Breaking and Mending Resilient Mix-nets

Lan Nguyen and Rei Safavi-Naini

School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
email: [ldn01,rei]@uow.edu.au

Abstract. In this paper we show two attacks against universally resilient mix-nets. The first attack can be used against a number of mix-nets, including Furukawa-Sako01 [6], Millimix [11], Abe98 [1], MiP-1, MiP-2 [2, 3] and Neff01 [19]. We give the details of the attack in the case of Furukawa-Sako01 mix-net. The second attack breaks the correctness of Millimix [11]. We show how to counter these attacks, and give efficiency and security analysis for the proposed countermeasures.

1 Introduction

Mix-net [4] is a cryptographic technique that is used to hide the origin of messages in network communication. It can be used in a wide range of applications, including anonymous email [4], Web browsing [7], electronic voting [22, 14], anonymous payment systems [9], secure multiparty computation [12] and privacy in advertisements [15]. A mix-net consists of a set of mix servers, each receiving as input a list of ciphertexts and outputting either a permuted list of the re-encrypted ciphertexts, or a permuted list of the corresponding plaintexts. By keeping the permutation secret, the mix-net can hide the correspondence between input items and output items hence providing *privacy* for the originators of messages. Other important properties of mix-nets are *robustness* and *verifiability.* Robustness means that the mix-net is able to operate correctly regardless of component failure. Verifiability means that the correctness of mix-net operation can be verified by any system participant. A mix-net that provides privacy, robustness and verifiability is called *resilient* [5].

A common way of proving correctness of the results is that each mix-server after producing the output, performs a *verification protocol.* If the verification results in *accept*, the mix-server is assumed *honest* and its output is taken to the next mix-server. If the verification outputs *reject*, the mix-server is found *dishonest* and is expelled from the mix-net and its input is passed to the next mix-server. Many mix-nets, including those given in [11, 2, 3, 6, 19] use zero-knowledge proofs based on the difficulty of *discrete logarithm problem* to perform their verification protocol.

In this paper we show that two mix-nets, proposed by Furukawa et al. [6] and Jakobsson et al. [11], are not resilient. We show that in both cases, despite provable security, a mix-server can produce incorrect output without being detected.

One of the attacks can also be used against other mix-nets, including mixing phase in MiP-1, MiP-2 [2, 3] and Neff01 [19] and decryption phase in Abe98 [1] and MiP-2 [2, 3].

The organization of the paper is as follows. In section 2, we recall cryptographic tools and systems that will be used in the rest of the paper. Section 3 gives brief descriptions of Furukawa-Sako01 and Millimix mix-nets. The next two sections show attacks, countermeasures and analysis for Furukawa-Sako01 and Millimix mix-nets. Section 6 concludes the paper.

2 Background

2.1 Model

A mix-net consists of the following participants that are all assumed polynomially bounded. *Users* send messages to mix-net. *Mix servers* perform mixing of then input messages and produce an output, which could be used as input to other mix-servers. Verification can be *external* where a trusted *verifier* verifies operation of the mix-net, or *internal* where each mix server is verified by other mix servers in the same mix-net. We assume there is a *bulletin board* which is a shared memory where all participants have read access to and can append messages after being authenticated. A bulletin board simulates an authenticated broadcast channel.

An *adversary* is a party whose objective is to compromise resiliency of the mix-net. An adversary that can corrupt t_u users and t_s mix servers is called a (t_u, t_s) adversary. Corruption is before the system starts operation (*static adversary*).

2.2 Requirements

To define resiliency we follow the definitions in [5].

- *privacy:* it is infeasible for the adversary to output a pair of input and the corresponding output of an honest user with probability significantly greater than random guess.
- *verifiability:* if a set of participating mix servers produce an output different from the one prescribed by the protocol, then the verification will be able to establish this fact and reveal the identities of the cheating servers. If verification only uses publicly available information of the mix-net, the mix-net is called *universally verifiable.*
- *robustness:* ensures that the probability of producing incorrect output is negligibly less than 1.
- *efficiency* is measured in terms of the computation and communication costs of participants.

A mix-net is *resilient* if it satisfies *privacy*, *robustness* and *verifiability.* A resilient mix-net is *universally resilient* if it is universally verifiable. It is *optimally* *resilient* if t_u and t_s have their maximum possible values, that is t_u is equal $n-2$ where n is he number of users, and t_s is equal to $\lfloor (s-1)/2 \rfloor$ in case of *internal verification* and is equal to $s-1$ in the case of *external verification.*

2.3 Cryptographic tools

El Gamal-Schnorr non-malleable encryption Inputs to a mix-net must be encrypted by a *non-malleable* encryption scheme. In a non-malleable encryption scheme, given a ciphertext it is computationally infeasible to generate a different ciphertext such that the corresponding plaintexts are related in a known manner. If the encryption scheme is malleable, an adversary can trace an input ciphertext ci, by creating a ciphertext ci' whose plaintext is related to the plaintext of ci in a known manner and in the output check for the plaintexts that satisfy the relationship. An example of this attack is shown in [23] against mix-net in [22]. Most of mix-net schemes use a combination of El Gamal encryption and Schnorr signature to efficiently achieve non-malleability. El Gamal-Schnorr non-malleable encryption scheme [2] can be described as follows. Let p and q be two large primes such that $p = 2kq + 1$, where k is a positive integer and g is a generator of a subgroup G_q of order q in Z_p^*. Hereafter, unless stated otherwise we assume all computations are in modulo p. The private key is $x \in Z_q$ and the public key is (y, g) where $y = g^x$. A ciphertext of message $m \in G_q$ is (α, β, c, z) where $\alpha = my^s$, $\beta = g^s$, $c = H(\alpha, \beta, g^w)$, $z = w - cs \bmod q$, H is a hash function $H : \{0,1\}^* \rightarrow 2^{|q|}$ and $s, w \in_R Z_q$ (i.e. chosen randomly and with uniform distribution from Z_q). Validity of a ciphertext can be verified by checking whether $c \overset{?}{=} H(\alpha, \beta, g^z\beta^c)$ and $\alpha, \beta \in G_q$. Intuitively, Schnorr signature is used to show that the ciphertext must have been encrypted by someone with the knowledge of s. The plaintext is computed as $m := \alpha/\beta^x$.

An El Gamal-only ciphertext (α, β) can be *re-encrypted* as another ciphertext $(\alpha \times y^r, \beta \times g^r)$ of the same plaintext m, where *re-encryption exponent* $r \in_R Z_q$.

Schnorr identification Let p,q,x and (y, g) be defined as above. A prover $\mathcal{P}$ can show his knowledge of the private key x to a verifier $\mathcal{V}$ using Schnorr identification protocol as follows.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: a commitment $w = g^e$, where $e \in_R Z_q$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge $c \in_R Z_q$
3. $\mathcal{P} \longrightarrow \mathcal{V}$: a response $s = e + cx \bmod q$

$\mathcal{V}$ then verifies that $g^s = wy^c$. Schnorr identification protocol can be converted into a Schnorr signature scheme by generating $c = H(w, m)$ for a message m that is to be signed using a hash function $H : \{0,1\}^* \rightarrow 2^{|q|}$. Schnorr signature is used in the encryption scheme above.

Disjunctive Schnorr identification Let p and q be defined as above. Suppose $(x_1, (y_1, g_1))$ and $(x_2, (y_2, g_2))$ are two instantiations of $(x, (y, g))$ above. A prover

$\mathcal{P}$ shows he has one of the private keys x_1 or x_2 to a verifier $\mathcal{V}$ by using the *Disjunctive Schnorr identification protocol* as follows. Assume that $\mathcal{P}$ possesses x_1.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: two commitments $w_1 = g_1^{e_1}$, $w_2 = g_2^{s_2} y_2^{-c_2}$, where $e_1, e_2, c_2, s_2 \in_R Z_q$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge $c \in_R Z_q$
3. $\mathcal{P} \longrightarrow \mathcal{V}$: responses $s_1 = e_1 + c_1 x_1 \bmod q$, s_2, $c_1 = c \oplus c_2$, c_2

$\mathcal{V}$ then checks if $g_i^{s_i} = w_i y_i^{c_i}$ for $i \in \{1, 2\}$. Similar to Schnorr identification protocol, Disjunctive Schnorr identification protocol can be converted into a non-interactive form.

Permutation Matrices A matrix $(A_{ij})_{n\times n}$ is a permutation matrix if there exists a permutation ϕ so that $\forall i, j \in \{1, ..., n\}$

$$A_{ij} = \begin{cases} 1 \bmod q \text{ if } \phi(i) = j \\ 0 \bmod q \text{ otherwise} \end{cases}$$

It is proved in [6] that an integer valued matrix $(A_{ij})_{n\times n}$ is a permutation matrix if and only if $\forall i, j, k \in \{1, ..., n\}$

$$\sum_{h=1}^{n} A_{hi} A_{hj} = \begin{cases} 1 \bmod q \text{ if } i = j \\ 0 \bmod q \text{ otherwise} \end{cases} \tag{1}$$

$$\sum_{h=1}^{n} A_{hi} A_{hj} A_{hk} = \begin{cases} 1 \bmod q \text{ if } i = j = k \\ 0 \bmod q \text{ otherwise} \end{cases} \tag{2}$$

Pairwise permutation network Abe [2, 3] used permutations that were constructed from switching gates. A *permutation network* is a circuit, which, on input $(1, ..., n)$ and an arbitrary permutation $\Pi : \{1, ..., n\} \rightarrow \{1, ..., n\}$, outputs $(\Pi(1), ..., \Pi(n))$. A *switching gate* is a permutation network for two input items. A *pairwise permutation network* [25] is a permutation network that is constructed from switching gates and requires $n \log_2 n - n + 1$ switching gates.

3 Mix-nets

In this section we recall two mix-nets that are subjected to the attacks proposed in this paper.

3.1 Furukawa-Sako01 Mix-net

Overview This is one of the two most efficient mix-nets with optimal universal resiliency. Let p, q, private key x and public key (y, g) be set as above, with $p = 2kq + 1$, where k is a positive integer. Input to a mix-server is a set of El

Gamal ciphertexts $\{(g_i, m_i)|i = 1, ..., n\}$ encrypted by the public key (y, g), and so $g_i, m_i \in G_q$, $i = 1, ..., n$. A mix-server uses a permutation ϕ and re-encryption exponents $\{r_i|i = 1, ..., n\}$ to compute its output $\{(g'_i, m'_i)|i = 1, ..., n\}$ as follows:

$$g'_i = g^{r_i} g_{\phi^{-1}(i)}$$
$$m'_i = y^{r_i} m_{\phi^{-1}(i)}$$

To prove the correctness of its operation, the mix-server needs to show the existence of a permutation matrix $(A_{ij})_{n\times n}$ and $\{r_i|i = 1, ..., n\}$ so that:

$$g'_i = g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}} \tag{3}$$

$$m'_i = y^{r_i} \prod_{j=1}^{n} m_j^{A_{ji}} \tag{4}$$

This can be done by a verification protocol below that proves the following statements:

– Given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as equation (3) using $\{r_i\}$ and a matrix that satisfies equation (1).
– Given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as equation (3) using $\{r_i\}$ and a matrix that satisfies equation (2).
– The matrix and $\{r_i\}$ in the above two statements are the same.
– For each pair (g'_i, m'_i), the same r_i and $\{A_{ij}\}$ has been used.

Verification The input is $p, q, g, y, \tilde{g}, \{\tilde{g}_i\}, \{(g_i, m_i)\}, \{(g'_i, m'_i)\}$, $i = 1, ..., n$, where $\{\tilde{g}, \tilde{g_1}, ..., \tilde{g_n}\}$ is a basis generated randomly and independently from the input ciphertexts, so that, under discrete logarithm assumption, it is computationally infeasible to obtain $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} \tilde{g_i}^{a_i} = 1$. The prover is $\mathcal{P}$ and the verifier is $\mathcal{V}$.

1. $\mathcal{P}$ generates: $\delta, \rho, \tau, \alpha, \alpha_i, \lambda, \lambda_i \in_R Z_q, i = 1, ..., n$
2. $\mathcal{P}$ computes:

$$t = g^{\tau}, v = g^{\rho}, w = g^{\delta}, u = g^{\lambda}, u_i = g^{\lambda_i}, i = 1, ..., n$$

$$\tilde{g_i}' = \tilde{g}^{r_i} \prod_{j=1}^{n} \tilde{g_j}^{A_{ji}}, i = 1, ..., n \tag{5}$$

$$\tilde{g}' = \tilde{g}^{\alpha} \prod_{j=1}^{n} \tilde{g_j}^{\alpha_j} \tag{6}$$

$$g' = g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j}$$

$$\begin{aligned}
m' &= y^{\alpha} \prod_{j=1}^{n} m_j^{\alpha_j} \\
\dot{t}_i &= g^{\sum_{j=1}^{n} 3\alpha_j A_{ji} + \tau\lambda_i}, i = 1, ..., n \\
\dot{v}_i &= g^{\sum_{j=1}^{n} 3\alpha_j^2 A_{ji} + \rho r_i}, i = 1, ..., n \\
\dot{v} &= g^{\sum_{j=1}^{n} \alpha_j^3 + \tau\lambda + \rho\alpha} \\
\dot{w}_i &= g^{\sum_{j=1}^{n} 2\alpha_j A_{ji} + \delta r_i}, i = 1, ..., n \\
\dot{w} &= g^{\sum_{j=1}^{n} \alpha_j^2 + \delta\alpha}
\end{aligned}$$

3. $\mathcal{P} \longrightarrow \mathcal{V}$: $t, v, w, u, \{u_i\}, \{\tilde{g_i}'\}, \tilde{g}', g', m', \{\dot{t_i}\}, \{\dot{v_i}\}, \dot{v}, \{\dot{w_i}\}, \dot{w}, i = 1, ..., n$
4. $\mathcal{P} \longleftarrow \mathcal{V}$: challenges $\{c_i | i = 1, ..., n\}$, $c_i \in_U Z_q$
5. $\mathcal{P} \longrightarrow \mathcal{V}$:

$$\begin{aligned}
s &= \sum_{j=1}^{n} r_j c_j + \alpha \\
s_i &= \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q, i = 1, ..., n \\
\lambda' &= \sum_{j=1}^{n} \lambda_j c_j^2 + \delta \bmod q
\end{aligned}$$

6. $\mathcal{V}$ verifies:

$$\tilde{g}^s \prod_{j=1}^{n} \tilde{g_j}^{s_j} = \tilde{g}' \prod_{j=1}^{n} \tilde{g_j}'^{c_j} \tag{7}$$

$$g^s \prod_{j=1}^{n} g_j^{s_j} = g' \prod_{j=1}^{n} g_j'^{c_j} \tag{8}$$

$$y^s \prod_{j=1}^{n} m_j^{s_j} = m' \prod_{j=1}^{n} m_j'^{c_j} \tag{9}$$

$$\begin{aligned}
g^{\lambda'} &= u \prod_{j=1}^{n} u_j^{c_j^2} \\
t^{\lambda'} v^s g^{\sum_{j=1}^{n} (s_j^3 - c_j^3)} &= \dot{v} \prod_{j=1}^{n} \dot{v_j}^{c_j} \dot{t_j}^{c_j^2} \\
w^s g^{\sum_{j=1}^{n} (s_j^2 - c_j^2)} &= \dot{w} \prod_{j=1}^{n} \dot{w_j}^{c_j}
\end{aligned}$$

3.2 Millimix

Overview Millimix is a mix-net for small input batches that provides optimal universal resiliency with internal verification. Millimix uses El Gamal scheme for encryption. The two primes p and q, private key x and public key (y, g) are set up as described above, and $p = 2q + 1$. To satisfy non-malleability, El Gamal-Schnorr non-malleable encryption scheme can be used. Input to the mix-net is a set of ciphertexts encrypted by the public key (y, g). If El Gamal-Schnorr non-malleable encryption scheme is used, and an input ciphertext (α, β, c, z) needs to pass non-malleability test before (α, β) is taken to the first mix-server. Each mix-server, except the first one, takes output of the previous mix-server as its input, and the output of the mix-net is the output of the last mix server.

Each mix server simulates a pairwise permutation network consisting of a number of switching gates. Each switching gate re-encrypts and permutes the two input ciphertexts. The mix server outputs the result of permutation and proves the correctness of each of its switching gate's operations using a verification protocol described in the following section. Once a corrupt mix server is found, it is expelled and its input is passed to the next mix server. If the corrupt mix-server is the last mix-server, its input is posted to the bulletin board as the output of the mix-net.

The system is efficient because for an input batch of n items, each mix server needs $O(nlogn)$ modular exponentiations with low constant to perform the re-encryption and internal verification.

Millimix uses threshold decryption to decrypt the input list of ciphertexts. We omit the details of this as it is not relevant to the attack described below, which breaks the correctness of the system.

Verification The verification is by proving correctness of the output of each switching gate. The input to a switching gate is a pair of ciphertexts (α_1, β_1), (α_2, β_2) of the two plaintexts m_1, m_2 respectively, and the output is a pair of ciphertexts (α'_1, β'_1), (α'_2, β'_2) of the two plaintexts m'_1, m'_2 respectively. The server shows its correctness of the switching gate by proving the following two statements:

– Statement 1: $m_1 m_2 = m'_1 m'_2$ using Plaintext Equivalent Proof (PEP) for ciphertexts $(\alpha_1\alpha_2, \beta_1\beta_2)$ and $(\alpha'_1\alpha'_2, \beta'_1\beta'_2)$.
– Statement 2: $m_1 = m'_1$ OR $m_1 = m'_2$ using DISjunctive Plaintext Equivalent Proof ($DISPEP$)

PEP proves a ciphertext (α', β') is a valid re-encryption of a ciphertext (α, β) encrypted using El Gamal public key (y, g). That is there exists $\gamma \in Z_q$ such that $\alpha = \alpha' y^\gamma$ and $\beta = \beta' g^\gamma$. Jakobsson et al [11] showed that this proof can be obtained using Schnorr identification protocol (or Schnorr signature for non-interactive case) as described below. Assume two ciphertexts (α, β) and (α', β') are given, compute $(y_s, g_s) = ((\alpha/\alpha')^z(\beta/\beta'), y^z g)$. Now if (α, β) and (α', β') are encryptions of the same message, then there exists $\gamma \in Z_q$ such that $(y_s, g_s) = ((y^z g)^\gamma, y^z g)$. The prover (mix-server) uses Schnorr identification protocol to show that it knows γ.

$DISPEP$ proves that a ciphertext (α_1, β_1) represents a re-encryption of one of the two ciphertexts (α'_1, β'_1) and (α'_2, β'_2). $DISPEP$ is implemented by having the prover to perform Disjunctive Schnorr identification protocol. Jakobsson et al. suggested to find $(y_{s1}, g_{s1}) = (\alpha_1/\alpha'_1, \beta_1/\beta'_1)$ and $(y_{s2}, g_{s2}) = (\alpha_1/\alpha'_2, \beta_1/\beta'_2)$ as two valid Schnorr public keys and use Disjunctive Schnorr identification protocol to show knowledge of one of the Schnorr private keys, which is also the El Gamal private key x of the ciphertexts. Therefore, this requires the mix-server to know the El Gamal private key x of the ciphertexts, which is not acceptable. In section 4.2, we show a revised version of this protocol which uses the approach in PEP and removes this problem.

4 Attacks

In this section we propose two attacks that break the resiliency of a number of mix-nets. We describe the first attack on Furukawa-Sako01 scheme and comment on its application to other schemes.

4.1 Attacking Furukawa-Sako01 Scheme

Description It is possible to break correctness of this mix-net with a success chance of at least 50%.

Let a be a generator of Z_p. Then $a^{kq} \neq 1$ and $a^{2kq} = 1$. The mix server modifies one of the output ciphertexts as

$$\begin{aligned} g'_{i_0} &= g^{r_{i_0}} g_{\phi^{-1}(i_0)} \\ m'_{i_0} &= y^{r_{i_0}} m_{\phi^{-1}(i_0)} a^{kq} \end{aligned}$$

where $i_0 \in \{1, ..., n\}$. (g'_{i_0}, m'_{i_0}) is not a valid re-encryption of $(g_{\phi^{-1}(i_0)}, m_{\phi^{-1}(i_0)})$. However, if the challenge c_{i_0} is even, then the verification protocol still accepts the output as correct, as shown below.

The mix server only modifies m'_{i_0} which only affects equation (9) in the verification protocol. If the verifier can successfully verify equation (9), the verification protocol produces incorrect results. Because c_{i_0} is even, $a^{c_{i_0} kq} = 1$. So

$$m'^{c_{i_0}}_{i_0} = (y^{r_{i_0}} m_{\phi^{-1}(i_0)} a^{kq})^{c_{i_0}} = (y^{r_{i_0}} m_{\phi^{-1}(i_0)})^{c_{i_0}}$$

Therefore, equation (9) remains correct.

In another version of this attack, the mix server modifies g'_{i_0} in a similar manner so that the incorrect ciphertext becomes

$$\begin{aligned} g'_{i_0} &= g^{r_{i_0}} g_{\phi^{-1}(i_0)} a^{kq} \\ m'_{i_0} &= y^{r_{i_0}} m_{\phi^{-1}(i_0)} \end{aligned}$$

.

Countermeasure Multiplying $y^{r_{i_0}} m_{\phi^{-1}(i_0)}$ by a^{kq} generates a m'_{i_0} that is not in G_q. The attack can be detected if the verifier checks to see if $g'_i, m'_i \in G_q$, $i = 1, ..., n$. If $k = 1$, it is the same as checking the Legendre symbol of g'_i, m'_i, for which an algorithm can be found in [16] (p. 73). The algorithm requires one extra modular multiplication.

If $k \neq 1$, two extra modular exponentiations are required. So the verification cost at each mix server will increase by $2n$ modular exponentiations, where n is the number of input items.

Security Furukawa-Sako01 protocol has been proved to be complete, sound and zero-knowledge. In the following, we show the effect of the above attack on the proof of soundness and note that completeness and zero-knowledgeness proofs will not be affected by the proposed attack.

The proof of soundness is based on Lemma 1 restated from [6]. We show the short-coming of the original proof of the lemma and how the proposed fix completes the proof.

Lemma 1. *Assume $\mathcal{P}$ knows $\{A_{ij}\}, \{r_i\}, \{\alpha_i\}$ and α satisfying equations (5) and (6), and $\{s_i\}$ and s satisfying equation (7). If equations (8) and (9) hold with non-negligible probability, then either the relationships*

$$\begin{cases} g' &= g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j} \\ g'_i &= g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}}, i = 1, ..., n \\ m' &= y^{\alpha} \prod_{j=1}^{n} m_j^{\alpha_j} \\ m'_i &= y^{r_i} \prod_{j=1}^{n} m_j^{A_{ji}}, i = 1, ..., n \end{cases}$$

hold or $\mathcal{P}$ can generate nontrivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} \tilde{g_i}^{a_i} = 1$ with overwhelming probability.

Proof. Replace $\tilde{g'}$ and $\{\tilde{g'_i}\}$ in equation (7) by those corresponding values in equation (5) and (6), we have the following:

$$\tilde{g}^{\sum_{j=1}^{n} r_j c_j + \alpha - s} \prod_{i=1}^{n} \tilde{g_i}^{\sum_{j=1}^{n} A_{ij} c_j + \alpha_i - s_i} = 1$$

Therefore, either the equations

$$\begin{cases} s = \sum_{j=1}^{n} r_j c_j + \alpha \\ s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \end{cases}$$

hold or $\mathcal{P}$ can generate nontrivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} \tilde{g_i}^{a_i} = 1$ with overwhelming probability.

If the equations hold, replace s and $\{s_i\}$ in equation (8), we have the following equation holds with non-negligible probability

$$1 = b_0 \prod_{i=1}^{n} b_i^{c_i} \tag{10}$$

where

$$b_0 = \frac{g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j}}{g'}$$

$$b_i = \frac{g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}}}{g'_i}, i = 1, ..., n$$

.

At this point, the proof in [6] reached the conclusion that $b_i = 1, i = 0, ..., n$. This conclusion is only correct if $b_i \in G_q, i = 0, ..., n$, as shown below.

Suppose $\mathcal{C}$ is the vector space that is spanned by the set S of all vectors

$$u = (1, c_1, c_2, ..., c_n)$$

such that $c_1, c_2, ..., c_n$ satisfy equation (10). As $b_i \in G_q, i = 0, ..., n$, we can assume $b_i = g^{e_i}, i = 0, ..., n$, where $e_i \in Z_q$. Define the vector $e = (e_0, ..., e_n)$, each vector $u \in S$ satisfies the equation

$$eu = 0$$

If $dim(\mathcal{C}) < n+1$, the size of the set S is at most q^{n-1} and so the probability that equation (10) holds is at most $q^{n-1}/q^n = 1/q$, which is negligible. If $dim(\mathcal{C}) = n + 1$, then $e = 0$ and so $b_i = 1, i = 0, ..., n$.

By repeating the same argument for m' and $\{m'_i\}$, Lemma 1 has been proved.

4.2 Millimix Attack

Description Millimix is vulnerable to two attacks. An attack similar to the one against Furukawa-Sako01 mix-net described above can be applied to Millimix. That is a malicious mix-server can output incorrect ciphertexts without being detected and the success chance is at least 50%. The attack can be prevented using the same proposed countermeasure. That is, at the beginning of the verification protocol, the verifier must verify if $\alpha'_i, \beta'_i \in G_q$, $i = 1, 2$. This is the same as calculating the Legendre symbol of α'_i, β'_i, for which the algorithm described in [16] (p. 73) requires one modular multiplication.

In the following, we describe a second attack and a method of countering the attack. As noted earlier, the original $DISPEP$ in [11] computes $(y_{s1}, g_{s1}) = (\alpha_1/\alpha'_1, \beta_1/\beta'_1)$ and $(y_{s2}, g_{s2}) = (\alpha_1/\alpha'_2, \beta_1/\beta'_2)$ as two Schnorr public keys and shows knowledge of one of the Schnorr private keys using Disjunctive Schnorr identification protocol. However, this requires the mix-server to know the El Gamal private key x of the ciphertexts, which is not acceptable. To remove this problem, we show a corrected version of $DISPEP$, which uses the approach in PEP, and then show that even with this modification, the correctness of the mix-net can be broken.

Modified $DISPEP$**:** $DISPEP$ proves that a ciphertext (α_1, β_1) represents a re-encryption of one of the two ciphertexts (α'_1, β'_1) and (α'_2, β'_2). Compute

$$(y_{s1}, g_{s1}) = ((\alpha_1/\alpha'_1)^{z_1}(\beta_1/\beta'_1), y^{z_1}g)$$
$$(y_{s2}, g_{s2}) = ((\alpha_1/\alpha'_2)^{z_2}(\beta_1/\beta'_2), y^{z_2}g)$$

as two Schnorr public keys. Assume w.l.o.g. that (α_1, β_1) is a re-encryption of (α'_1, β'_1), then there exists $\gamma_1 \in Z_q$ such that $(y_{s1}, g_{s1}) = ((y^{z_1}g)^{\gamma_1}, y^{z_1}g)$. The prover (mix-server) uses Disjunctive Schnorr identification protocol with $(y_{s1}, g_{s1}), (y_{s2}, g_{s2})$ to show that it knows γ_1.

Attack against Verification:
The attack exploits the fact that the exponents z in PEP and z_1, z_2 in $DISPEP$ can be arbitrarily chosen. Let (α_1, β_1) and (α_2, β_2) denote the two input ciphertexts to a switching gate of a malicious mix-server. The server computes its output ciphertexts as follows.

$$\begin{aligned}
(\alpha'_1, \beta'_1) &= (\alpha_1 y^{-r_1 - s_1 z_1} g^{-s_1}, \beta_1 g^{-r_1}) \\
(\alpha'_2, \beta'_2) &= (\alpha_2 y^{-r_2 + s_1 z_1 - sz} g^{s_1 - s}, \beta_2 g^{-r_2})
\end{aligned}$$

Although $(\alpha'_1, \beta'_1), (\alpha'_2, \beta'_2)$ are not valid outputs of the switching gate, but, using PEP and $DISPEP$ the server can still show that: (i) $(\alpha'_1\alpha'_2, \beta'_1\beta'_2)$ is the re-encryption of $(\alpha_1\alpha_2, \beta_1\beta_2)$, and (ii) either (α'_1, β'_1) or (α'_2, β'_2) re-encrypts (α_1, β_1). To show (i), the server computes

$$\begin{aligned}
(\alpha/\alpha', \beta/\beta') &= (\alpha_1\alpha_2/\alpha'_1\alpha'_2, \beta_1\beta_2/\beta'_1\beta'_2) \\
&= (y^{r_1+r_2+sz} g^s, g^{r_1+r_2}) \\
(y_s, g_s) &= ((\alpha/\alpha')^z(\beta/\beta'), y^z g) = ((y^z g)^{r_1+r_2+sz}, y^z g) \\
&= (g_s^{r_1+r_2+sz}, g_s)
\end{aligned}$$

Now Schnorr identification protocol will be performed as follows.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: a commitment $w = g_s^e$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge c
3. $\mathcal{P} \longrightarrow \mathcal{V}$: a response $s = e + c(r_1 + r_2 + sz)$

$\mathcal{V}$ then check if $g_s^s = wy_s^c$. This equation is correct and PEP has been broken.

To show (ii), we note that

$$\begin{aligned}
(y_{s1}, g_{s1}) &= ((\alpha_1/\alpha'_1)^{z_1}(\beta_1/\beta'_1), y^{z_1}g) = ((y^{z_1}g)^{r_1+s_1z_1}, y^{z_1}g) \\
&= (g_{s1}^{r_1+s_1z_1}, g_{s1})
\end{aligned}$$

and so Disjunctive Schnorr identification protocol can be performed as follows.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: two commitments $w_1 = g_{s1}^{e_1}$, $w_2 = g_{s2}^{s_2} y_{s2}^{-c_2}$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge c
3. $\mathcal{P} \longrightarrow \mathcal{V}$: responses $s_1 = e_1 + c_1(r_1 + s_1z_1)$, s_2, $c_1 = c \oplus c_2$, c_2

$\mathcal{V}$ then check if $g_{si}^{s_i} = w_i y_{si}^{c_i}$, $i = 1, 2$. These equations hold, so $DISPEP$ succeeds.

Countermeasures To counter the attack against PEP, z must be either chosen interactively by the verifier after the switching gate has produced the output, or non-interactively calculated as $z = H(\alpha' \parallel \beta' \parallel \alpha \parallel \beta)$, using a hash function $H : \{0,1\}^* \rightarrow 2^{|q|}$. With this modification, the non-interactive version of the protocol will be as follows.

To prove that (α', β') is a re-encryption of (α, β), the prover provides a tuple (z, c, s). A verifier can verify the proof by checking if

$$z \stackrel{?}{=} H(\alpha' \parallel \beta' \parallel \alpha \parallel \beta) \bmod q$$
$$c \stackrel{?}{=} H(g' \parallel y' \parallel g'^s y'^c) \bmod q$$

where $(y', g') = ((\alpha/\alpha')^z(\beta/\beta'), y^z g)$.

$DISPEP$ can be modified in a similar way. That is both z_1 and z_2 must be either chosen by the verifier after the switching gate has produced the output, or computed as $z_1 = z_2 = H(\alpha'_1 \parallel \beta'_1 \parallel \alpha'_2 \parallel \beta'_2 \parallel \alpha_1 \parallel \beta_1 \parallel \alpha_2 \parallel \beta_2)$. The prover then performs Disjunctive Schnorr identification (or signature) protocol, in which the public keys are

$$(y_{s1}, g_{s1}) = ((\alpha_1/\alpha'_1)^{z_1}(\beta_1/\beta'_1), y^{z_1}g)$$
$$(y_{s2}, g_{s2}) = ((\alpha_1/\alpha'_2)^{z_2}(\beta_1/\beta'_2), y^{z_2}g)$$

Security Verification protocol in Millimix has been proved to be complete, sound and honest-verifier zero-knowledge. In the following, we show the effect of the above attack on the proof of soundness and note that completeness and zero-knowledgeness proofs will not be affected by the proposed attack.

The proof of soundness is based on Lemmas 2 and 3 taken from [11]. The proof of the Lemmas have not been given in the original paper and in any case because of the attack, the proofs must be revisited. We show a revised statement and proof of Lemma 2, which shows the importance of choosing z carefully. Lemma 3 can be revised similarly.

Lemma 2. *Let (α, β) and (α', β') be two ciphertexts for which PEP produces* accept *response.*

- *if z is chosen by the prover, then (α', β') is not necessarily a valid re-encryption of (α, β).*
- *if z is chosen by the verifier or computed by hash function as shown above, then either (α', β') is a valid re-encryption of (α, β) or the prover can find the El Gamal private key x.*

Proof. Suppose z is chosen by the prover. If (α', β') and (α, β) have the relationship shown in the attack, then PEP outputs accept whereas (α', β') is not a valid re-encryption of (α, β).

Now let z be chosen by the verifier or computed using a secure hash function. Suppose K is the set of all elements z in Z_q such that the prover knows $o \in Z_q$ satisfying $(\alpha/\alpha')^z(\beta/\beta') = (y^z g)^o$. Let $|K|$ be the number of elements in the set K. If z is chosen randomly by the verifier or computed by the hash function whose output is uniformly distributed over Z_q, the probability that PEP outputs *accept* is $|K|/q$. With sufficiently large q, we can assume $|K| \geq 3$. Otherwise, $|K|/q$ is negligible and so is the success chance of PEP.

So w.l.o.g., assume there exists three distinct elements z_0, z_1 and z_2 in K. Let $\alpha/\alpha' = g^u$ and $\beta/\beta' = g^v$. The prover knows $o_0, o_1, o_2 \in Z_q$ satisfying $(\alpha/\alpha')^{z_i}(\beta/\beta') = (y^{z_i}g)^{o_i}$, $i = 0, 1, 2$ and so has the following system of three linear equations with three unknowns u, v and x:

$$\begin{cases} z_0u + v - o_0z_0x = o_0 \\ z_1u + v - o_1z_1x = o_1 \\ z_2u + v - o_2z_2x = o_2 \end{cases}$$

As $\alpha, \beta, \alpha', \beta' \in G_q$, then u, v, x must exist, and so the system of equations must have a solution. If the solution is unique, the prover will be able to solve it and find the value of x and that demonstrates a knowledge extractor for x.

On the other hand, if the system has more than one solution, the following determinants are equal zero.

$$det = \begin{vmatrix} z_0 & 1 & -o_0z_0 \\ z_1 & 1 & -o_1z_1 \\ z_2 & 1 & -o_2z_2 \end{vmatrix} = 0$$

$$det_x = \begin{vmatrix} z_0 & 1 & -o_0 \\ z_1 & 1 & -o_1 \\ z_2 & 1 & -o_2 \end{vmatrix} = 0$$

This implies that,

$$\begin{aligned} 0 &= det + z_0det_x \\ &= z_0z_2o_0 - z_0z_2o_2 + z_2z_1o_2 - z_2z_1o_1 + z_1z_0o_1 - z_1z_0o_0 \\ &\quad + z_0^2o_2 - z_0^2o_1 - z_0z_1o_2 + z_0z_2o_1 + z_0z_1o_0 - z_0z_2o_0 \\ &= (o_2 - o_1)(z_0 - z_1)(z_0 - z_2) \end{aligned}$$

and so $o_2 = o_1$. This leads to $u = vx$, which means that $\alpha/\alpha' = (\beta/\beta')^x$ and so (α', β') is a valid re-encryption of (α, β).

Lemma 3. *Let (α_1, β_1), (α_1', β_1') and (α_2', β_2') be ciphertexts for which $DISPEP$ produces* accept *response.*

- *if z_1 and z_2 are chosen by the prover, then (α_1, β_1) is not necessarily a valid re-encryption of either (α_1', β_1') or (α_2', β_2').*
- *if z_1 and z_2 are chosen by the verifier or computed by hash function as shown above, then either (α_1, β_1) is a valid re-encryption of either (α_1', β_1') or (α_2', β_2') or the prover can find the El Gamal private key x.*

5 Conclusion

In this paper, we presented attacks against several universally resilient mix-nets and showed countermeasures against these attacks. We also analyzed security and efficiency of the proposed countermeasures. The first attack that is shown against Furukawa-Sako01 mix-net and Millimix can also be used against a number of other mix-nets, more specifically, in breaking proofs of correctness of the mixing phase in MiP-1, MiP-2 [2, 3] and Neff01 [19], and breaking proofs of correctness of the decryption phase in Abe98 [1] and MiP-2 [2, 3]. MiP-1 and MiP-2 are very similar to Millimix and so the first attack can be similarly used. The correctness of Neff01 mix-net relies on the Iterated Logarithm Multiplication Proof protocol that can be easily subjected to the first attack. Abe98 protocol uses threshold El Gamal Decryption and introduces a way of jointly decrypting and proving correctness. It can be shown that the attack is also applicable in this case. The details of these attacks will be provided in the final version of this paper. We note that all these proofs are based on the hardness of discrete logarithm problem. It is conceivable that the attack could have wider implications for a range of proofs that are based on discrete logarithm assumption and so must be carefully considered in all such proofs. The second attack breaks the verification protocol of Millimix. The attack can be countered by carefully choosing the challenge.

References

1. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In K. Nyberg, editor, EUROCRYPT '98, pages 437-447. Springer-Verlag, 1998. LNCS No. 1403.
2. M. Abe. A mix-network on permutation networks. In K.Y. Lam, C. Xing, and E. Okamoto, editors, ASIACRYPT '99, pages 258-273, 1999. LNCS no. 1716.
3. M. Abe and F. Hoshino. Remarks on Mix-Network Based on Permutation Networks. Public Key Cryptography 2001, pages 317-324.
4. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2):84-88, 1981.
5. Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In B. Preneel, editor, EUROCRYPT '00, pages 557-572. Springer-Verlag, 2000. LNCS no. 1807.
6. J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle, pages 368 ff. J. Kilian (Ed.), CRYPTO 2001. LNCS 2139
7. E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized Web browsing simple, secure, and anonymous. In R. Hirschfeld, editor, Financial Cryptography '97, pages 17-31, 1997.
8. M. Jakobsson. A practical mix. In K. Nyberg, editor, EUROCRYPT '98, pages 448-461. Springer-Verlag,1998. LNCS No. 1403.
9. M. Jakobsson and D. M'Raihi. Mix-based electronic payments. In E. Tavares S and H.Meijer, editors, SAC '93, pages 057-473. Springer-Verlag, 1998. LNCS no. 1505.
10. M. Jakobsson. Flash mixing. In PODC '99, pages 83-89. ACM, 1999.
11. M. Jakobsson and A. Juels. Millimix: Mixing in small batches, 1999. DIMACS Technical Report 99-33.

12. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, ASIACRYPT '00, pages 162-177, 2000. LNCS No. 1976.
13. M. Jakobsson, A. Juels, "An Optimally Robust Hybrid Mix Network", PODC '01
14. M. Jakobsson, A. Juels and R. Rivest, "Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking", USENIX Security '02.
15. A. Juels. Targeted advertising and privacy too. In D. Naccache, editor, RSA Conference Cryptographers' Track, 2801.
16. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography. CRC Press 1996.
17. Markus Michels, Patrick Horster: Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme. ASIACRYPT 1996, pages 125-132.
18. M. Mitomo and K. Kurosawa. Attack for flash mix. In T. Okamoto, editor, ASIACRYPT '00, pages 192-204, 2000. LNCS No. 1976.
19. A. Neff, A verifiable secret shuffle and its application to e-voting. In P. Samarati, editor, ACM CCS '01, pages 116-125. ACM Press, 2001.
20. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In Proc. ICICS '97, pages 440-444, 1997. LNCS No. 1334.
21. M. Ohkubo and M. Abe. A length-invariant hybrid mix. In T. Okamoto, editor, ASIACRYPT '00, pages 178-191, 2000. LNCS No. 1976.
22. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In T. Helleseth, editor, EUROCRYPT '93, pages 248-259. Springer-Verlag, 1993. LNCS No. 765.
23. B. Pfitzmann. Breaking an Efficient Anonymous Channel. EUROCRYPT '94, pages 332-340. Springer-Verlag, 1995. LNCS No. 950.
24. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L.C. Guillou and J.-J. Quisquater, editors, EUROCRYPT '95. Springer-Verlag, 1995. LNCS No. 921.
25. A. Waksman. A permutation network. J. ACM, 15(1):159-163, 1968.