How to Break a Practical MIX and Design a New One

Yvo Desmedt[1,2*] and Kaoru Kurosawa[3]

[1] Department of Computer Science, Florida State University
PO Box 4530, 206 Love Building
Tallahassee, FL 32306-4530, USA
desmedt@cs.uwm.edu
[2] Dept. of Mathematics, Royal Holloway,
University of London, UK
[3] Dept. of Electrical and Electronic Engineering,
Faculty of Engineering, Tokyo Institute of Technology
2–12–1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan
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 first shows that the Jakobsson's MIX net of Eurocrypt'98, which was believed to be resilient and very efficient, is broken. We next propose an efficient t-resilient MIX net with $O(t^2)$ servers in which the cost of each MIX server is $O(N)$. Two new concepts are introduced, existential-honesty and limited-open-verification. They will be useful for distributed computation in general.

1 Introduction

1.1 Background

In his extensive work to achieve anonymity, Chaum introduced the concept of a MIX net [6]. MIX nets have found many applications in anonymous communication [6], election schemes [6, 11, 19, 24] and payment systems [14]. A MIX net takes from each user a ciphertext and outputs a permuted list of the plaintexts without revealing who has sent which plaintext, i.e., which plaintext corresponds to which ciphertext. This aspect of a MIX net is also known as *privacy*. Although Pfitzmann-Pfitzmann [21] showed an attack against the RSA implementation of Chaum's MIX scheme, the concept itself was not broken but it was refined. The original MIX net given by Chaum [6] satisfies *privacy* only under the condition that all the senders are honest. To address this issue, one needs *robustness*. A topic that was studied prior to robustness is *verifiability*, to allow to *detect* that

* A part of this research was done while the author visited the Tokyo Institute of Technology, March 4-19, 1999. He was then at the University of Wisconsin – Milwaukee. A part of his research was funded by NSF CCR-9508528.

the output of the MIX net is incorrect. If an outsider can verify this, the scheme is called *universally verifiable*.

Before surveying robustness and verifiability, another problem of Chaum's MIX net based on RSA should be pointed out, which is that the size of each ciphertext c_i is long, i.e., proportional to the total number of MIX servers. Park et al. overcame this problem by using the ElGamal encryption scheme so that the size of each c_i became independent of the number of MIX servers [19]. Almost all MIX nets proposed from then on are based on the ElGamal encryption scheme.

A general method to achieving verifiability is to have each MIX server prove that it behaved correctly in zero knowledge. Sako and Kilian [24] showed such an efficient proof system for Park et al.'s MIX net. The above MIX nets are, however, not *robust*. If at least one MIX server stops, then the entire system stops. Ogata et al. showed the first MIX net satisfying privacy, verifiability and robustness [18]. We call a scheme satisfying all these properties *resilient*.

For comparison, we focus on the random permutation stage of the MIX net because almost all known resilient MIX nets consist of two stages, a random permutation stage and a threshold decryption stage. Also, there are some cases where we want a permutation, but not a decryption. If a MIX net does a permutation only, it is possible that the MIX servers do not need to know the decryption key, which is an advantage.

Now in Ogata et al.'s MIX net, the computational cost of each MIX server is $O(\kappa tN)$, where N is the number of users, κ is the security parameter and t stands for the threshold number of untrusted MIX servers. Subsequently, Abe showed a more efficient resilient MIX net which is also universally verifiable in which the external verifier's cost is reduced to $O(\kappa N)$ [1].

At the same time, Jakobsson showed a very efficient resilient MIX net at Eurocrypt '98 [13] (but not universally verifiable). Later, he showed a more efficient MIX net at PODC'99 [15]. In these schemes, the computational cost of each MIX server is $O(tN)$.

Recently Abe [2, 3] showed his second resilient MIX net which is efficient for a small number of users. In this MIX net, the complexity is $O(tN \log N)$. Jakobsson and Juels showed a MIX net which has the same advantage [16]. 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.

1.2 Our Contribution

This paper first shows that the Jakobsson's first MIX net (presented at Eurocrypt'98) [13], which was believed to be resilient and very efficient, is not robust. We present an attack such that at least one malicious MIX server can prevent computing the correct output. We exploit a homomorphic property of Jakobsson's Eurocrypt '98 scheme to attack it. Observe that we make no claims about other MIX networks, such as the PODC'99 Jakobsson paper [15].

We also propose a new and very efficient resilient MIX net (but it is not universally verifiable). To obtain this scheme, we introduce three new concepts:

Work-sharing-MIX in which (significantly) more MIX servers are being used than one trusts. In threshold schemes a tradeoff is used between reliability and privacy. The motivation of work-sharing-MIX is to have a tradeoff between the number of MIX severs and the computational effort per MIX sever. When N is large (as in national elections), and one wants a sufficiently high security (i.e., a large t), then the computational effort of existing schemes may be prohibitive. We share the computational effort over several machines while maintaining the requirements as privacy, robustness and verifiability.

Existential-honesty divides the MIX servers into blocks of which we can guarantee that one is free of dishonest MIX servers, assuming the number of dishonest MIX servers is bounded by t.

Limited-open-verification is the opposite of zero-knowledge. To prove that a computation has been done correctly the party that did the computation in a block will open the secret it used. However, she will only open this to the members in the same block.

More details are given later on. Those concepts may be useful in other contexts such as secure distributed computation. We achieve 100% robustness in contrast with prior schemes (i.e. the probability of the failure of robustness is 0).

Although the total computational cost of our scheme is comparable to the one of Jakobsson's MIX net of PODC'99 [15] (i.e. $O(t^2N)$), the computational cost of each MIX server is significantly smaller (i.e. $O(N)$) in ours versus the one in Jakobsson's scheme ($O(tN)$). To achieve this we need $O(t^2)$ MIX servers rather than the usual $O(t)$. This introduces several open problems, which we discuss in Sect. 6.

Other details, such as the computational complexity assumptions we need to prove privacy are discussed later on.

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

ElGamal based encryption scheme was commonly used in some of the previous robust MIX nets [18, 1, 13]. 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 is (p, q, g, y).

The MIX servers share a secret key x using a $(t+1, v)$ threshold scheme [27], where v denotes the number of MIX servers.

To encrypt a value $m \in G_q$, a random number $\gamma \in_u Z_q$ is chosen and the ciphertext $(a, b) = (g^\gamma, my^\gamma)$ is calculated. For decryption, $m = b/a^x$ is calculated by a threshold decryption scheme [DF89,20, 12]

As pointed out by Jakobsson, 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.

2.3 Non-malleable ElGamal

Malicious users may post copies or correlated ciphertexts of some encrypted messages of honest users (repeated ciphertext attack). They can then determine (with some probability) what the decryption of the attacked message was, by counting repeats or correlations in the output list. Therefore, it is necessary to use a non-malleable encryption scheme. A public key cryptosystem is said to be non-malleable [9] if there exists no probabilistic polynomial time (p.p.t.) adversary such that given a challenge ciphertext c, he can output a different ciphertext c' such that the plaintexts m, m' for c, c' are meaningfully related. (For example, $m' = m + 1$.)

Tsiounis and Yung [28], and independently Jakobsson [13], showed a non-malleable ElGamal encryption scheme by combining Schnorr's signature scheme [25] with ElGamal encryption scheme under some cryptographic assumption in the random oracle model. Jakobsson used the non-malleable ElGamal encryption scheme in his MIX net for users' encryption to prevent the repeated ciphertext attack [13]. (For a detailed study of the security consult [26].) We also use this scheme in our MIX net of Sect. 4.

3 An Attack for Jakobsson's Practical MIX

In this section, we show how to break the Jakobsson's MIX net of Eurocrypt'98 [13], which was believed to be t-resilient and very efficient.

Jakobsson first showed that a MIX net is obtained by using MIXEXP which takes a list of items $\bar{\mu} = (c_1, \ldots, c_N)$ and robustly computes a permutation $(c_1^{\delta}, \ldots, c_N^{\delta})$. To avoid cut and choose methods, Jakobsson [13] developed a subprotocol in which each MIX server proves that the product of his input elements and the product of his output elements satisfy a certain relation. However, this does not imply proving that each MIX server behaved correctly even if the subprotocol is combined with his other subprotocols. We show an attack such that all the output elements of a MIX server can be affected in a proper way. We also exploit a homomorphic property of his scheme to attack it.

His MIX net is not robust if the MIXEXP is not robust. Therefore, the details of MIXEXP are given in Sect. 3.1. Our attack is given in Sect. 3.2.

If the reader is not interested or is already familiar with his scheme, he can go directly to Sect. 3.2.

3.1 Structure of the Scheme Attacked [13]

Let

$$\Delta_j = g^{\delta_j} \tag{1}$$

be the public information of a MIX server j, where δ_j is his secret. Define

$$\delta \stackrel{\triangle}{=} \prod_{j \in Q} \delta_j, \tag{2}$$

where Q denotes a quorum. MIXEXP takes a list of items $\bar{\mu} = (c_1, \ldots, c_N)$ and robustly computes a permutation $(c_1^{\delta}, \ldots, c_N^{\delta})$.

Jakobsson then showed an efficient implementation of MIXEXP. It consists of four protocols, Blinding I, Blinding II, Unblinding I and Unblinding II. For simplicity, let $Q = \{1, 2, \cdots, t+1\}$. For a list $\bar{\xi} = (d_1, \ldots, d_N)$ and $e \in Z_q$, define

$$\bar{\xi}^e \stackrel{\triangle}{=} (d_1^e, \ldots, d_N^e).$$

Let κ be a security parameter.

Blinding I: (see Fig. 1) For $1 \leq \lambda \leq \kappa$,

1. MIX server 1 chooses a random number $\rho_{I\lambda_1}$ and a random permutation $\pi_{I\lambda_1}$. He then computes

$$\pi_{I\lambda_1}(c_1^{\rho_{I\lambda_1}}, c_2^{\rho_{I\lambda_1}}, \cdots, c_N^{\rho_{I\lambda_1}}).$$

2. MIX server 2 chooses a random number $\rho_{I\lambda_2}$ and a random permutation $\pi_{I\lambda_2}$. He then computes

$$\pi_{I\lambda_2} \circ \pi_{I\lambda_1}(c_1^{\rho_{I\lambda_1}\rho_{I\lambda_2}}, c_2^{\rho_{I\lambda_1}\rho_{I\lambda_2}}, \cdots, c_N^{\rho_{I\lambda_1}\rho_{I\lambda_2}}),$$

and so on.

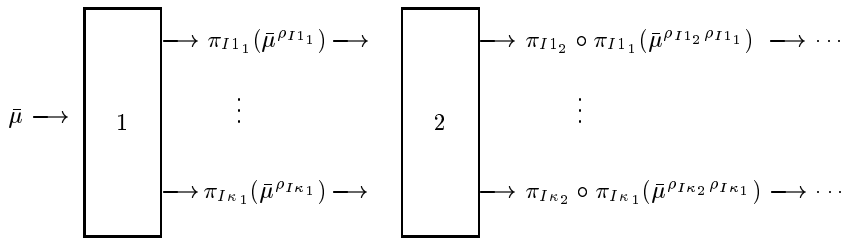


Fig. 1. Blinding I

The final output (in Blinding I) from MIX server $t+1$ is:

$$\bar{\mu}_{I\lambda} \stackrel{\triangle}{=} \pi_{I\lambda}(c_1^{\rho_{I\lambda}}, c_2^{\rho_{I\lambda}}, \cdots, c_N^{\rho_{I\lambda}}), \tag{3}$$

where

$$\pi_{I\lambda} \stackrel{\triangle}{=} \prod_{j \in Q} \pi_{I\lambda j}, \quad \rho_{I\lambda} \stackrel{\triangle}{=} \prod_{j \in Q} \rho_{I\lambda j}.$$

That is, MIXEXP outputs $\bar{\mu}_{I1}, \cdots, \bar{\mu}_{I\kappa}$ on input $\bar{\mu}$ in Blinding I.

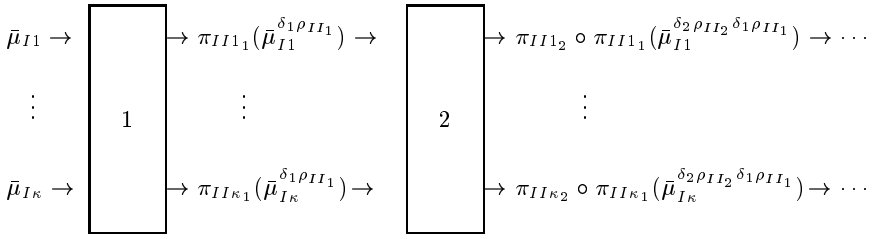


Fig. 2. Blinding II

Blinding II: (see Fig. 2) For $1 \leq \lambda \leq \kappa$,

1. MIX server 1 chooses a random number ρ_{II_1} and a random permutation $\pi_{II\lambda_1}$. He then computes

$$\pi_{II\lambda_1}(\bar{\mu}_{I\lambda}^{\delta_1\rho_{II1}}) = \pi_{II\lambda_1} \circ \pi_{I\lambda}((c_1^{\rho_{I\lambda}})^{\delta_1\rho_{II1}}, (c_2^{\rho_{I\lambda}})^{\delta_1\rho_{II1}},$$

$$\cdots, (c_N^{\rho_{I\lambda}})^{\delta_1\rho_{II1}})$$

from $\bar{\mu}_{I\lambda}$. Note that ρ_{II_1} is independent of λ while $\rho_{I\lambda_1}$ depends on λ.

2. MIX server 2 chooses a random number ρ_{II_2} and a random permutation $\pi_{II\lambda_2}$. He then computes

$$\pi_{II\lambda_2} \circ \pi_{II\lambda_1} \circ \pi_{I\lambda}((c_1^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}}, (c_2^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}},$$

$$\cdots, (c_N^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}})$$

and so on.

The final output (in Blinding II) of MIX server $t+1$ is:

$$\bar{\sigma}_{II\lambda} \stackrel{\triangle}{=} \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\rho_{I\lambda}\delta\rho_{II}}, c_2^{\rho_{I\lambda}\delta\rho_{II}}, \cdots, c_N^{\rho_{I\lambda}\delta\rho_{II}}), \tag{4}$$

where δ is defined in eq. (2) and

$$\pi_{II\lambda} \stackrel{\triangle}{=} \prod_{j\in Q} \pi_{II\lambda j}, \quad \rho_{II} \stackrel{\triangle}{=} \prod_{j\in Q} \rho_{IIj}.$$

That is, MIXEXP outputs $\bar{\sigma}_{II1}, \cdots, \bar{\sigma}_{II\kappa}$ on input $\bar{\mu}_{I1}, \cdots, \bar{\mu}_{I\kappa}$ in Blinding II.

From eq. (4), we see that

$$\bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}}).$$

Note that $(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}})$ of the right hand side is independent of λ. Therefore, $\bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}}$ must be equal for $1 \le \lambda \le \kappa$ if each list $\pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}})$ is sorted. Unblinding I, based on this observation, is described as follows.

Unblinding I:

1. Each MIX server j publishes $\{\rho_{I\lambda j}\}$ for $1 \le \lambda \le \kappa$.
2. Each MIX server computes $\rho_{I\lambda} = \prod_{j\in Q} \rho_{I\lambda j}$ and

$$\bar{\sigma}_{I\lambda} \stackrel{\triangle}{=} \bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}}) \tag{5}$$

   for $1 \le \lambda \le \kappa$.
3. The lists $\bar{\sigma}_{I\lambda}$ with $1 \le \lambda \le \kappa$ are sorted and compared. If they are all equal, and no element is zero, then the result is labeled valid, otherwise invalid.

Next in Blinding II for $\lambda = 1$, let M_j denote the product (modulo p) of all the elements constituting the input to MIX server j. Similarly, S_j denotes the product of all the output elements of MIX server j. Then it must hold that

$$S_j = M_j^{\delta_j\rho_{II_j}}. \tag{6}$$

On the other hand, from eq. (1), we have $\Delta_j^{\rho_{II_j}} = g^{\delta_j\rho_{II_j}}$. Therefore, it holds that

$$S_j = M_j^z \text{ and } \Delta_j^{\rho_{II_j}} = g^z$$

for $z = \delta_j\rho_{II_j}$. Unblinding II, based on this observation, is described as follows.

Unblinding II: (for valid results only)

1. The MIX servers publish $\{\pi_{I1j}\}_{j\in Q}$.
2. The computation of $\bar{\mu}_{I1}$ in "Blinding I" is verified.
3. The MIX servers publish $\{\rho_{IIj}\}_{j\in Q}$.
4. Each MIX server j proves that

$$S_j = M_j^z \text{ and } \Delta_j^{\rho_{II_j}} = g^z. \tag{7}$$

   holds for some z by using one of the methods of [7, 25].
5. The MIX servers compute $\bar{\sigma}_1 = \bar{\sigma}_{I1}^{1/\rho_{II}}$, and output $\bar{\sigma}_1$. Note that $\bar{\sigma}_1$ is a permutation of $(c_1^\delta, \ldots, c_N^\delta)$ from eq. (5).

Jakobsson claims that the final output $\bar{\sigma}_1$ is a permutation of $(c_1^\delta, \ldots, c_N^\delta)$ if the above protocol (MIXEXP) ends successfully.

3.2 Our Attack

We show that Jakobsson's MIXEXP is not robust. This means that his MIX net is not robust. Our attack succeeds if at least one MIX server is malicious.

We exploit a homomorphic property. A dishonest MIX server will first multiply the received inputs. The data is then organized to prevent detection. We now describe the details.

For simplicity, suppose that the last MIX server $t+1$ of Q is malicious. In Blinding II, let her input be $(d_{\lambda 1}, \ldots, d_{\lambda N})$ for $1 \le \lambda \le \kappa$. Let

$$X_\lambda \stackrel{\triangle}{=} d_{\lambda 1} * \cdots * d_{\lambda N}.$$

In our attack, she first chooses random numbers $\alpha_1, \cdots, \alpha_N$ such that

$$\alpha_1 + \alpha_2 + \cdots \alpha_N = 1 \bmod q.$$

Next she outputs

$$\tilde{\sigma}_{II\lambda} = (X_\lambda^{\alpha_1 \delta_{t+1} \rho_{II_{t+1}}}, X_\lambda^{\alpha_2 \delta_{t+1} \rho_{II_{t+1}}}, \ldots, X_\lambda^{\alpha_N \delta_{t+1} \rho_{II_{t+1}}}) \tag{8}$$

for $1 \le \lambda \le \kappa$.

We next show that the MIXEXP ends successfully and our cheating is not detected.

Theorem 1. *The check of Unblinding I is satisfied.*

Proof: In Blinding II, the output of MIX server $t+1$ is $\bar{\sigma}_{II\lambda}$ of eq. (4) if she is honest. Therefore, her input must be

$$\begin{aligned}(d_{\lambda_1}, \ldots, d_{\lambda_N}) &= \pi_{II\lambda_{t+1}}^{-1}(\bar{\sigma}_{II\lambda}^{1/\delta_{t+1}\rho_{II_{t+1}}}) \\ &= \theta_\lambda(c_1^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}}, \ldots, c_N^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}})\end{aligned}$$

for some permutation θ_λ for $1 \leq \lambda \leq \kappa$. Therefore,

$$X_\lambda = d_{\lambda_1} * \cdots * d_{\lambda_N} = (c_1 * \cdots * c_N)^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}} = Z^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}},$$

where $Z \stackrel{\triangle}{=} c_1 * \cdots * c_N$. Then eq. (8) is written as

$$\tilde{\sigma}_{II\lambda} = (Z^{\alpha_1\rho_{I\lambda}\delta\rho_{II}}, \ldots, Z^{\alpha_N\rho_{I\lambda}\delta\rho_{II}}).$$

Finally, at Step 2 of Unblinding I, each MIX server computes

$$\tilde{\sigma}_{I\lambda} = \tilde{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = (Z^{\alpha_1\delta\rho_{II}}, \ldots, Z^{\alpha_N\delta\rho_{II}}) \tag{9}$$

for $1 \leq \lambda \leq \kappa$. Note that $(Z^{\alpha_1\delta\rho_{II}}, \ldots, Z^{\alpha_N\delta\rho_{II}})$ is independent of λ. Therefore, we see that $\tilde{\sigma}_{I1} = \tilde{\sigma}_{I2} = \cdots = \tilde{\sigma}_{I\kappa}$. This means that the check of Unblinding I is satisfied. □

Theorem 2. *The check of Unblinding II is satisfied.*

Proof: Note that $M_{t+1} = X_1$ and S_{t+1} is the product of all the elements of eq. (8) for $\lambda = 1$. Therefore, we have

$$\begin{aligned} S_{t+1} &= X_1^{\alpha_1\delta_{t+1}\rho_{II_{t+1}}} * \cdots * X_1^{\alpha_N\delta_{t+1}\rho_{II_{t+1}}} = (X_1^{\delta_{t+1}\rho_{II_{t+1}}})^{\alpha_1+\cdots+\alpha_N} \\ &= (M_{t+1})^{\delta_{t+1}\rho_{II_{t+1}}}. \end{aligned}$$

Thus, eq. (6) is satisfied. Hence, eq. (7) is satisfied for some z. □

Finally, from eq. (9) and Step 5 of Unblinding II, the output of the MIXEXP becomes as follows.

$$\begin{aligned} \tilde{\sigma}_1 &= \tilde{\sigma}_{I1}^{1/\rho_{II}} \\ &= (Z^{\alpha_1\delta}, \cdots, Z^{\alpha_N\delta}) \\ &= ((c_1 * \cdots * c_N)^{\alpha_1\delta}, \cdots, (c_1 * \cdots * c_N)^{\alpha_N\delta}). \end{aligned}$$

This is clearly different from a permutation of $(c_1^\delta, \ldots, c_N^\delta)$. (See Step 5 of Unblinding II.) Therefore, the MIXEXP does not compute the correct output without being detected.

4 Proposed MIX Net

In this section, we show an efficient t-resilient MIX net by using a certain combinatorial structure over the set of MIX servers.

In Sect. 4.1, we introduce two new concepts, existential-honesty and limited-open-verification. They will be useful for distributed computation in general. We also define a combinatorial structure which guarantees that our scheme is t-resilient.

Our scheme is given in Sect. 4.2 and Sect. 4.3. We further show an efficient construction of the combinatorial structure by using *covering* [29] in Sect. 4.4. *Covering* has recently been used in another cryptographic application: robust secret sharing by Rees et al. [23].

4.1 Existential-Honesty and Limited-Open-Verification

A set system is a pair $(X, \mathcal{B})$, where $X \stackrel{\triangle}{=} \{1, 2, \ldots, v\}$ and $\mathcal{B}$ is a collection of blocks $B_i \subset X$ with $i = 1, 2, \ldots, b$. First, we define (v, b, t)-verifiers set systems.

Definition 1. *We say that $(X, \mathcal{B})$ is a (v, b, t)-verifiers set system if*

1. $|B_i| = t + 1$ *for* $i = 1, 2, \ldots, b$ *and*
2. *for any subset* $F \subset X$ *with* $|F| \leq t$, *there exists a* $B_i \in \mathcal{B}$ *such that* $F \cap B_i = \emptyset$.

Let $(X, \mathcal{B})$ be a (v, b, t)-verifiers set system. We identify X with the set of MIX servers. Therefore, B_i is a subset of MIX servers of size $t+1$. We choose $P_i \in B_i$ arbitrarily for $1 \leq i \leq b$. P_i is called a prover. The other MIX servers of B_i are called verifiers.

We introduce two new concepts in this paper,

- Existential-honesty and
- t-open-verification

which we now describe.

Existential honesty follows from Definition 1. Although, existential honesty is not limited to applications in the MIX context, and may be useful in other distributed computation, we focus on its MIX application. In each block one designated party will mix the ciphertexts. *As long as one block of MIX servers is free of dishonest machines, the goal of mixing has been achieved.* Now we do not know which block satisfies this property. However, Definition 1 guarantees that there always exists one block of honest parties. So, we let *each* block mix the ciphertexts (i.e. the designated party of that block). What do we do when the designated party of a block B_j is dishonest? Since a block has $t+1$ parties, it must be detected. Indeed, there are at most t dishonest parties. If it is detected, then we ignore the output and proceed with the output of block B_{j-1} (or an even earlier one if the designated party in B_{j-1} was dishonest). Now, what happens when one of the verifiers falsely accuses the mixing party of having been dishonest. Then we know that this block is not free of dishonest parties, and therefore the block under consideration is not the one of Definition 1, so we can just ignore the output of the block. In other words, we do not have to decide whether the one who mixed it was honest or not.

We now explain *t-open-verification*. In many secure distributed computation protocols zero-knowledge is used to prove that the computation was done correctly. In our approach we do not need zero-knowledge, the prover will reveal the secrets he used. However, he will only do this to t parties. The existential honesty guarantees that all parties in *at least one* of the blocks of MIX servers will *all* be honest. So, the prover can reveal the secret he used. This speeds up the verification dramatically.

We now formally describe the scheme in full detail.

4.2 Initialization

Let $y(= g^x \bmod p)$ be a public key of the ElGamal scheme as shown in Sect. 2.2. We assume that the secret key x is distributed among v MIX servers by using Shamir's $(t+1, v)$ secret sharing scheme. Actually, we use a robust $(t+1, v)$ threshold ElGamal decryption scheme. (See the end of the next subsection for more details.)

1. Each user i computes a ciphertext $c_i = (a_i, b_i, aux_i, sig_i)$ by the non-malleable ElGamal encryption scheme [28] as shown in Sect. 2.3. That is, $(a_i, b_i) = (g^{\gamma_i}, m_i y^{\gamma_i})$ is the ciphertext of the usual ElGamal scheme, aux_i is the auxiliary information and sig_i is the Schnorr's signature of (a_i, b_i, aux_i) such that a_i is a public key and γ_i is the secret key.
2. Each user i posts his ciphertext c_i to the bulletin board.
3. c_i is discarded if the signature is not valid.

4.3 Main Protocol

We assume that all MIX servers of B_i share a common key e_i for $1 \le i \le b$. We extract (a_i, b_i) from a valid ciphertext c_i. Let

$$A_0 = ((a_1, b_1), \ldots, (a_N, b_N)).$$

We wish to produce a random permutation of the list $(m_1, \ldots, m_N)$, where $m_i = b_i / a_i^x$ is the plaintext of (a_i, b_i). A prover of a block B_i first publishes A_1 which is a randomly permuted list of reencrypted ciphertexts of A_0. He then privately broadcasts the secret random string R_i he used to the verifiers in the same block B_i. Each verifier of B_i checks the validity of A_0 by using R_i.

For $j = 1, \ldots, b$, do:

Step 1. Let

$$A_0 = ((\hat{a}_1, \hat{b}_1), \ldots, (\hat{a}_N, \hat{b}_N)).$$

The prover P_j of block B_j chooses random numbers $s_1, \ldots, s_N$ and a random permutation π_j. She computes

$$A_1 = \pi_j((\hat{a}_1 g^{s_1}, \hat{b}_1 y^{s_1}), \ldots, (\hat{a}_N g^{s_N}, \hat{b}_N y^{s_N})).$$

and then publishes A_1. (A_1 is commonly used for all the verifiers of B_j.)

Step 2. P_j encrypts $s_1, \ldots, s_N$ and π_j by the key e_j of block B_j. Then P_j publishes these ciphertexts. (P_j is broadcasting $s_1, \ldots, s_N$ and π_j secretly to all the verifiers of B_j.)

Step 3. Each verifier of block B_j decrypts the above ciphertexts and checks whether A_1 is computed correctly by using $s_1, \ldots, s_N$ and π_j. He outputs "ACCEPT" if A_1 is computed correctly and "REJECT" otherwise.

Step 4. If some verifier of block B_j outputs "REJECT", then A_1 is *ignored*. Otherwise, let $A_0 := A_1$.

Let the final result be $A_0 = ((\hat{c}_1, \hat{d}_1), \ldots, (\hat{c}_N, \hat{d}_N))$.

Next any $(t+1)$ MIX servers decrypt each $(\hat{c}_i, \hat{d}_i)$ by using a robust $(t+1, v)$ threshold ElGamal decryption scheme. Finally, we obtain a random permutation of the list $(m_1, \ldots, m_N)$.

Gennaro et al. showed a robust threshold RSA signature scheme in [12]. A robust $(t+1, v)$ threshold ElGamal decryption scheme is easily obtained by applying their technique to ElGamal decryption.

4.4 Construction of the Set System

Let $v = (t+1)^2$, $b = t+1$, $X = \{1, 2, \cdots, (t+1)^2\}$ and $B_i = \{(i-1)(t+1)+1, \cdots, i(t+1)\}$ for $1 \leq i \leq b$. Then it is easy to see that $(X, \mathcal{B})$ is a $((t+1)^2, t+1, t)$-verifiers set system.

We next show a more efficient (v, b, t)-verifiers set system. A set system $(X, \mathcal{B})$ is called a (v, k, t)-covering if [29]

1. $|B_i| = k$ for $1 \leq i \leq b$ and
2. every t-subset of X is included in at least one block.

From [17], we have the following proposition. (We learned about proposition 1 from [23].)

Proposition 1. *Suppose that $k = even$ and*

$$3 \leq s \leq \frac{t+3}{2},$$
$$k\left(t - \frac{s-3}{2}\right) \leq v < k\left(t - \frac{s-4}{2}\right).$$

Then there exists a $(v, v-k, t)$-covering such that $b = t+s$. Further, each element $a \in X$ is included in at most two blocks.

See [17, 23] for the construction.

We next borrow the following lemma from [23] in which the lemma was used for robust secret sharing schemes. The proof will be clear.

Lemma 1. *$(X, \mathcal{B})$ is a (v, b, t)-verifiers set system if and only if the set system $(X, \mathcal{B}^c)$ is a $(v, v-t-1, t)$-covering, where $\mathcal{B}^c \stackrel{\triangle}{=} \{X \backslash B_i \mid B_i \in \mathcal{B}\}$.*

Then we obtain a (v, b, t)-verifiers set system as follows.

Corollary 1. *For $t = odd$, there exists a (v, b, t)-verifiers set system such that*

$$v = \frac{3}{4}(t+1)^2 \text{ and } b = \frac{3}{2}(t+1). \tag{10}$$

Further, each element $a \in X$ is included in at most two blocks.

Proof: In Proposition 1, let $k = t + 1$ and $s = (t + 3)/2$. □

The fact that each MIX server is included in at most two blocks is primordial to understand the efficiency analysis described further on.

We show a small example of Corollary 1. Let $t = 3, b = 6, v = 12$ and

$$B_1 = (1, 2, 3, 4), B_2 = (3, 4, 5, 6), B_3 = (5, 6, 1, 2),$$

$$B_4 = (7, 8, 9, 10), B_5 = (9, 10, 11, 12), B_6 = (11, 12, 7, 8)$$

Then it is easy to see that this is a $(12, 6, 3)$-verifiers set system which satisfies Corollary 1.

4.5 Efficiency

In the (v, b, t)-verifiers set system of Corollary 1, each MIX server is included in at most two blocks. Therefore, each MIX server acts as a prover at most twice and acts as a verifier at most twice.

In Step 1 and Step 2, each prover computes A_1 and encrypts $s_1, \cdots, s_N$ and π_j. This computation cost is $O(N)$. He publishes A_1 and the ciphertexts of $s_1, \cdots, s_N$ and π_j. This communication cost is $O(N)$. Therefore, the total cost of the prover is $O(N)$. In Step 3, each verifier decrypts the ciphertexts of $s_1, \cdots, s_N$, π_j and checks the validity of A_1. This computation cost is $O(N)$. He publishes "ACCEPT" or "REJECT". This communication cost is $O(1)$. Therefore, the total cost of the verifier is $O(N)$. In the end, the total cost of each MIX server is $O(N)$.

An alternative method to compute the computation cost per user is to analyze the total cost and then divide by the total number of MIX servers. One needs then to take into account that the number of MIX servers is $O(t^2)$, compared to $O(t)$ in previous work.

5 Security of the Protocol

5.1 Verifiability

Suppose that the prover P_j of B_j is malicious and A_1 is not correctly computed. Then there exists at least one honest verifier in B_j because $|B_j| = t+1$ and there exist at most t malicious MIX servers. The honest verifier outputs "REJECT" at Step 3. Therefore, A_1 is ignored at Step 4.

5.2 Robustness

For any t malicious MIX servers, there exists at least one B_i in which all MIX server are honest from Def.1. This B_i computes A_1 correctly. On the other hand, any invalid A_1 is ignored from the verifiability. Therefore, our protocol outputs a random permutation of $(m_1, \ldots, m_N)$ correctly even if there are at most t malicious MIX servers.

5.3 Privacy (Sketch)

The ElGamal based encryption scheme of [28] is known to be non-malleable under adaptive chosen ciphertext attack. Let c_1, c_2 be two ciphertexts and m_1, m_2 be the plaintexts. Then by using the result of [4], we can show that there exists no probabilistic polynomial time (p.p.t.) Turing machine D (distinguisher) which can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. This is the minimum requirement that any MIX net of this type must satisfy. We also assume that it satisfies *plaintext awareness* [5] which means that no p.p.t. adversary can create a ciphertext c without knowing its underlying plaintext m.

Now consider an adversary M_0 who can control at most t MIX servers and at most $N-2$ out of the N users posting encrypted messages. It is the goal of the adversary M_0 to match each one of the two plaintexts m_1, m_2 to their corresponding ciphertexts c_1, c_2 that he does not control. In other words, M_0 wishes to distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1).

Suppose that there exists a p.p.t. adversary M_0 who can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. For simplicity, suppose that M_0 controls users $3, \cdots, N$.

We will show a distinguisher D. The input to D is (c_1, c_2, z_1, z_2), where $(z_1, z_2) = (m_1, m_2)$ or (m_2, m_1). D first gives c_1, c_2 to M_0 and runs the users part of M_0. Then M_0 outputs $c_3, \cdots, c_N$. From the *plaintext awareness* assumption, M_0 knows the plaintexts $m_3, \cdots, m_N$ for $c_3, \cdots, c_N$. Therefore, D knows the set of $\{m_1, m_2, m_3, \cdots, m_N\}$.

D next runs the main body of our protocol in such a way that D simulates the part of honest MIX servers faithfully and uses M_0 for the part of malicious MIX servers. Let the output of the main body be $A_0 = ((\hat{c}_1, \hat{d}_1), \ldots, (\hat{c}_N, \hat{d}_N))$. Note that A_0 is a random permutation of randomized ciphertexts $c_1, \cdots, c_N$ from Sect. 5.2.

Let π be a random permutation. Let $\hat{m}_i$ denote the plaintext of $(\hat{c}_i, \hat{d}_i)$ for $1 \leq i \leq N$. Then we can show that M_0 cannot distinguish $(\hat{c}_i, \hat{d}_i, \hat{m}_i)$ and $(\hat{c}_i, \hat{d}_i, m_{\pi(i)})$ under the decision Diffie-Hellman assumption. D finally generates a view of M_0 for the robust $(t+1, n)$ threshold ElGamal decryption scheme with $(\hat{c}_i, \hat{d}_i, m_{\pi(i)})$ by using the technique of [8, 12] for $1 \leq i \leq N$.

Then M_0 can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability from our assumption on M_0. Hence, D can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. However, this is a contradiction.

6 Open Problems

This paper introduces several open problems, in particular:

- whether the new tools of existential-honesty and limited-open-verification can be used in other secure distributed computation.

- whether there are other choices of v. Indeed, when t is large the required number of MIX servers only grows quadratic. Although this is reasonable for a theoretician, from a practical viewpoint, the question is worth addressing.
- is $O(N)$ the minimum required effort per MIX server while maintaining t-privacy, t-verifiability, and t-robustness, in a network with $O(t^2)$ servers.

Acknowledgement

The authors thank Prof. Stinson for providing in August 1999 the second author of the paper with a preliminary copy of their Rees et al. paper [23]. The authors are grateful for the many comments received from the referee and from Dr. Jakobsson who shepherded the rewriting of the paper. Many of these comments have significantly contributed to increase the readability of the text.

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. M. Bellare, A. Desai, D. Poincheval, P. Rogaway, "Relations among notions of security for public key encryption schemes," Crypto '98, pp. 26–45.
5. M. Bellare, P. Rogaway, "Optimal asymmetric encryption - How to encrypt with RSA," Eurocrypt '94, pp. 92–111.
6. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, Vol. 24, 1981, pp. 84-88.
7. D. Chaum, H. Van Antwerpen, "Undeniable signatures," Crypto '89, pp. 212-216.
8. Y. Desmedt, Y. Frankel, "Threshold cryptosystems," Crypto '89, pp. 307–315.
9. D. Dolev, C. Dwork, M. Naor, " Non-malleable cryptography," STOC '91, pp. 542-552.
10. T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," Crypto '84, pp. 10-18.
11. A. Fujioka, T. Okamoto, K. Ohta, "A practical secret voting scheme for large scale elections," Auscrypt '92, pp. 244-251.
12. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust and efficient sharing of RSA functions," Crypto '96, pp. 157–172.
13. M. Jakobsson, "A practical MIX," Eurocrypt '98, pp. 448–461.
14. M. Jakobsson, D. M'Raihi, "Mix-based electronic payments," SAC'98, pp. 157–173.
15. M. Jakobsson, "Flash mixing," PODC'99, pp. 83–89.
16. M. Jakobsson, A. Juels "Millimix: Mixing in small batches," DIMACS Technical report 99-33 (June 1999)
17. W. H. Mills, "Covering design I: coverings by a small number of subsets," Ars Combin. 8, (1979), pp. 199–315.
18. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, "Fault tolerant anonymous channel," ICICS '97, pp. 440-444.
19. C. Park, K. Itoh, K. Kurosawa, "All/nothing election scheme and anonymous channel," Eurocrypt '93, pp. 248-259.

20. T. P. Pedersen, "A threshold cryptosystem without a trusted party," Eurocrypt '91, pp. 522-526.
21. B. Pfitzmann, A. Pfitzmann. "How to break the direct RSA-implementation of MIXes," Eurocrypt '89, pp. 373-381.
22. D. Pointcheval, J. Stern, "Security proofs for signature schemes," Eurocrypt '96, pp. 387-398.
23. R. Rees, D. R. Stinson, R. Wei, G. H. J. van Rees, "An application of covering designs: Determining the maximum consistent set of shares in a threshold scheme," Ars Combin. 531 (1999), pp. 225–237.
24. K. Sako, J. Kilian, "Receipt-free mix-type voting scheme," Eurocrypt '95, pp. 393-403.
25. C. P. Schnorr, "Efficient signature generation for smart cards," Crypto '89, pp. 239–252.
26. C. P. Schnorr, M. Jakobsson, "Security of discrete log cryptosystems in the random oracle + generic model," `http://www.bell-labs.com/user/markusj/`
27. A. Shamir, "How to share a secret," Communications of the ACM, Vol. 22, 1979, pp. 612-613
28. Y. Tsiounis, M. Yung, "On the security of ElGamal based encryption," PKC'98, pp. 117–134.
29. Edited by C. J. Colbourn and J. H. Dinitz, Handbook of Combinatorial Design, CRC Press (1996)