A Length-Invariant Hybrid Mix

Miyako Ohkubo and Masayuki Abe

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

Abstract. This paper presents a secure and flexible Mix-net that has the following properties; it efficiently handles long plaintexts that exceed the modulus size of underlying public-key encryption as well as very short ones (*length-flexible*), input ciphertext length is not impacted by the number of mix-servers (*length-invariant*), and its security in terms of anonymity is proven in a formal way (*provably secure*). One can also add robustness i.e. it outputs correct results in the presence of corrupt servers. The security is proved in the random oracle model by showing a reduction from breaking the anonymity of our Mix-net to breaking a sort of indistinguishability of the underlying symmetric encryption scheme or solving the Decision Diffie-Hellman problem.

Keywords: Mix-net, Hybrid Encryption, Anonymous Channel, Hybrid Mix

1 Introduction

1.1 Background

Mix-net is a cryptographic primitive that provides anonymity to message senders. It takes a list of encrypted messages sent from a sufficient number of users and outputs a list of corresponding plaintexts sorted in random order so that it conceals the correspondence between each plaintext and user. Accordingly, it provides anonymity by hiding the individual user in the mass. Such a primitive was first introduced in [7] with a heuristic construction based on public key encryption. Since then, many works have improved its usability and security. In [18], Park, et al., constructed a scheme based on El Gamal encryption, where the encryption work and resulting ciphertext length were independent of the number of mix-servers. Robustness was addressed in [22,16,17,3,12,1,13,14,9]. Attacks are found in [20,19,16,9].

A promising application of Mix-net is electronic voting as it can convey any style of ballots, e.g., simple binary value of Yes/No voting and free-format questionnaires, without changing the protocol. It is also useful in other applications such as anonymous payments and anonymous bids.

To support wide availability, Mix-net should be able to efficiently handle messages of various lengths that differ depending on the application. Some applications where users anonymously send signatures issued by an authority (possibly in a blind way) need Mix-net to convey signature-message pairs which require thousands of bits to be stored depending on the signature algorithm. Despite the need for handling long plaintexts, all previous Mix-nets limit messages to be shorter than a single block of ElGamal encryption or handle long messages in a heuristic way. For example, if ElGamal encryption is implemented over an elliptic curve for speed with typical settings, the messages are limited to just 160 bits, which greatly limits applicability. Although one can handle long messages simply by dividing each message into some blocks and repeating the atomic mix processing a sufficient number of times, such an approach results in an inefficient scheme.

Very short plaintexts are also dealt in an inefficient way as they are expanded to one block of underlying public-key encryption. For instance, ElGamal encryption with 1024 bit modulus expands 1 bit message to $1024 + 1024$ bits ciphertext. Hence, previous schemes incur higher communication costs than actually needed.

A common approach that overcomes such shortcomings would be to use hybrid encryption schemes that combine asymmetric key exchange and symmetric (common key) encryption. Although some provably secure hybrid encryption schemes are available in the literature, e.g., [10,23], applying those schemes does not immediately result in a secure and efficient Mix-net. It is not clear whether a secure hybrid encryption scheme provides security even in the context of Mix-net. Furthermore, a straightforward use of hybrid encryption in the original construction of Chaum [7] obviously extends the resulting input ciphertext linearly depending on the number of servers.

1.2 Our Contribution

This paper presents Mix-nets that realize, for the first time, the following properties all at the same time.

- ***Length-flexibility***: The size of the public-key of Mix-net does not limit plaintext length. Plaintexts of any length are encrypted efficiently in terms of computation and resulting ciphertext length.
- ***Length-invariance***: The length of input ciphertexts is independent of the number of mix-servers.
- ***Provable security***: The security, in terms of anonymity, of our Mix-net can be proven in the random oracle model [5] assuming the intractability of the Decision Diffie-Hellman problem and the availability of a symmetric encryption scheme that ensures a sort of indistinguishability.

Furthermore, we show an approach to add *robustness* so that correct output is obtained even if some of the users and servers behave maliciously.

To achieve the above goals, we developed a novel hybrid encryption scheme with group decryption feature that suits Mix-net. Informally, it conceals the correspondence between inputs and outputs at each step of group decryption performed by each server.

Our scheme saves communication cost for short messages as well as long ones since the encryption only extends the message with modulus length. For instance, input ciphertext is 1024 + 1 bits long for a 1 bit message with 1024 bit modulus. Computational cost of encryption grows linearly depending on the number of servers. We show, however, in section 7, that the cost can actually be smaller than that of previous standard schemes in many settings.

We first introduce a basic scheme that highlights our key idea. It is secure only against honest but curious users and mix-servers. We then add security to withstand distrustful users (mix-servers are still honest but curious). If needed, one can add individual verifiability to these basic schemes in a simple standard way in order to detect the deviation of servers with some probability. Such schemes would be applicable for applications where mix-servers are chosen carefully and thus are more creditable than users. Such schemes would also be used in the applications, such as anonymous donation or payment, in which each user is not concerned about the input of other users and thus individual verifiability is sufficient.

We then add robustness by following [9]. As in their scheme, the resulting Mix-net is robust in such a sense that it outputs a correct result and provides anonymity in the presence of corrupt servers, but does not provide universal verifiability. That is, only the servers can be convinced of the correctness of the results while no external parties can verify them. Such a model was also addressed in [12,13]. Accordingly, such scheme would be useful, for instance, for small scale applications where every user can act as a mix-server.

2 Model

2.1 Scenario

There are n users and m mix-servers. Let U_i and M_j denote user i and server j, respectively. For simplicity, we assume that all communication between these participants are done via a bulletin board. The scenario consists of three phases.

Preliminary phase: The maximum length of each plaintext, say ℓ_{msg} is announced to all users together with other application-dependent information. It is stressed that ℓ_{msg} is independent of the public key size and is determined by the Mix-net application. Theoretically, ℓ_{msg} can be any positive integer bound by a polynomial of the security parameter.

Casting phase: Each user encrypts his message and sends it to the bulletin board. Appropriate padding may be applied to the message before encryption so that the length of the message equals ℓ_{msg}.

Mixing phase: Let L_0 be a list of all ciphertexts sent from the users. The first server takes L_0 and outputs a list, L_1, to the bulletin board. Similarly, server i takes L_{i-1} and outputs L_i. The final output of the mix-net is L_m. If all servers work correctly, L_m is a list of plaintexts sorted in random-order.

Let $\bar{L}_0$ be a list of messages obtained by correctly decrypting each ciphertext in L_0. The output of mix-net, L_m, is said to be *correct* if there exists a permutation between $\bar{L}_0$ and L_m. We say, informally, that the mix-net provides *anonymity* if it is intractable to distinguish two plain messages in L_m that originate from two honest users.

2.2 Adversaries

We represent the power of an adversary by $(t_u, t_s)^{**}$ where $*$ is either $\mathtt{A}$ or $\mathtt{P}$ meaning *active* or *passive*, respectively. For instance, $(t_u, t_s)^{\mathtt{AP}}$-adversary means that the adversary can thoroughly control up to t_u users (i.e., active against users), and can obtain views from up to t_s mix-servers (i.e., passive against servers). Note that adversaries that are passive against servers only attempt to violate anonymity as they can not control the servers so as to output incorrect results. We assume that the adversaries are static meaning that they decide, before the protocol begins, which users and servers they will attack.

In this paper, we deal with the following types of adversaries. The types are listed in order of increasing strength.

– $(n-2, m-1)^{\mathtt{PP}}$-adversary; There are at least two unattacked users and an unattacked server. The adversary can obtain views from attacking users and servers but can not control either of them. Our basic scheme is safe against this type of adversary.
– $(n-2, m-1)^{\mathtt{AP}}$-adversary; The same as above, but can control corrupt users. Since the adversary can send any ciphertexts through the corrupt users and let the servers decrypt them, it can launch chosen ciphertext attacks. Our extended scheme withstands this type of adversary.
– $(n-2, \mathcal{O}(\sqrt{m}))^{\mathtt{AA}}$-adversary; This type of adversary, which is the strongest of the three, attempts to violate anonymity *or correctness*. Our third scheme withstands such an adversary.

3 The Basic Scheme

Let $\mathcal{G}$ be a discrete logarithm instance generator such that $(p, q, g) \leftarrow \mathcal{G}(1^k)$ where k is a security parameter, and p, q are primes that satisfy $q|p-1$, and g is an element of Z_p^* whose order is q. Let $\langle g \rangle$ denote a unique subgroup of Z_p^* generated by g. All the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

$(\mathcal{E}, \mathcal{D}, \mathcal{K}, \mathcal{M}, \mathcal{C})$ denotes a symmetric encryption scheme where $\mathcal{E}, \mathcal{D}$ are the encryption and decryption algorithms and $\mathcal{K}, \mathcal{M}, \mathcal{C}$ are the spaces for keys, messages, and ciphertexts, respectively. $\mathcal{E}_K(x)$ denotes the result of encrypting plaintext x with common key K. Similarly, $\mathcal{D}_K(x)$ denotes the plaintext obtained by decrypting ciphertext x with key K. We assume that the symmetric encryption scheme is length-preserving, i.e., $\mathcal{M} = \mathcal{C} = \{0,1\}^{\ell_{msg}}$. Let H be a hash function, $H : \langle g \rangle \to \mathcal{K}$.

[Key generation]
Server i randomly selects a pair of private keys a_i, x_i from Z_q^* and computes

$$h_i := h_{i-1}^{a_i}, \text{ and}$$
$$y_i := h_i^{x_i},$$

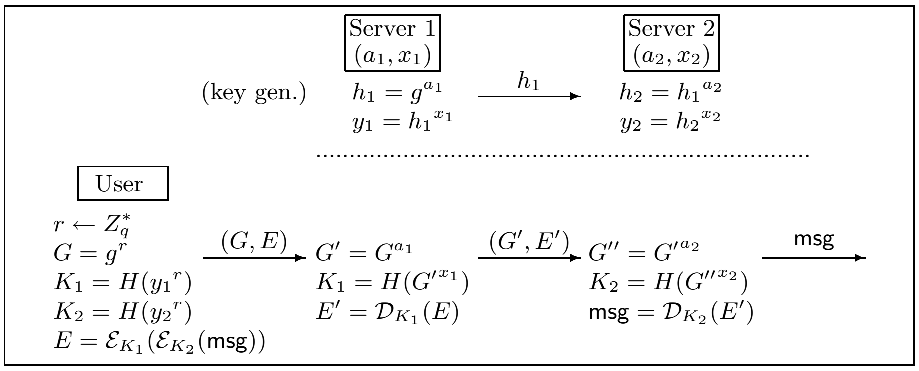

Fig. 1. The basic hybrid Mix-net with two servers. The input to Server 1 is a list of (G, E)s made by each user. Server 1 outputs a list of randomly ordered (G', E')s. Server 2 finally outputs a list of randomly ordered plaintexts. $\mathcal{E}, \mathcal{D}$ are the encryption and decryption algorithms. H is a hash function.

for h_{i-1} given from the previous server. (Let $h_0 = g$ for the first server.) It then publishes y_i and h_i as a pair of public keys.

[Encryption]
User encrypts message $msg \in \{0,1\}^{\ell_{msg}}$ to ciphertext C as

$$C := (G, E) = (g^r,\ \mathcal{E}_{K_1} \cdots \mathcal{E}_{K_m}(msg))$$

where r is randomly taken from Z_q^*, and $K_1, \dots, K_m$ are session keys for symmetric encryption $\mathcal{E}$ and are computed as

$$K_i := H({y_i}^r).$$

[Mix Decryption]
For $i = 1$ to m, server i decrypts each ciphertext $C = (G, E)$ in list L_{i-1} to get $C' = (G', E')$ as

$$G' := G^{a_i},$$
$$E' := \mathcal{D}_{K_i}(E) \text{ where } K_i = H({G'}^{x_i}).$$

(For the last server, let $C' = E'$.) Server i then selects a random permutation π_i of $\{1, \dots, n\}$ and puts the resulting ciphertexts into L_i in the random order defined by π_i.

Figure 1 illustrates the above basic scheme with two servers. For each i, $H({G'}^{x_i}) = H(h_{i-1}^{r a_i x_i}) = H(h_i^{r x_i}) = H({y_i}^r) = K_i$ holds. Thus, if every server works correctly, correct session keys are retrieved by each server and the correct plaintext is obtained.

4 Security of the Basic Scheme

4.1 Definitions and Assumptions

For $\rho := (p, q, g)$ generated by $\mathcal{G}(1^k)$, we assume that all poly-time algorithms solve the following problems only with negligible (in k) advantage over randomly guessing.

Definition 1. (Computational Diffie-Hellman Problem : CDHP)

Input: (ρ, g^a, g^b) *where* $a, b \leftarrow Z_q$.

Output: g^{ab}

Definition 2. (Decision Diffie-Hellman Problem : DDHP)

Input: $(\rho, g^a, G_0, G_1, {G_b}^a)$ *where* $b \leftarrow \{0, 1\}$*,* $a \leftarrow Z_q$*,* $G_0, G_1 \leftarrow \langle g \rangle$.

Output: b

Definition 3. (Matching Diffie-Hellman Problem : MDHP)

Input: $(\rho, g^a, G_0, G_1, {G_b}^a, {G_{\bar{b}}}^a)$ *where* $b \leftarrow \{0, 1\}$*,* $a \leftarrow Z_q$*,* $G_0, G_1 \leftarrow \langle g \rangle$

Output: b

It holds that CDHP $>$ DDHP $>$ MDHP, i.e., CDHP is the hardest to solve. The reverse relation between CDHP and DDHP is not known. For DDHP and MDHP, we can show that MDHP $=$ DDHP following [11] or [21].

Next we define the Matching Find-Guess problem, which is closely related to the Find-Guess problem [10] which defines a sort of indistinguishability of symmetric encryption schemes.

Definition 4. (Matching Find-Guess Problem : MFGP)

Input: $(\mathcal{E}_{K_0}(x_0), \mathcal{E}_{K_1}(x_1), x_b, x_{\bar{b}})$ *where* $x_0, x_1 \leftarrow \mathcal{M}$*,* $K_0, K_1 \leftarrow \mathcal{K}$*,* $b \leftarrow \{0, 1\}$.

Output: b

We say that a symmetric encryption is secure in the sense of MFG if for all poly-time algorithms MFGP can be solved only with negligible advantage over 1/2. Clearly, a one-time pad provides security in the sense of MFG. A stream cipher also provide the same security if its generator produces a pseudorandom bit-stream. For the sake of efficiency, we expect that existing carefully designed symmetric encryption schemes used in an appropriate mode of operation such as OFB provide such security as well.

In our construction, session keys for symmetric encryption are derived by applying hash function H to the results of the Diffie-Hellman key exchange. Namely, the Diffie-Hellman key exchange and the symmetric encryption are *connected* by hash function H. The security of our hybrid encryption scheme is related to the following problem.

Definition 5. (MFG-MDH Joint Problem)

Input: $(\rho, g^a, g^{ax}, M_0, G_0, M_1, G_1, M'_b, G'_b, M'_{\bar{b}}, G'_{\bar{b}})$
Where $a, x \leftarrow Z_q^*$, $b \leftarrow \{0,1\}$, *and*
for $i = (0,1)$, $G_i \leftarrow \langle g \rangle$, $M_i \leftarrow \mathcal{M}$, $G'_i = G_i^a$, $M'_i = \mathcal{D}_{K_i}(M_i)$
where $K_i = H({G'_i}^x)$.

Output: b

It will be shown that if H is an ideal hash function then solving the above joint problem is as hard as solving either MFGP or MDHP.

4.2 Theorems and Proofs

Theorem 1. *The basic scheme provides anonymity in the presence of* $(n-2, m-1)^{\mathsf{PP}}$*-adversary if DDHP and the MFGP are intractable.*

To support this theorem, we will prove the following lemmas.

Lemma 1. *If there exists an* $(n-2, m-1)^{\mathsf{PP}}$*-adversary* $\mathcal{A}_{Xp}$ *that breaks anonymity in our Mix-net, then there exists machine* $\mathcal{A}_M$ *that solves the MFG-MDH joint problem with probability non-negligibly better than 1/2.*

Lemma 2. *If* $\mathcal{A}_M$ *exists, then there exists machine* $\mathcal{A}_D$ *that solves, at least, either MFGP or MDHP with probability non-negligibly better than 1/2.*

Lemma 3. *If* $\mathcal{A}_D$ *solves MDHP, then there exists a machine* $\mathcal{A}_H$ *that solves DDHP with probability non-negligibly better than 1/2.*

Here, we sketch the proof of Lemma 1 and put the proof of Lemma 2 in the Appendix. Lemma 3 can be proven in the same way as shown in [11], so its proof is omitted.

Proof of Lemma 1 (sketch): Let M_ξ be the server that $\mathcal{A}_{Xp}$ does not attack, i.e, the one whose view is not given to $\mathcal{A}_{Xp}$. Given an MFG-MDH joint problem instance $(\mathbf{p}, \mathbf{q}, \mathbf{g}, \mathbf{g^a}, \mathbf{g^{ax}}, \mathbf{M_0}, \mathbf{G_0}, \mathbf{M_1}, \mathbf{G_1}, \mathbf{M'_b}, \mathbf{G'_b}, \mathbf{M'_{\bar{b}}}, \mathbf{G'_{\bar{b}}})$, $\mathcal{A}_M$ simulates the view of $\mathcal{A}_{Xp}$ as follows.

Simulating Keys: For server M_ξ, $\mathcal{A}_M$ sets $h_\xi = \mathbf{g^a}$ and $y_\xi = \mathbf{g^{ax}}$. For the keys of descending servers, $\mathsf{M}_{\xi+1}, \ldots, \mathsf{M}_m$, $\mathcal{A}_M$ follows the key generation procedure, i.e. randomly chooses private keys a_i, x_i and computes corresponding public keys $h_i = h_{i-1}^{a_i}, y_i = h_i^{x_i}$. For the keys of ascending servers, a little thought is needed. Let $h_{\xi-1} = \mathbf{g}$. Then, for $i = \xi - 1$ to 1, $\mathcal{A}_M$ chooses a_i, x_i and computes $h_{i-1} = h_i^{1/a_i}, y_i = h_i^{x_i}$. It finally sets $\rho = (\mathbf{p}, \mathbf{q}, h_0)$.

Simulating Lists: $\mathcal{A}_M$ puts $(\mathbf{M'_b}, \mathbf{G'_b})$ and $(\mathbf{M'_{\bar{b}}}, \mathbf{G'_{\bar{b}}})$ into random positions in L_ξ. It then randomly generates other entries of L_ξ by taking M randomly from ciphertext space $\mathcal{C}$ and computing G as h_ξ^r with randomly chosen r. Next, it selects random permutation π_ξ and computes each entry of $L_{\xi-1}$ by encrypting the corresponding entry in L_ξ. (This is possible because $\mathcal{A}_M$ can retrieve the correct session key by computing y_ξ^r.) For the two special entries of $L_{\xi-1}$ that correspond to $(\mathbf{M'_b}, \mathbf{G'_b})$ and $(\mathbf{M'_{\bar{b}}}, \mathbf{G'_{\bar{b}}})$, $\mathcal{A}_M$ inserts $(\mathbf{M_0}, \mathbf{G_0})$ and $(\mathbf{M_1}, \mathbf{G_1})$.

Now, the rest of ascending lists, $L_{\xi-2}$ to L_1, can be computed in order by *encrypting* the previous lists with the simulated private keys and randomly generated permutations. (Note that those permutations are chosen so that the two special entries of $L_{\xi-1}$ correspond to the inputs from unattacked users U_1 and U_2.) Similarly, the rest of descending lists, $L_{\xi+1}$ to L_m, can be computed by *decrypting* from L_ξ to L_{m-1} in order. In the course of the above simulation, $\mathcal{A}_M$ consults random oracle H to compute the session keys.

Views of attacked users and servers can be appropriately simulated by using messages in L_m and random choices of above simulation. Given the perfectly simulated views and lists, and free access to H, $\mathcal{A}_{Xp}$ distinguishes two messages in L_m originated from U_1 and U_2. From the result of $\mathcal{A}_{Xp}$, $\mathcal{A}_M$ can derive the correspondence between two special positions in $L_{\xi-1}$ and L_ξ where the given instances were placed by using the permutation taken by the simulated servers, except for M_ξ. The success provability of $\mathcal{A}_M$ is the same as that of $\mathcal{A}_{Xp}$. □

5 Securing against Corrupt Users

The key idea to add security against corrupt users is to make the underlying encryption non-malleable so that they cannot launch chosen ciphertext attacks. Although several efficient non-malleable encryption schemes are available (e.g. [24,25,8,6,2,10,23]), few meet our requirements. For our security proof, we need the underlying encryption scheme that provides *plaintext awareness* [4] and *public verifiability*. The latter functionality allows the validity of ciphertexts to be checked without using the decryption key. Our solution is based on [24].

Overall, the protocols are unchanged except that users attach a kind of proof and the mix-servers screen ciphertexts that come with invalid proofs.

[Encryption]
Message *msg* is encrypted to $C = (G, E)$ in the same way as in the basic scheme. Let $G = g^r$. A proof of knowing r is defined as $P := (e, z, \bar{G}, \tilde{G}, \eta, \bar{\eta}, \tilde{\eta})$ such that

$$\begin{aligned}
&\bar{g} := H_2(G), \tilde{g} := H_3(G),\\
&\bar{G} := \bar{g}^r, \tilde{G} := \tilde{g}^r,\\
&\eta := g^\varsigma, \bar{\eta} := \bar{g}^\varsigma, \tilde{\eta} := \tilde{g}^\varsigma,\\
&e := H_4(E, g, \bar{g}, \tilde{g}, G, \bar{G}, \tilde{G}, \eta, \bar{\eta}, \tilde{\eta}),\\
&z := \varsigma - re \bmod q,
\end{aligned}$$

where $\varsigma \leftarrow Z_q$ and H_2, H_3, H_4 are hash functions. The output is (C, P).

[Mix Decryption]
Each server first verifies that

$$e = H_4(E, g, \bar{g}, \tilde{g}, G, \bar{G}, \tilde{G}, g^z G^e, \bar{g}^z \bar{G}^e, \tilde{g}^z \tilde{G}^e)$$

holds for each input ciphertext. Here $\bar{g}$, $\tilde{g}$ are computed as $\bar{g} := H_2(G)$, $\tilde{g} := H_3(G)$, respectively. After they agree on the result of the verification, they put C, which came with valid P, into list L_0. The rest of the process is the same as that of the basic scheme.

The benefit of P is that it makes the simulation of a server, say M_ξ, possible without knowing its private keys. The trick is as follows. We want to derive K_ξ and G' for each ciphertext in L_ξ. Set $\bar{g} = H_2(G) = {y_\xi}^u$ and $\tilde{g} = H_3(G) = {h_\xi}^v$ taking u and v randomly from Z_q. It follows that valid P should contain $\bar{G} = \bar{g}^r = y_\xi^{ru}$ and $\tilde{G} = \tilde{g}^r = h_\xi^{rv}$. Thus, we can compute $H(\bar{G}^{1/u}) = H(y_\xi^r) = K_\xi$ and $\tilde{G}^{1/v} = h_\xi^r = G'$ as expected.

Theorem 2. *The extended scheme provides anonymity in the presence of* $(n-2, m-1)^{\mathsf{AP}}$*-adversaries if DDHP and MFGP are intractable.*

To prove the above theorem, it is sufficient to prove the following lemma. The rest of the proof is supported by Lemma 2 and 3.

Lemma 4. *If there exists an* $(n-2, m-1)^{\mathsf{AP}}$*-adversary that breaks anonymity in our Mix-net, there exists machine that solves the MFG-MDH joint problem.*

Proof (sketch) The difference of this proof from that of Lemma 1 is twofold:

- the proof-part P of the inputs from honest users has to be simulated, and
- the simulator A_M has to correctly decrypt the input ciphertexts coming from a corrupt user without knowing the private keys of the unattacked server.

For the first point, we will use the standard simulation technique for the honest verifier public-coin zero-knowledge proofs by regarding H_4 as a random oracle.

For the second point, we exploit the plaintext awareness (PA) of the underlying encryption scheme. $\mathcal{A}_M$ first computes L_m by using the PA property, and subsequently computes $L_{m-1}, ..., L_\xi$ by encrypting each entry of the previous list with the simulated public keys. $\mathcal{A}_M$ then computes L_0 to $L_{\xi-1}$ by correctly performing decryption with the simulated decryption keys. In this way, the resulting $L_{\xi-1}$ and L_ξ have the same relation as the one in the real execution with regard to the public key of unattacked server M_ξ. $\square$

6 Securing against Corrupt Servers

Robustness is added following [9]. Let us briefly introduce the main idea here and omit the details due to page restriction.

To prevent corrupt servers from behaving maliciously, we group servers in such a way that every group contains at least one honest server, and at least one group consists only of honest servers. Such grouping is easily formed by placing

$t+1$ servers in each of $t+1$ groups when $m=(t+1)^2$. Then, a representative member in each group executes mix decryption and all other members are given all private information (secret keys and random choices) used in mix decryption and monitor its behavior. Thus malicious deviation in each group is always detected by an honest member. Since there is at least one perfectly honest group, the group works correctly and outputs correct result.

7 Efficiency Analysis

Here, the computational cost of our basic scheme is compared to that of [18], which is one of the widely known schemes based on ElGamal encryption. It provides the same level of security as our basic scheme. Their scheme is summarized as follows; Server i has private key x_i and public key $y_i = g^{x_i}$. For each ElGamal ciphertext $(G, M) = (g^r, msg \cdot y^r)$ where $y = \prod_{j=1}^{m} y_j$, server i computes $(G', M') = (Gg^t, Mg^{-x_i}\hat{y}_i^t)$ where $\hat{y}_i = \prod_{j=i}^{m} y_j$. In this scheme, q must be very large so that all potential messages are in the subgroup generated by g. Hence we assume $|p| \approx |q|$. On the other hand, since our scheme needs randomness sufficient for generating symmetric keys, $|q|$ can be much smaller than $|p|$.

Table 1 shows the number of modular multiplications needed for encryption assuming the use of the binary method for exponentiation. For double-base and single-base exponentiation, we assume the simple table-lookup method described in [15] which costs $\frac{7}{4}|q|$ multiplications for a double-base exponentiation, and $\frac{3}{2}|q|$ for a single-base exponentiation. Although user's computation is linear in the number of servers, for a typical setting, say $|p| = 1024$ and $|q| = 160$, our scheme enjoys lesser computation (excl. symmetric encryption) up to 11 servers.

This advantage will be lost if one considers elliptic curve implementation where $|p| \approx |q|$. However, our scheme still saves computation if messages exceeds 160 bits as symmetric encryption is 100 to 1000 times faster than scalar multiplication over an elliptic curve.

Scheme	User	Each Server
El Gamal [18]	$\frac{3}{2}\|p\| \times 2$	$\frac{7}{4}\|p\| + \frac{3}{2}\|p\|$
Ours (basic)	$\frac{3}{2}\|q\| \times (m+1)$	$\frac{3}{2}\|q\| \times 2$

Table 1. Number of modular multiplications per message. m is number of servers.

8 Open Problems

The resulting robust scheme still has some issues that must be resolved. First, it is preferable to provide public verifiability so that anyone outside of the mix can be convinced of the output. Second, optimal resiliency should be provided. At least, we need linear resiliency, i.e., $\mathcal{O}(m)$, instead of $\mathcal{O}(\sqrt{m})$.

Our security proof of Lemma 1 is for the case where honest users select messages uniformly from $\{0,1\}^{\ell_{msg}}$. However, the users might be restricted to choose messages from exponentially sparse space (with length ℓ_{msg}). In such a case, our simulation in the proof of Lemma 1 is not suitable. That is because the plain messages obtained by decrypting the given instance of the joint problem are not likely to fall into the exponentially sparse space. Since we assume that the underlying encryption scheme provides security equally for all messages, such restriction on message space is not likely to impact security. It remains, however, as an open problem to prove this in a formal way.

Acknowledgment

The authors thank Kazue Sako for her invaluable suggestions and comments.

References

1. M. Abe. Mix-networks on permutation networks. *Asiacrypt '99*, LNCS 1716, pages 258–273. 1999.
2. M. Abe. Robust distributed multiplication without interaction. In M. Wiener, editor, *in Cryptology — CRYPTO '99*, LNCS 1666, pages 130–147. 1999.
3. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. *IEICE Trans. Fundamentals*, E83-A(7):1431–1440, July 2000. Presented in Eurocrypt'98.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *CRYPTO '98*, LNCS 1462, pages 26–45. 1998.
5. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *ACM CCCS*, pages 62–73. 1993.
6. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. *Eurocrypt '99*, LNCS 1592, pages 90–106. 1999.
7. D. L. Chaum. Untraceable electronic mail, return address, and digital pseudonyms. *Comm. of the ACM*, 24:84–88, 1981.
8. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *CRYPTO '98*, LNCS 1462, pages 13–25. 1998.
9. Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. *Eurocrypt 2000*, LNCS 1807, pages 557–572. 2000.
10. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *CRYPTO '99*, LNCS 1666, pages 537–554. 1999.
11. H. Handschuh, Y. Tsiounis, and M. Yung. Decision oracles are equivalent to matching oracles. *PKC '99*, LNCS 1560, pages 276–289. 1999.
12. M. Jakobsson. A practical mix. *Eurocrypt '98*, LNCS 1403, pages 448–461. 1998.
13. M. Jakobsson. Flash mixing. *PODC99*, 1999.
14. A. Juels and M. Jakobsson. Millimix. Tech. Report 99-33, DIMACS, June 1999.

15. A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
16. M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. *Asiacrypt '96*, LNCS 1163, pages 125–132. 1996.
17. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. *ICICS '98*, LNCS 1334, pages 440–444. 1998.
18. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. *Eurocrypt '93*, LNCS 765, pages 248–259. 1994.
19. B. Pfitzmann. Breaking an efficient anonymous channel. *Eurocrypt '94*, LNCS 950, pages 339–348. 1995.
20. B. Pfitzmann and A. Pfitzmann. How to break the direct RSA implementation of MIXes. *Eurocrypt '89*, LNCS 434, pages 373–381. 1989.
21. T. Saitoh. The efficient reductions between the decision Diffie-Hellman problem and related problems. In the joint meeting of SCI2000 and ISAS2000, July 2000. available from the author.
22. K. Sako and J. Kilian. Receipt-free mix-type voting scheme — a practical solution to the implementation of a voting booth —. *Eurocrypt '95*, LNCS 921, pages 393–403. 1995.
23. V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. *Eurocrypt 2000*, LNCS 1807, pages 275–288. 2000.
24. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Eurocrypt '98*, LNCS 1403, pages 1–16. 1998.
25. Y. Tsiounis and M. Yung. On the security of El Gamal based encryption. *PKC '98*, LNCS 1431, pages 117–134. 1998.

Appendix (Proof of Lemma 2)

We first show that $\mathcal{A}_M$ can be used to solve CDHP, or else can be used to solve either MFGP or MDHP. Let $(\mathbf{M_0}, \mathbf{M_1}, \mathbf{M'_{b_F}}, \mathbf{M'_{\bar{b}_F}})$ be an instance of MFGP and $(\rho, \mathbf{h}, \mathbf{G_0}, \mathbf{G_1}, \mathbf{G^a_{b_D}}, \mathbf{G^a_{\bar{b}_D}})$ be an instance of MDHP. Let $y := \mathbf{h}^x$ for $x \leftarrow Z_q$. The input to $\mathcal{A}_M$ is a tuple such as : $Input^*_{\mathcal{A}_M} := \{\rho, y, \mathbf{h}, (\mathbf{M_0}, \mathbf{G_0}), (\mathbf{M_1}, \mathbf{G_1}), (\mathbf{M'_{b_F}}, \mathbf{G^a_{b_D}}), (\mathbf{M'_{\bar{b}_F}}, \mathbf{G^a_{\bar{b}_D}})\}$. Note that this may not be a correct instance of the joint problem as b_D and b_F may not be the same. Now we observe the behavior of $\mathcal{A}_M$ given this input. Let q_F be the maximum number of queries from $\mathcal{A}_M$ to H. Here q_F is limited to a polynomial in the security parameter κ. Let Q_i denote the i-th query to H. If there exists i such that $Q_i = (G^{ax}_{b_D})$, then $G^{ax}_{b_D}$ is the answer of CDHP $[h, y, G^a_{b_D}]$ ($[h, y, G^a_{b_D}]$=$[g^a, g^{ax}, g^{ar}]$, $G^{ax}_{b_D} = g^{axr}$). Similarly, if there exists i such that $Q_i = (G^{ax}_{\bar{b}_D})$, $G^{ax}_{\bar{b}_D}$ is the answer of CDHP $[h, y, G^a_{\bar{b}_D}]$. Define $\mathcal{P}_{DH}$ as $\mathcal{P}_{DH} = Pr[{}^\exists i \in \{1, \ldots, \mathsf{q}_F\}, {}^\exists j \in \{0, 1\}\ ;\ Q_i = (G^{ax}_j)]$. As above, $\mathcal{A}_M$ can be used to solve CDHP.

Next we show that $\mathcal{A}_M$ can be used to solve either MFGP or MDHP. Suppose that no such queries exist. In this case, the symmetric keys used in the MFGP are independent of the MDHP part because of the randomness of H; the adversary $\mathcal{A}_D$ makes those keys randomly without asking random oracle H. We next consider the relation between b_F and b_D.

1. The case of $b_F = b_D$.
   $Input^*_{\mathcal{A}_M}$ is perfectly indistinguishable from the correct input coming from the answer of random oracle H. So $\mathcal{A}_M$ will output $\tilde{\tilde{b}}$ as $\tilde{\tilde{b}} = b$ with probability $(> \frac{1}{2} + \mu_A)$, i.e., the same as the success probability of $\mathcal{A}_{Xp}$
2. The case of $b_F \neq b_D$.
   In this case, the distribution of $Input^*_{\mathcal{A}_M}$ is not correct.
   If $\mathcal{A}_M$ does not stop within t_A, $\tilde{\tilde{b}}$ is randomly taken from $\{0,1\}$. Now, define $\mathcal{P}_F$ and $\mathcal{P}_D$ as follows. $\mathcal{P}_F = Pr[\tilde{\tilde{b}} = b_F | b_F \neq b_D]$, $\mathcal{P}_D = Pr[\tilde{\tilde{b}} = b_D | b_F \neq b_D]$.

   If $b_F \neq b_D$, $\tilde{\tilde{b}}$ must equal to either b_F or b_D. Hence, $\mathcal{P}_F + \mathcal{P}_D = 1$.
   Since both $b_F = b_D$ and $b_F \neq b_D$ happen with probability $\frac{1}{2}$, we have

$$Pr[\tilde{\tilde{b}} = b_F] = (1 - \mathcal{P}_{DH})\{\frac{1}{2}(\frac{1}{2} + \mu_A) + \frac{\mathcal{P}_F}{2}\} = (1 - \mathcal{P}_{DH})\{\frac{1 + 2\mathcal{P}_F}{4} + \frac{\mu_A}{2}\} \quad (1)$$

$$Pr[\tilde{\tilde{b}} = b_D] = (1 - \mathcal{P}_{DH})\{\frac{1}{2}(\frac{1}{2} + \mu_A) + \frac{\mathcal{P}_D}{2}\} = (1 - \mathcal{P}_{DH})\{\frac{1 + 2\mathcal{P}_D}{4} + \frac{\mu_A}{2}\} \quad (2)$$

According to equation (8), either $\mathcal{P}_F$ or $\mathcal{P}_D$ is not less than $\frac{1}{2}$. Therefore, either Equation (1) or (2) is not negligible, or both.

The common key used in MFGP is perfectly indistinguishable from the actual common key derived from the answer of random oracle H because it is decided randomly. So if the answer of the CDHP is not contained among the list of queries from $\mathcal{A}_M$ to H, we can say that $(M_0, M_1, M'_b, M'_{\bar{b}})$ and $(G_0, G_1, G'_b, G'_{\bar{b}})$ have no relation to each other, though they affect each other through the common key in the actual input. Hence MFGP and MDHP are independent, and neither provides any help in solving the other.

Based on the above observation, we construct Block T1, Block T2, Block T3 that solve CDHP, MFGP and MDHP, respectively.

[Block T1]

1. Receive CDHP instance $(\rho, \mathbf{g}^\alpha, \mathbf{g}^\beta)$.
2. Make an MDHP instance as follows.
   – Choose $b_1 \leftarrow \{0,1\}$.
   – $y := \mathbf{g}^\alpha$, $G'_{b_1} := \mathbf{g}^\beta$, $G'_{\bar{b}_1} \leftarrow \langle g \rangle$
   – $a \leftarrow Z^*_q$, $g := \mathbf{g}^{\frac{1}{a}}$, $h := \mathbf{g}^a$, $G_0 := {G'_0}^{\frac{1}{a}}, G_1 := {G'_1}^{\frac{1}{a}}$
   – $\rho = (\mathbf{p}, \mathbf{q}, g)$
3. Make an MFGP instance as $M_i \leftarrow \mathcal{M}$, $K_i \leftarrow \mathcal{K}$, $M'_i := \mathcal{D}_{K_i}(M_i)$ for $i = 0, 1$.
4. Choose $b \leftarrow \{0,1\}$.
5. Choose I randomly from $1 \leq i \leq q_F$.
6. Input the following to $\mathcal{A}_M$.
   $Input'_{\mathcal{A}_M} = \{\rho, h, y, (M_0, G_0), (M_1, G_1), (M'_b, G'_b), (M'_{\bar{b}}, G'_{\bar{b}})\}$
7. If $\mathcal{A}_M$ poses query to H, return a random value chosen from key space $\mathcal{K}$. If it is the I-th query, output and stop.

Observe that the simulation is perfect only if the correct answer of CDHP is Q_I, or it is not asked to H (otherwise we have answered to the query with randomly chosen session key that may confuse $\mathcal{A}_M$).

[Block T2]

1. Receive MFGP instance $(\mathbf{M_0}, \mathbf{M_1}, \mathbf{M'_{b_F}}, \mathbf{M'_{\bar{b}_F}})$.
2. Make an MDHP instance as follows.
   - $(p, q, g) \leftarrow \mathcal{G}(1^\kappa)$
   - $a, x \leftarrow Z_q^*,\ y := g^{ax},\ h := g^a$
   - $r_0, r_1 \leftarrow Z_q^*,\ G_0 := g^{r_0}, G_1 := g^{r_1},\ G'_0 := h^{r_0}, G'_1 := h^{r_1}$
3. Choose $b_D \leftarrow \{0,1\}$. Next input the following to $\mathcal{A}_M$.
   $Input''_{\mathcal{A}_M} = \{\rho, h, y, (\mathbf{M_0}, G_0), (\mathbf{M_1}, G_1), (\mathbf{M'_{b_F}}, G'_{b_D}), (\mathbf{M'_{\bar{b}_F}}, G'_{\bar{b}_D})\}$
4. For all queries from $\mathcal{A}_M$ to H, return a random value from $\mathcal{K}$.
5. Output $\tilde{\tilde{b}}$ that $\mathcal{A}_M$ outputs.

[Block T3]

1. Receive MDHP instance $(\rho, \mathbf{h}, \mathbf{G_0}, \mathbf{G_1},\ \mathbf{G^a_{b_D}}, \mathbf{G^a_{\bar{b}_D}})$.
2. Make an MFGP instance as $M_i \leftarrow \mathcal{M}$, $K_i \leftarrow \mathcal{K}$, $M'_i := \mathcal{D}_{K_i}(M_i)$ for $i = \{0,1\}$.
3. Choose $b_F \leftarrow \{0,1\}$. Next input the following to $\mathcal{A}_M$.
   $Input'''_{\mathcal{A}_M} = \{\rho, h, y, (M_0, \mathbf{G_0}), (M_1, \mathbf{G_1}), (M'_{b_F}, \mathbf{G'_{b_D}}), (M'_{\bar{b}_F}, \mathbf{G'_{\bar{b}_D}})\}$
4. For all queries from $\mathcal{A}_M$ to H, return a random value from $\mathcal{K}$.
5. Output $\tilde{\tilde{b}}$ that $\mathcal{A}_M$ outputs.

By using the above blocks, we construct $\mathcal{A}_D$ as follows.

[**Construction of $\mathcal{A}_\mathbf{D}$**]

1. Receive CDHP instance $(\rho, \mathbf{g}^\alpha, \mathbf{g}^\beta)$, MFGP instance $(\mathbf{M_0}, \mathbf{M_1}, \mathbf{M'_{b_F}}, \mathbf{M'_{\bar{b}_F}})$, and MDHP instance $(\rho, \mathbf{h}, \mathbf{G_0}, \mathbf{G_1}, \mathbf{G^a_{b_D}}, \mathbf{G^a_{\bar{b}_D}})$.
2. Input each instance to the appropriate block.
3. Output the result provided by each block as the answer to the corresponding problem.

Now we discuss the success probability of $\mathcal{A}_D$.

Case 1 ($\mathcal{P}_{DH}$ is not negligible.)
The output from block T1 is a correct answer of CDHP if $Q_I = {G'_{b'}}^{\log_h y}$ for $b' = b_1$. This happens with probability $\frac{\mathcal{P}_{DH}}{2q_F}$ which is not negligible.

Case 2 ($\mathcal{P}_{DH}$ is negligible.)
If $\mathcal{P}_F \geq 1/2$, from Equation (1), we have

$$Pr[\tilde{\tilde{b}} = b_F] = (1 - \mathcal{P}_{DH})\{\frac{1 + 2\mathcal{P}_F}{4} + \frac{\mu_A}{2}\}$$
$$\geq (\frac{1}{2} + \frac{\mu_A}{2}) - (\frac{1}{2} + \frac{\mu_A}{2})\mathcal{P}_{DH} \geq \frac{1}{2} + \mu_{MFG}$$

for some μ_{MFG} which is not negligible. Otherwise, if $\mathcal{P}_{DH} \geq 1/2$, we have $Pr[\tilde{\tilde{b}} = b_D] \geq \frac{1}{2} + \mu_{MFG}$ for some μ_{MFG} which is not negligible. Thus, either MFGP or MDHP will be solved with an advantage that is not negligible. $\square$