$$E_{i,1} = \theta_i(\gamma_{i,1}, \cdots, \gamma_{i,N+2}) \text{ for } 2 \le i \le \tau,$$

where θ_i is defined in eq.(9) and $R_1, \cdots, R_{N+2}$ are unique elements in the domain of f. It is easy to see that eq.(5) is satisfied with $\Phi_i = \theta_1 \theta_i^{-1}$.

Therefore, the verification of relative sorting phase ends successfully. □

Theorem 1 $\sim$ 4 show that each phase of the unblinding protocol ends successfully and no cheating is detected.

4.3 Output of flash MIX

Let the input to flash MIX be a list of ciphertexts

$$((a_1, b_1), \cdots, (a_N, b_N)),$$

where (a_i, b_i) is an ElGamal encryption of a message m_i with respect to the public key (p, q, g, y).

Then after threshold decryption, flash MIX must output a random permutation of

$$(m_1, \cdots, m_N).$$

However, in our attack, flash MIX outputs

$$((m_1 \cdots m_N)^{\alpha_1}, \ \cdots, \ (m_1 \cdots m_N)^{\alpha_N})$$

which is clearly different from $(m_1, \cdots, m_N)$. Therefore, flash MIX does not compute the correct output without being detected.

5 Countermeasure

In this section, we show a countermeasure for our attack. The blinding protocol is unchanged. The new unblinding protocol is as follows.

(1) Open dummies of the first re-encryption.
Each MIX server publishes how he permuted the two dummies in the first re-encryption. He next proves that he knows the re-encryption exponents of the two dummies in zero-knowledge.
(2) Verification of dummy values in the second re-encryption.
Unchanged.
(3) Verification of products in the second re-encryption.
Unchanged.
(4) Verifying the first re-encryption.
Unchanged.
(5) Aggregation.
(6) Verification of relative sorting in the second re-encryption.
Unchanged.

Note that

1. (1) is newly introduced. In (1), the re-encryption exponent of the second dummy is not revealed.
2. (4) was put at the beginning of the unblinding protocol in the original scheme.

Then our attack does not work. Theorem 1, 2 and 4 hold. However, Theorem 3 does not hold.

It will be a further work to study about the security of our countermeasure.

References

1. M. Abe, "Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-centers," Eurocrypt '98, pp. 437–447.
2. M. Abe, "A Mix-Network on permutation networks," ISEC Technical report 99-10 (in Japanese) (May, 1999)
3. M. Abe, "Mix-Networks on permutation networks," Asiacrypt '99, pp. 258–273.
4. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, ACM 1981, pp. 84-88 "Undeniable Signatures,"
5. Y.Desmedt and K.Kurosawa, "How to break a practical MIX and design a new one", Eurocrypt'2000.
6. T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," Crypto '84, pp. 10-18
7. A. Fujioka, T. Okamoto and K. Ohta, "A practical secret voting scheme for large scale elections," Auscrypt '92, pp. 244-251
8. M. Jakobsson, "A practical MIX," Eurocrypt '98, pp. 448–461.

9. M. Jakobsson and D. M'Raihi, "Mix-based Electronic Payments," SAC'98, pp. 157–173.
10. M. Jakobsson, "Flash Mixing," PODC'99, pp. 83–89.
11. M. Jakobsson and A. Juels "Millimix: Mixing in small batches," DIMACS Technical report 99-33 (June 1999)
12. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, "Fault Tolerant Anonymous Channel," ICICS '97, pp. 440-444
13. C. Park, K. Itoh, K. Kurosawa, "All/nothing election scheme and anonymous channel," Eurocrypt '93, pp. 248-259
14. B. Pfitzmann and A. Pfitzmann. "How to break the direct RSA-implementation of MIXes," Eurocrypt '89, pp. 373-381
15. K. Sako, J. Kilian, "Receipt-Free Mix-Type Voting Scheme," Eurocrypt '95, pp. 393-403
16. A. Shamir, "How to Share a Secret," Communications of the ACM, Vol. 22, 1979, pp. 612-613