A Secure and Efficient Mix-Network using Extended Binary Mixing Gate

Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee

Information Security Research Centre,
Queensland University of Technology,
GPO BOX 2434, Brisbane, QLD, 4001, Australia

Abstract. A mix-network accepts a set of ciphertexts and outputs the corresponding plaintexts in a random order. It is an important tool in schemes requiring anonymity of messages, such as in secure e-voting and e-auction schemes. A mix-network is comprised of shuffling and decryption operations. A robust mix-network must provide proofs that it shuffles and decrypts its input ciphertexts and outputs their corresponding plaintexts correctly. Verifying such proofs is often a bottleneck affecting the performance of the mix-network. We propose a secure and efficient mix-network employing extended binary mixing gates (**EBMG**s) to shuffle the ciphertexts and prove the correctness of the shuffling. Batching techniques are used in the EBMGs, such that the mix-network is very efficient. The proposed mix-network offers sufficient anonymity and high performance level compared to other mix-network schemes.

Keywords: Mix-network, Efficient mix-network, Extended binary mixing gate, Batch re-encryption, Batch verification, Proof of equality of discrete logarithms.

1 Introduction

Applications such as secure e-voting and e-auction require that votes and bids are to be anonymous. Voter-vote relationships must be kept private in secret-ballot voting, and only the winning bidder is to be identified in sealed-bid auction schemes. This requirement is called anonymity, which is essential to protect the privacy of the voters or bidders. To ensure anonymity, many of the schemes employ anonymous channels in communicating messages.

An anonymous channel is a mechanism to achieve anonymity, such that outputs of the channel cannot be linked to their original sender. This is normally implemented using a mix-network. A mix-network shuffles a number of ciphertext inputs (each from one user) and provides the same number of plaintext outputs, such that:

1. outputs of the mix-network is a random permutation of the plaintexts in the input ciphertexts,
2. the random permutation is kept secret, such that input and output relationships are publicly unknown.

These two main properties of a mix-network are known as *correctness* and *privacy* respectively.

Two other important properties are *robustness* and *public verifiability*. A mix-network is robust if it manages to operate properly under anomalous conditions, such as failure of one or more of its mixing nodes. A mix-network is publicly verifiable if its correctness can be publicly verified.

As input and output relationships have to be kept secret by the mix-network, proving correctness of the mixing operation is often made complicated. Currently, there is no acceptably efficient method to check this. Hence, mix-network schemes to date do not accommodate real-world applications requiring a large number of messages to be shuffled, e.g. in a national election. Current research in this area aims at producing a secure and practical scheme for implementation in the real-world.

One approach to design a correct and private mix-network is to use the combination of multiple small-scale mixings (mixing a small number of inputs to the same number of outputs) to implement a large-scale mixings (mixing a large number of inputs to the same number of outputs) [1, 2, 24]. In Abe's work [1, 2], binary mixing gates (mixing two inputs to two outputs) are used, while larger mixing gates (the number of mixed inputs in each gate is larger than two, but much smaller than the number of mixed inputs in the whole mix-network) are employed in the scheme by Peng *et al.* [24]. Each of these two solutions has its own advantages and disadvantages. Our proposed scheme combines the advantages of these two solutions and avoids their disadvantages by using a new technique called *extended binary mixing gate* (**EBMG**). With the help of batch cryptology, mixing based on **EBMG**s achieves better trade-off between security and efficiency than [1, 2, 24]. The design is based on a re-encryption chain mix-network employing individual mix server verification.

In the new mix-network, a batch re-encryption technique is employed to improve the efficiency of re-encryption. The permutation is gate-based and simplified, such that correctness verification of the shuffling can be batched.

The remainder of the paper is organised as follows: Section 2 provides a summary of previous work in the mix-network research area. Section 3 describes batch re-encryption and batch verification of equality of logarithms of the same base. These provide two essential foundations to the proposed scheme. Section 4 details an EBMG employing ElGamal cryptosystem, as a building block to the proposed mix-network. Section 5 describes the protocol of the mix-network made up of EBMGs. The core mix-network protocol and its privacy analysis are detailed, and an alternative protocol is provided. Section 6 offers security and efficiency analysis of the proposed mix-network. Section 7 is a conclusion.

2 Related Work

Mix-networks are frequently applied to implement anonymous channels, so is needed in any application with a requirement of anonymity. It is well known that mix-networks can be employed to achieve anonymity in email systems and e-voting. Recently Peng et al [23] define relative bid privacy in e-auction and design a mix network to implement anonymous bid submission, and thus relative bid privacy.

In this section, we briefly review related work in the mix-network research area. Background on the importance of re-encryption chain mix-network is discussed, and a classification of mix-network schemes to date is provided.

2.1 Re-encryption Chain Mix-Network

A mix-network is normally composed of a number of mix servers (**mixers**) working sequentially. Each mixer obtains its inputs from the previous server, processes them, randomly permutes their ordering, and forwards the outputs to the next server. This is such that the relationships between the original inputs and the final outputs are kept secret if at least one server conceals his permutation.

According to the processing performed by the servers, mix-networks can be classified into two types: decryption chain and re-encryption chain mix-networks. In the former type, each input is sequentially encrypted by the user for the servers to decrypt. Consequently, failure of any server means that the input messages cannot be recovered if each server holds its own private key secret (as required to achieve strong privacy). Therefore, decryption chain mix-networks inherently lack robustness. Thus, we only consider re-encryption chain mix-networks employing threshold decryption in this paper.

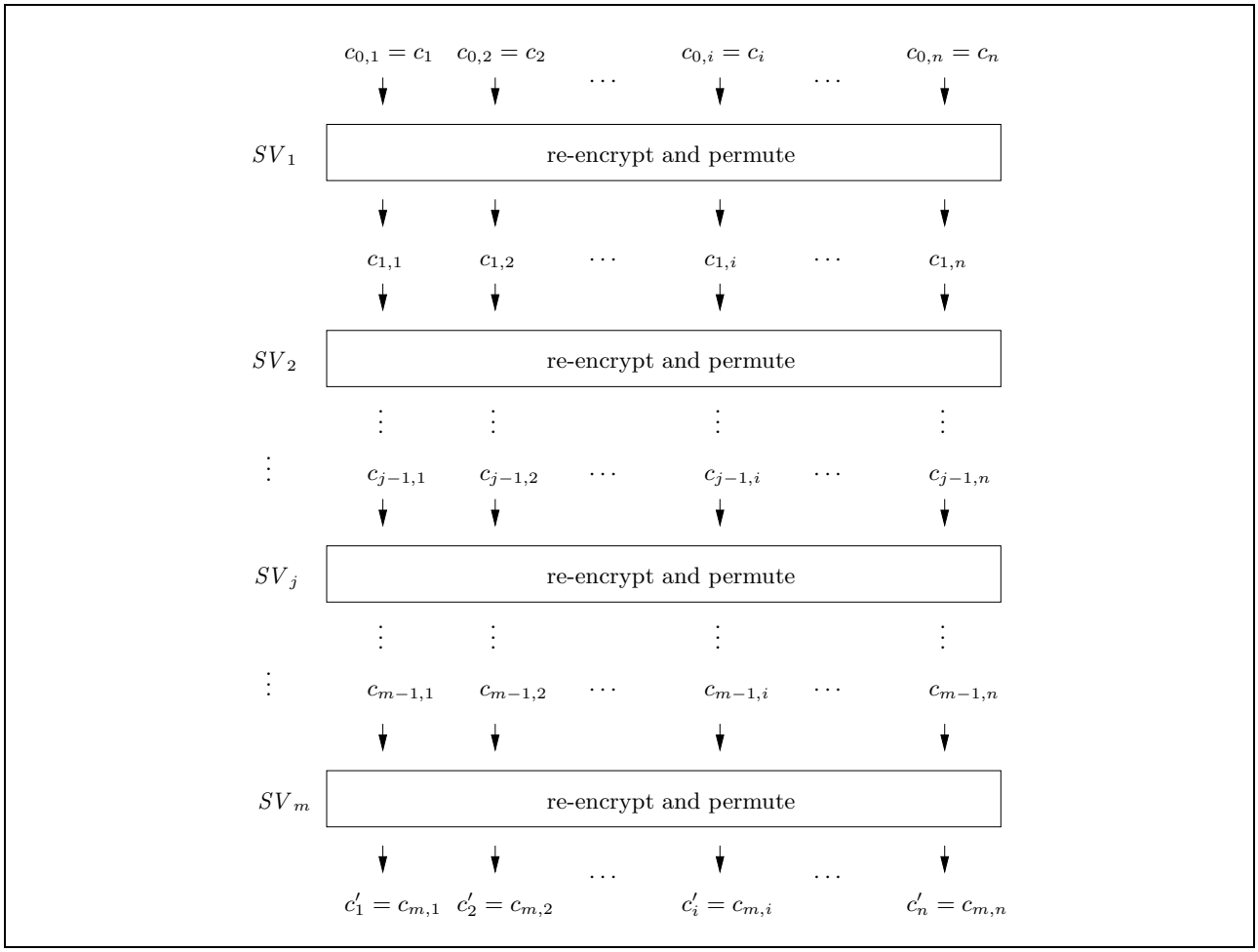

Fig. 1. Basic structure for re-encryption mix-networks introduced by Ogata *et al.* [18].

Ogata *et al.* [18] introduced a basic structure for re-encryption chain mix-networks illustrated in Figure 1. Their scheme was further developed in many later papers. Suppose an ElGamal encryption with distributed decryption is employed as in Pedersen [22]. Decryption authorities employ a distributed key generation scheme [10, 21, 12] to generate a private key x, which is shared by them. The public key is $(g, y = g^x)$. For $i = 1, \ldots, n$ and $j = 1, \ldots, m$, m servers SV_j form a mix-network to shuffle n encrypted inputs c_i. Inputs to SV_j are $c_{j-1,i}$, while $c_i = c_{0,i}$. On server SV_j, input $c_{j-1,i} = (a_{j-i,i}, b_{j-1,i})$ is re-encrypted and permuted to $c_{j,\pi_j(i)} = (a_{j,\pi_j(i)}, b_{j,\pi_j(i)}) = (g^{r_{j,i}} a_{j-i,i}, y^{r_{j,i}} b_{j-1,i})$, where the value of $r_{j,i}$ is randomly chosen, and π_j is a secret random permutation of $[1, n]$. The outputs of SV_j are $c_{j,i}$, while the final outputs $c'_i = c_{m,i}$, where $i = 1, \ldots, n$. The shuffling from n inputs to n outputs in every server is denoted as $PN(n)$, in which the correctness must be verified. Finally, a quorum of the decryption authorities (e.g. the mixers themselves) cooperate to decrypt c'_i, where $i = 1, \ldots, n$.

2.2 Verification of Correctness of the Mixing Operation

Many of the early mix-network schemes offer security and robustness in exchange for efficiency. Verification of correctness of mixing and correctness of decryption operations are often time consuming. Some of the latest schemes improve on the efficiency by sacrificing verification of correctness

of mixing and decryption operations. According to the different correctness verification mechanisms, mix-networks can be classified into three categories:

1. **No verification**: In this category, correctness is not verified and the mixers are trusted to perform the shuffling correctly. An example of this category is the scheme by Ohkubo and Abe [19]. Strong trust is necessary in such mix-networks.
2. **Global mix verification**: Mix-networks in this category do not provide a verification of correct shuffling by each mixer separately. Instead, correctness of the shuffling by the whole mix-network is verified after the final plaintext outputs are produced. Schemes in this category include [6, 20, 26, 13]. Drawbacks in this category include:
   (a) A cheating mixer cannot be identified instantly.
   (b) If an incorrect shuffling is found, a mix-network in the third category must be employed to perform re-shuffling.
   (c) Some outputs may be revealed in plaintext even when the shuffling is incorrect and a re-shuffling is needed.
3. **Individual mixer verification**: In this category [25, 15, 1, 2, 11, 18, 16, 4, 17, 14, 24], each mixer first verifies the correctness of the previous mixer's shuffling. Then, he shuffles his inputs, proves the correctness of his own shuffling, and forwards the outputs to the next mixer. Although schemes in the first two categories are more efficient, schemes in this category are still very useful as;
   (a) they overcome the shortcomings of the first two categories.
   (b) they form a necessary sub-function (to handle anomalous situations when cheating in the shuffling is found) in the second category.

   However, there exist various problems in this category. For example: the scheme by Juels and Jakobsson [15] is not publicly verifiable, some schemes [16, 4] do not provide sufficient correctness and privacy guarantee for many applications, and other schemes [1, 2, 18, 25] are inefficient. Three recently proposed schemes [11, 17, 14] offer great improvement in this category. However, these three schemes are still not optimally efficient for large-scale applications (e.g. national election) as their computational cost is linear to the number of inputs.

In the third category, a naive method to verify correctness of shuffling by SV_j is to test the following condition:

$$\begin{aligned}
&(\log_g(a_{j,\pi_{j,1}(i)}/a_{j-1,i}) = \log_y(b_{j,\pi_{j,1}(i)}/b_{j-1,i})), \text{ for } i = 1,\ldots,n.\\
\vee&(\log_g(a_{j,\pi_{j,2}(i)}/a_{j-1,i}) = \log_y(b_{j,\pi_{j,2}(i)}/b_{j-1,i})), \text{ for } i = 1,\ldots,n.\\
&\vdots\\
\vee&(\log_g(a_{j,\pi_{j,n!}(i)}/a_{j-1,i}) = \log_y(b_{j,\pi_{j,n!}(i)}/b_{j-1,i})), \text{ for } i = 1,\ldots,n.
\end{aligned} \tag{1}$$

where $\pi_{j,l}$ for $l = 1, 2 \ldots, n!$ represents all $n!$ possible permutations in the choice of π_j. This naive proof is a proof of n-out-of-$n \times n!$ equality of discrete logarithms based on the proof of equality of logarithms and the proof of partial knowledge by Cramer *et al.* [8]. This proof is honest-verifier-zero-knowledge and guarantees that the applied permutation is one of the $n!$ possible permutation, thus it guarantees correctness of the mix-network without compromising privacy. However, this test is too inefficient because the computational cost for both the prover and verifier on every mixer is $O(n \cdot n!)$ exponentiations.

An essential efficiency improvement over the naive verification was proposed in [1, 2, 24]. In these three papers, combination of multiple small-scale mixings is used to implement a large-scale mixings. In Abe's work [1, 2], an n-input-to-n-output mixing (denoted as $PN(n)$ in [1]) is divided into a

number of 2-input-to-2-output mixing (namely binary gate, denoted as $PN(2)$ in [1]). Suppose two encrypted inputs $c_1 = (a_1, b_1)$ $c_2 = (a_2, b_2)$ are encrypted and shuffled to two outputs $c'_1 = (a'_1, b'_1)$ $c'_2 = (a'_2, b'_2)$. Verification of validity of the gate is to test the condition:

$$\begin{aligned}&((\log_g(a'_1/a_1) = \log_y(b'_1/b_1)) \wedge (\log_g(a'_2/a_2) = \log_y(b'_2/b_2)))\\ \vee&((\log_g(a'_2/a_1) = \log_y(b'_2/b_1)) \wedge (\log_g(a'_1/a_2) = \log_y(b'_1/b_2)))\end{aligned} \tag{2}$$

This proof is a proof of 2-out-of-4 equality of discrete logarithms based on the proof of equality of logarithms and the proof of partial knowledge by Cramer *et al.* [8]. Its principle is the same as that of Proof (1), namely the applied permutation (either n-to-n or 2-to-2) must be proved to be one of the possible permutations. This type of proof is called *permutation-based proof*. As permutation-based proof in a binary gate (Proof (2)) is much more efficient than permutation-based proof in a large mixing (Proof (1)), Abe's mix-network is more efficient than a mix-network with the naive verification even though there are many gates needed in his scheme. The cost of Abe's mix-network is $O(n \log_2 n)$ exponentiations while a mix-network with the naive verification is $O(n \times n!)$ exponentiations.

However, Abe's schemes are still not efficient enough for many applications as $O(n \log_2 n)$ (a large number) binary gates are needed. So, Peng *et al.* [24] proposed an improved shuffling technique to further improve efficiency. Larger mixing gates (the number of mixed inputs in each gate is larger than two, but much smaller than the number of mixed inputs in the whole mix-network) are employed in [24][1] and the number of mix gates is greatly reduced. In each gate (group) mixing k inputs ($k > 2$), validity of the mixing must be proved and verified. If permutation-based proof is used, each gates requires $k \times k!$ exponentiations, which is inefficient as k is larger than 2 (e.g. $k = 20$). So permutation-based proof is not employed in [24]. Instead, a server mixing $c_1, c_2, \ldots, c_k$ to $c'_1, c'_2, \ldots, c'_k$ proves the following condition:

$$\begin{aligned}&((\log_g(a'_1/a_1) = \log_y(b'_1/b_1)) \vee (\log_g(a'_2/a_1) = \log_y(b'_2/b_1)) \vee \ldots \vee (\log_g(a'_k/a_1) = \log_y(b'_k/b_1)))\\ \wedge&((\log_g(a'_1/a_2) = \log_y(b'_1/b_2)) \vee (\log_g(a'_2/a_2) = \log_y(b'_2/b_2)) \vee \ldots \vee (\log_g(a'_k/a_2) = \log_y(b'_k/b_2)))\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots\\ \wedge&((\log_g(a'_1/a_k) = \log_y(b'_1/b_k)) \vee (\log_g(a'_2/a_k) = \log_y(b'_2/b_k)) \vee \ldots \vee (\log_g(a'_k/a_k) = \log_y(b'_k/b_k)))\end{aligned} \tag{3}$$

This type of proof is called *input-based proof*, which guarantees that each input is correctly shuffled to an output. Input-based proof is much more efficient than permutation-based proof and Proof (3) is only of $O(k^2)$ exponentiations. However, input-based proof cannot provide as strong a guarantee of correct mixing as permutation-based proof. It is explained by Peng *et al.* [24] that when two users with inputs containing the same message in the same gate (group) conspire with the first mixer, they can perform an attack to compromise correctness of the mixing.

In summary, each of the two gate-based solutions has its advantages and disadvantages. Abe's mix-network applies binary gates and permutation-based proof in each gate, so that correctness and high efficiency can be achieved in every gate. Its drawback is that too many gates are needed. In the mix-network by Peng *et al.* [24], fewer gates are needed, but high efficiency in each gate is obtained at the cost of weakened correctness. In this paper, the above two merits (from Abe and Peng *et al.*) are combined and their drawbacks are addressed to produce a better mix-network.

[1] A mixing gate is called a group in [24]. For example, in a voting involving 1000 voters, 100 groups of size 10 are employed to implement a mixing of 1000 votes.

3 Batch Cryptology

This section discusses batching techniques of re-encryption and verifying proof of equality of discrete logarithms of the same base. These techniques are employed in the proposed mix-network to improve the performance of re-encryption, and proving and verifying correctness of the mixing operation.

3.1 Batch Re-encryption in A Shuffling

In re-encryption chain mix-networks, re-encryption and permutation are performed on every server. The computational cost of a shuffling of n ciphertexts on a server is $O(n)$. In this section, the n instances of re-encryption is batched, such that the computational cost on a mixer is reduced to $O(\log_2 n)$.

Suppose an encryption algorithm $E()$ is:

- semantically secure, in which a message m is encrypted to $c = E(m, r)$, where r is a random integer;
- homomorphic, and there is an identity message I such that for any message $D(E(m)E(I)) = m$ (e.g. I is 1 in ElGamal encryption or 0 in Paillier encryption).

Ciphertexts $c_1, c_2, \ldots, c_n$ encrypted with $E()$ have to be shuffled. For simplicity, we assume that $n = 2^s$ (s indicates the bit length of n). The batch re-encryption and permutation of n inputs in a shuffling (of a mix network) is as follows, where $B_k(i)$ denotes the k^{th} bit of integer i.

1. The server randomly selects $2s$ secret integers $r_{k,j}$, for $k = 1, 2, \ldots, s$ and $j = 0, 1$.
2. The server performs $2s$ different probabilistic encryptions $R_i = E(I, \prod_{k=1}^{s} r_{k,B_k(i-1)})$ for $i = 1, 2, \ldots, n$.
3. The server calculates $c'_i = c_{\pi(i)} R_{\phi(i)}$, where $\pi()$ and $\phi()$ are random permutations of $\{1, 2, \ldots, n\}$.

As only $2\log_2 n$ random factors $r_{k,j}$, for $k = 1, 2, \ldots, s$ and $j = 0, 1$ are used in the re-encryption of n inputs, some special relations exist between a certain subset of inputs and the corresponding outputs. For example, when $n = 8$, $R(1)R(8) = R(2)R(7)$, thus $\frac{c'_{\phi(1)-1}}{c_{\pi(\phi(1)-1)}} \frac{c'_{\phi(8)-1}}{c_{\pi(\phi(8)-1)}} = \frac{c'_{\phi(2)-1}}{c_{\pi(\phi(2)-1)}} \frac{c'_{\phi(7)-1}}{c_{\pi(\phi(7)-1)}}$. Knowledge of these relations can be exploited by an attacker to compromise privacy. The attack requires a brute-force test of every possible permutation until a special relation is found. However, as two unknown random permutations $\pi()$ and $\phi()$ are involved in the shuffling, the cost of the attack is $O((n!)^2)$ exponentiations. Such attacks are not practical when n is large, e.g: $n = 1000$.

This technique of batch re-encryption can be applied to any re-encryption chain mix-network. It does not complicate validity verification of correct shuffling operation as the original verification function can still be used after batch re-encryption is employed.

3.2 Batch Verification for Equality of Logarithms of The Same Base

We describe a batch technique to batch both proof and verification of equality of logarithms of the same base. This technique is essential in batch verifying correctness of ElGamal re-encryption, providing a foundation for Section 4.

G is the subgroup of $\mathbb{Z}_p^*$ with order q, where $p - 1 = 2q$, and p, q are large primes. Let g and h be generators of G. For $i = 1, \ldots, n$, and $y_i, z_i \in \mathbb{Z}_p^*$, we need to prove and verify $\log_g y_i = \log_h z_i$. Naively, the n equations can be proved and verified separately using Chaum-Pedersen's

zero knowledge proof of equality of discrete logarithms [7]. Using the naive verification method, the computational cost for the prover is $n(2ExpCost(\langle q \rangle) + 1)$ multiplications, and for the verifier is $n(2ExpCost^2(\langle q \rangle))$ multiplications in $\mathbb{Z}_p^*$, where $\langle q \rangle$ is the bit length of q and $ExpCost^n(L)$ denotes the time to compute n exponentiations with an L-bit exponent as in [3].

Based on the batch technique by Bellare *et al.* [3] and Boyd and Pavlovski [5], Theorem 1 illustrates that these n instances of proofs can be batched, such that computational cost for the prover and verifier can be greatly improved.

Definition 1. $|\ |$ *is the absolute-value function from* Z_p^* *to* G *defined by:*

$$|\sigma| = \begin{cases} \sigma & \text{if } \sigma \in G \\ g_0^q \sigma & \text{if } \sigma \in Z_p^* \setminus G \end{cases}$$

Theorem 1 ([24]). *Suppose* $y_i \in Z_p^*$ *and* $z_i \in Z_p^*$ *for* $i = 1, 2, \ldots, n$. *Let* L *be a security parameter and* t_i *satisfying* $t_i < 2^L < q$ *for* $i = 1, 2, \ldots, n$ *be random values. If there exists* v, *such that* $1 \leq v \leq n$ *and* $\log_g |y_v| \neq \log_h |z_v|$, *then* $\log_g |\prod_{i=1}^n y_i^{t_i}| \neq \log_h |\prod_{i=1}^n z_i^{t_i}|$ *with a probability no less than* $1 - 2^{-L}$.

According to Theorem 1, for $i = 1, \ldots, n$, verification of $\log_g |y_i| = \log_h |z_i|$ can be batched as $\log_g |\prod_{i=1}^n y_i^{t_i}| = \log_h |\prod_{i=1}^n z_i^{t_i}|$. Using this batching technique, the computational cost for the prover is $2ExpCost(\langle q \rangle)+n+1$ multiplications, and for the verifier is $4ExpCost(\langle q \rangle)+2ExpCost^n(L)+$ 2 multiplications. The probability that $\log_g |\prod_{i=1}^n y_i^{t_i}| = \log_h |\prod_{i=1}^n z_i^{t_i}|$ is correct while $\log_g |y_i| \neq \log_h |z_i|$ is no more than 2^{-L}.

4 Extended Binary Mixing Gate

Using the batch re-encryption technique in Section 3.1 and the batching theorem from Section 3.2, we detail an Extended Binary Mixing Gate (**EBMG**) using ElGamal re-encryption in this section. The proposed mix-network is comprised of EBMGs and the mix-network protocol is described in Section 5.

A normal binary mixing gate (as in Abe's scheme [1]) mixes two inputs to two outputs by re-encrypting the inputs and randomly permutes the ordering of its output. The mixing is required to be:

- correct: the plaintexts of the outputs are permutation of the plaintexts of the inputs; and
- private: the permutation must be kept secret.

An EBMG shuffles $2k$ inputs $c_1, c_2, \ldots, c_{2k}$, to $2k$ outputs $c'_1, c'_2, \ldots, c'_{2k}$ by re-encrypting the inputs and partially permuting them. Illustrated in Figure 2, the mixing by an EBMG must be:

- correct: for $i = 1, \ldots, 2k$, and a decryption function D, the mixing is correct if $D(c_i) = D(c'_i)$ or $D(c_i) = D(c'_{i+k \bmod 2k})$; and
- private: for $i = 1, \ldots, 2k$, whether $D(c_i) = D(c'_i)$ or $D(c_i) = D(c'_{i+k \bmod 2k})$ is not revealed.

Either ElGamal or Paillier cryptosystem may be employed to encrypt and re-encrypt the input messages, and decrypt the output messages in a threshold manner. We use ElGamal cryptosystem as an example and detail the EBMG protocol accordingly below.

ElGamal cryptosystem is employed with the public key of (p, g, y) and input messages $c_i = (a_i, b_i)$, where $i = 1, \ldots, 2k$, and $2k$ is the number of input messages. The mixing are shuffled as follows.

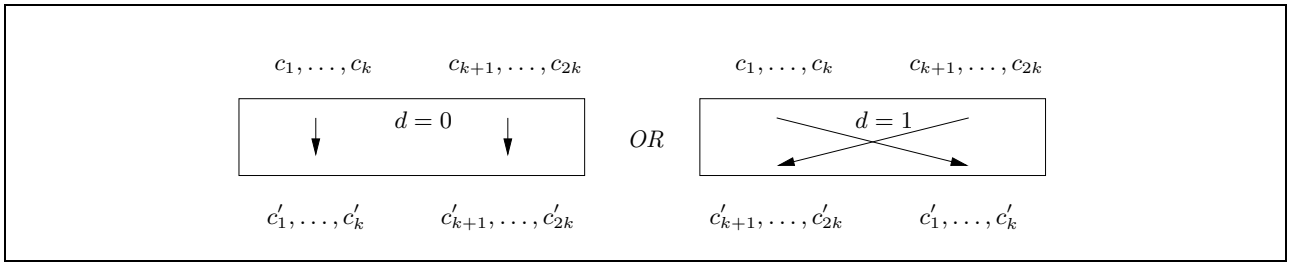

Fig. 2. Inputs to outputs permutation in an EBMG.

1. Randomly select a bit d as a switching variable (Figure 2).
2. For $i = 1, \ldots, 2k$, output $c'_j = (a'_j, b'_j) = (a_i g^{r_i} \bmod p, b_i y^{r_i} \bmod p)$, where $j = i + kd \bmod 2k$.
3. Prove correctness of the mixing operation by giving the following permutation-based proof:

$$(\log_g \left| \frac{a'_i}{a_i} \bmod p \right| = \log_y \left| \frac{b'_i}{b_i} \bmod p \right|) \ \vee \ (\log_g \left| \frac{a'_{i+k \bmod 2k}}{a_i} \bmod p \right| = \log_y \left| \frac{b'_{i+k \bmod 2k}}{b_i} \bmod p \right|)$$
$$\text{for } i = 1, \ldots, 2k \tag{4}$$

The computational cost for proving Equation (4) is $6k$ full-length exponentiations, and the corresponding verification costs $8k$ full-length exponentiations. When k is large, the computational cost increases accordingly.

Theorem 1 (Section 3.2) is applied to batch verify the condition in (4) as:

$$(\log_g \left| \prod_{i=1}^{2k} \left(\frac{a'_i}{a_i} \right)^{t_i} \bmod p \right| = \log_y \left| \prod_{i=1}^{2k} \left(\frac{b'_i}{b_i} \right)^{t_i} \bmod p \right|) \vee \tag{5}$$
$$(\log_g \left| \prod_{i=1}^{2k} \left(\frac{a'_{i+k \bmod 2k}}{a_i} \right)^{t_i} \bmod p \right| = \log_y \left| \prod_{i=1}^{2k} \left(\frac{b'_{i+k \bmod 2k}}{b_i} \right)^{t_i} \bmod p \right|)$$

Using the batch technique, computational cost for proving Equation (5) is 6 full-length exponentiations, and the corresponding verification costs 8 full-length exponentiations. We ignore exponentiation cost using an L-bit exponents (e.g. $\langle L \rangle = 20$ bits) since the computational cost is much less compared to full-length exponentiations (e.g. $\langle q \rangle = 1024$ bits) and is typically smaller than 5% of the entire cost.

5 The Mix-Network Protocol

Our proposed scheme combines the advantages of the two mix-networks in [1, 2] and [24] respectively and avoids their disadvantages by using a new technique called **EBMG** (see Section 4). An EBMG can mix more than two inputs, such that not many gates are necessary. The number of EBMGs needed in our mix-network is n, which is much fewer than the number of gates in [1, 2]. In each EBMG, although more than two inputs are mixed, only two permutations are possible. So permutation-based proof of correctness of mixing can be applied to achieve strong correctness. Namely, permutation-based proof, a merit of [1, 2] and simple structure, a merit of [24] are combined. As a result, strong correctness can be achieved efficiently. Although the number of possible permutation in our mix-network is reduced, each of its inputs is equally likely to be shuffled to any of its outputs. So the privacy of any single input (the desired property of privacy in many applications) is still strong in the new mix-network.

The new mix-network consists of a number of mixers, each in charge of a level of EBMGs. For simplicity, we suppose the number of inputs to the mix-network n is a power of 2.

5.1 The Core of The Mix-Network

There are a total of $m = \log_2 n$ levels, where each mixer is responsible for one level. In the j^{th} level, there are $2^{-j}n$ EBMGs, where $j = 1, \ldots, m$. The number of EBMGs required in the mix-network core is $n - 1$. This is illustrated in Figure 3. As the EBMGs can employ either ElGamal (Section 4)

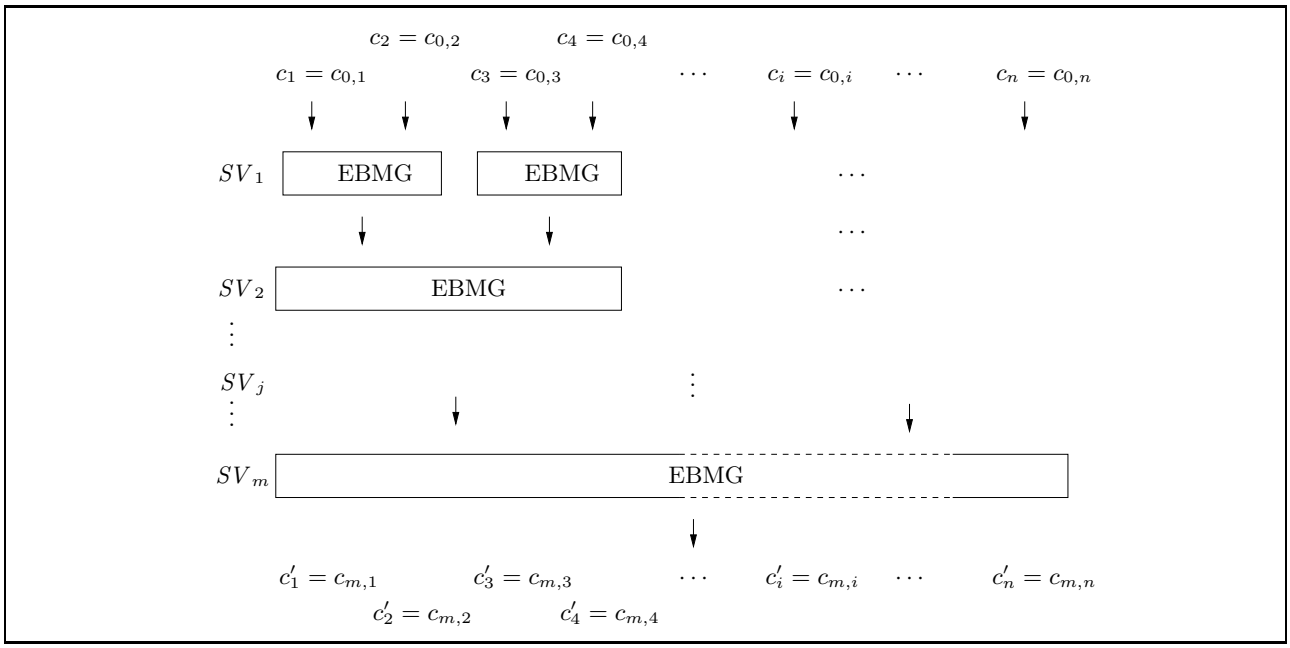

Fig. 3. Core EBMG structure in the proposed mix-network.

or Paillier re-encryption, the proposed mix-network can either use threshold version of ElGamal [22] or Paillier [9] cryptosystem. Batch re-encryption technique in Section 3.1 is applied at each mixers (levels in Figure 3), and batch verification technique in Section 3.2 is applied at each EBMG in the core mix-network. The core mix-network protocol is detailed as follows:

1. The first level
   The inputs $c_{0,1}, \ldots, c_{0,n}$ are batch re-encrypted as in Section 3.1. Then they are divided into $\frac{n}{2}$ pairs in the first mixer (first level of EBMGs; refer to SV_1 in Figure 3). For $i = 1, \ldots, \frac{n}{2}$, every two successive inputs of $c_{0,2i-1}$ and $c_{0,2i}$ are shuffled to $c_{1,2i-1}$ and $c_{1,2i}$ by an EBMG. The shuffled outputs $c_{1,1}, \ldots, c_{1,n}$ of the first mixer SV_1 are then sent to the second level SV_2 to be its inputs.
2. The l^{th} level
   Inputs $c_{l-1,1}, \ldots, c_{l-1,n}$. For $i = 1, \ldots, \frac{n}{2^l}$ are batch re-encrypted as in Section 3.1. Then every 2^l successive inputs $c_{l-1,2^l(i-1)+1}, c_{l-1,2^l(i-1)+2}, \ldots, c_{l-1.2^l i}$ are mixed to $c_{l,2^l(i-1)+1}, c_{l,2^l(i-1)+2}, \ldots, c_{l.2^l i}$ by one EBMG. There are altogether $\frac{n}{2^l}$ EBMGs in level l. The outputs $c_{l,2^l(i-1)+1}, c_{l,2^l(i-1)+2}, \ldots, c_{l.2^l i}$ of this l^{th} level are forwarded as inputs to be shuffled in the $l + 1^{th}$ level.
3. Each output of the last level of mixing is verified to be in G. Any output not in G is changed to its absolute value (see Definition 1).

4. The final output ciphertexts are decrypted in a threshold manner by some decryption authorities (e.g. the mixers themselves).

5.2 Ciphertexts Distances

Illustrated in Figure 2, the value of the switching variable d determines the output positions of the ciphertexts processed in an EBMG. Structured as in Figure 3, the final output positions of the ciphertexts shuffled by the core mix-network are determined by the values of the switching variables, each in the EBMGs processing the ciphertexts.

A user with a unique input in c_i can identify his own message output by the core mix-network (in plaintext after c'_i is decrypted) and identify the values of switching variables in every EBMG the ciphertext went through[2]. Using this information, the user can further identify the initial input positions of the output plaintexts (after the final output ciphertexts are decrypted) with a certain probability in relation to the distances between the ciphertexts. This is because after going through the core shuffling, although all input ciphertexts are output to a different output position, the distances between those ciphertexts (Definition 2) are constant.

Definition 2. *The function $\Delta(c_a, c_t)$ denotes the distance of the target ciphertext c_t from the anchor ciphertext c_a. The value of $\Delta(c_a, c_t) = (\log_2 \varepsilon(c_a, c_t)) - 1$, where $\varepsilon(c_a, c_t)$ denotes the number of elements in the smallest set of 2^k successive ciphertexts $\{c_i, c_{i+1}, \ldots, c_{i+2^k-1}\}$ containing c_a and c_t, the value of $i = \mu 2^k + 1$, and μ is an integer.*

For n number of inputs, there are $\log_2 n$ bits of switching variables determining the final output positions of each ciphertexts. Each of the bits indicates the value of each switching variable in each EBMG that the ciphertext went through, where the most significant bit indicates the value of the switching variable in the EBMG in SV_m, and $m = \log_2 n$.

For n number of inputs, and $1 \leq a, t \leq n$, a malicious user with a unique plaintext input in c'_a can identify the initial input position t of the final ciphertext output c'_t from $2^{\Delta(c'_a, c'_t)}$ possible number of initial input ciphertexts. This is because $\Delta(c_a, c_t) = \Delta(c'_a, c'_t)$. The malicious user requires $\Delta(c'_a, c'_t)$ bits of switching information on the final ciphertext output of c'_t to determine the initial input position of c'_t.

Since the smallest possible distance Δ between two ciphertexts is 0, shuffling using the core mix-network only achieves pairwise privacy. Furthermore, the privacy of the core mix-network can be compromised when half of the users collaborate, such that each of the ciphertexts input by the honest user is an immediate neighbour (have the minimum distance of $\Delta = 0$) of one of the ciphertexts input by the malicious users. The best case scenario for an honest user is when the distance between his ciphertext c_t and the ciphertext of a malicious user c_a is maximum, where $\Delta(c_a, c_t) = (\log_2 n) - 1$.

An attack scenario with 8 input ciphertexts and one malicious user is as follows. Assume $n = 8$, where a malicious user submits a unique input c_1 into the core mix-network. As input c_2 is in the same pair with c_1, their distance is $\Delta(c_1, c_2) = 0$. Thus, the malicious user can successfully identify the user with input c_2. We call c_2 the immediate neighbour of c_1 as they have the minimum possible distance. The value of $\Delta(c_1, \{c_3, c_4\}) = 1$, and $\Delta(c_1, \{c_5, \ldots, c_8\}) = 2$. Thus, the malicious user can identify the senders of c_3 and c_4 as one of two users, and identify the senders of $\{c_5, \ldots, c_8\}$ as one of four users. This is because the malicious user can only guess the switching position d in EBMG

[2] The user can trace-back and deduce the value of each switching variable d in each EBMG that his ciphertext went through, from the EBMG in SV_m to the EBMG in SV_1. Thus, the switching position in those EBMGs are revealed.

in SV_1 for c_3 and c_4, and need to guess the switching position d in EBMG in SV_2 and SV_1 for $\{c_5, \ldots, c_8\}$. Note that $\{c_5, \ldots, c_8\}$ have the maximum possible distance to c_1.

When the number of choices for the input messages is very small compared to the number of the input n (e.g. in a "yes/no" voting system), the probability for any input to be unique is negligible. Thus, the privacy of the mix-network can be achieved in most cases. However, when the number of choices for the input messages is not small enough, the probability for an input to be unique is not negligible anymore. Then, the previously described attack scenario is feasible. In this case, a solution is required to overcome this problem. We provide a method to prevent the attack using two rounds of shuffling in the next subsection.

5.3 Two Rounds of Shuffling

Limiting the allowable input format is not acceptable for schemes requiring more flexible input format, such as in e-auction schemes with unspecified threshold of biddable price, or in e-voting schemes using preferential system. Two rounds of mixing are required to sufficiently alter the relative positions of the ciphertexts, such that the ciphertexts are in maximum distances to each other.

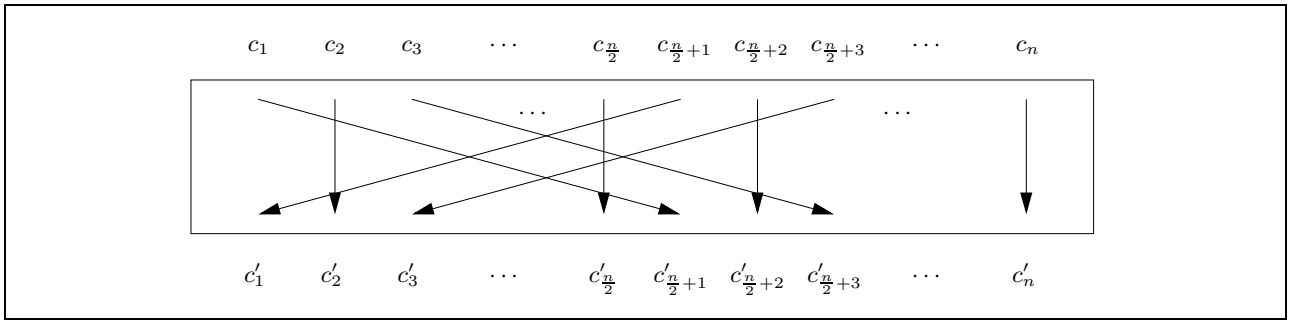

Fig. 4. Public fixed n-to-n permutation replacing adjacent messages for maximum distances in each pair of ciphertexts.

We employ two rounds of mixing and a public fixed n-to-n permutation in between the two rounds. The public n-to-n permutation (Figure 4) permutes the ciphertexts, such that the final output distance in any two consecutive ciphertexts in the pair is maximum. The two rounds of mixing protocol are as follows:

1. Input ciphertexts are shuffled as in the core mixing protocol (Section 5.1). The output ciphertexts of the first round of mixing are directly forwarded to the public fixed n-to-n permutation.
2. For n ciphertexts c_i, the public fixed n-to-n permutation outputs $c_j = c_{(i+k(i \bmod 2)) \bmod n}$, where $i = 1, \ldots, n$.
3. The output ciphertexts of the public fixed n-to-n permutation are forwarded as inputs to the second round of mixing, shuffled as in the core mixing protocol (Section 5.1).
4. The final output ciphertexts are decrypted in a threshold manner by some decryption authorities (e.g. the mixers themselves).

The final output ciphertexts are not decrypted at the end of the first round, and are directly forwarded to the public fixed n-to-n permutation before being submitted as inputs to the second round. A malicious user can only guess the initial position of the message at the second round. Thus, the best case scenario for a malicious user is to guess the initial position of his final ciphertext output

pair from the other half of the initial input ciphertexts with a probability of $\frac{2}{n}$. The probability of a malicious user successfully identifying the initial input ciphertext position of any of the other ciphertexts is $\frac{1}{n-1}$. A collaboration of $n-1$ malicious users is required to successfully compromise the privacy of the mix-network.

6 Analysis

Security and efficiency of the proposed mix-network is analysed in this section. Correctness and privacy level of the mix-network are discussed. Privacy and computational cost comparison of the proposed scheme and other efficient schemes is also provided.

6.1 Security

Mixing in each EBMG in the new mix-network is correct as its proof of correctness is permutation-based and the batch verification technique in Section 3 fails with a negligible probability. Thus, the new mix-network is correct with a very large probability even if some users and mixers collude to attack. This is an advantage over the scheme by Peng *et al.* [24].

As the batch verification technique is honest-verifier-zero-knowledge, each EBMG is private. Privacy of the new mix-network depends on two factors: security of batch re-encryption in Section 3.1 and the diffusion of the inputs.

- Security of batch re-encryption
  As stated in Section 3.1, the complexity of the mentioned brute-force attack against privacy is the product of two permutations: the shuffling permutation $\pi()$ and the random-factor-distribution permutation $\phi()$. As extended binary gates are applied, the complexity of $\pi()$ on the j^{th} level is reduced to $2^{n/2^j}$. However, the complexity of $\phi()$ is still $n!$ as the re-encryption is level-based. For a large n, the brute-force attack to guess the permutation and compromise the privacy of the shuffling (as described in Section 3.1) is impractical.
- Diffusion of the inputs
  Any input to the mix-network may be shuffled to any of the n outputs in the mix-network. Moreover, for any input, the n possible shuffling are equally likely. Thus, diffusion of any single input is optimally achieved.
  Since there are only two possible permutations in each extended binary gates, diffusion of all the inputs as a whole is not complete. After the core mixing process has concluded, a malicious user with a unique input in c_a can identify the message of his pair input shuffled by the same EBMG in SV_1. Furthermore, collaboration of malicious users can further weaken the privacy of the core mix-network. We refer to Section 5.2 and Section 5.3 for a detailed analysis and two alternatives to alleviate this problem.
  The number of possible permutations using the core mix-network is 2^{n-1}, where each permutation is equally likely. Two rounds of mixing achieves $2^{2(n-1)}$ possible permutations. Although the proposed mix-network does not offer all possible permutations, we consider the privacy level to be strong enough for many applications requiring a large number of messages to be communicated anonymously, i.e. where n is large.

Table 1 compares the privacy of our proposed scheme and other high-performance mix-network schemes. The degree of privacy is measured in terms of diffusion offered by the mix-network. A mix-network with perfect privacy has $n!$ permutations, each of them equally likely. It is demonstrated that the new scheme achieves strong enough privacy for most applications.

Table 1. Comparison of privacy level in terms of diffusion achieved, where ϵ indicates the number of honest mixers in a t-out-of-m threshold cryptosystem and k is the size of a gate (group).

Mix-network scheme	One input	All inputs	Uniform
Abe [1]	1 among n if $\epsilon > t$	$n!$ permutations if $\epsilon > t$	no
Abe & Hoshino [2]	1 among n if $\epsilon > t$	$n!$ permutations if $\epsilon > t$	yes
Furukawa & Sako [11]	1 among n	$n!$ permutations	yes
Neff [17]	1 among n	$n!$ permutations	yes
Groth [14]	1 among n	$n!$ permutations	yes
Peng *et al* [24]	1 among n if $\epsilon > t$	$(k!)^{\epsilon}$ permutations	yes
Our scheme (core mix-network)	1 among n	2^{n-1} permutations	yes
Our scheme (two rounds)	1 among n	$2^{2(n-1)}$ permutations	yes

In the proposed mix-network, if one mixer is compromised or the mixing for that mixer is revealed, the number of possible shuffling for an input and the number of possible permutations in the mix-network is reduced by half (i.e. reveal the values of switching variables as in Section 5.2), while the rest of possible shuffling and permutations are still equally likely. Addressing this problem to achieve a stronger privacy, the entire mixing process can be repeated a number of times sufficiently (i.e. extend from Section 5.3).

6.2 Performance

The computational cost of one mixing operation (core mix-network) is as follows:

- Re-encryption: $4(\log_2 n)^2$ full-length exponentiations.
- Proof of valid mixing: $6(n-1)$ full-length exponentiations.
- Verification of valid mixing: $10(n-1)$ full-length exponentiations.

Table 2. Comparison of correctness computational cost for shuffling the ciphertexts in full-length exponentiations.

Mix-network scheme	Correctness	Mixing	Verification of correct mixing
Abe [1]	Strong	$> 16(n \log_2 n - n + 1)$	$> 16(n \log_2 n - n + 1)$
Furukawa & Sako [11]	Strong	$40n$	$40n$
Groth [14]	Strong	$32n + 12n/\kappa + 12$	$24n + 12n/\kappa + 24$
Peng *et al* [24]	Weaker	$8n + 4k(4k-2)$	$16k^2$
Our scheme (one round)	Strong	$4(\log_2 n)^2 + 6(n+1)$	$10(n-1)$
Our scheme (two rounds)	Strong	$8(\log_2 n)^2 + 12(n+1)$	$20(n-1)$

We compare correctness and computational cost of the proposed mix-network against other mix-network schemes currently considered efficient. The comparison figure is summarised in Table 2 based on the use of ElGamal cryptosystem, where we assume 4 mixers are used in the mix-network [11, 17, 14], n is the number of inputs to the mix-network, k is the size of group in the scheme by Peng *et al.* [24] and κ is a parameter smaller than n.

In the mix-network scheme by Abe [1], we only provide a lower threshold of its computational cost as the concrete number of gates was not provided. The mix-network by Neff [17] is not included

in the table as the protocol is not provided in great detail. However, its performance should be similar to the mix-network scheme by Groth [14], as they employ similar techniques.

Research in mix-network schemes mainly focus on improving the efficiency of verifying correct mixing operation. Although the performance gain is not significant, the batch re-encryption technique in Section 3.1 can also be implemented in other re-encryption chain mix-network schemes to improve their overall efficiency.

From Table 2, our proposed mix-network scheme achieves a better trade-off between correctness and efficiency. The only mix-network more efficient than ours is the scheme by Peng *et al.* [24], which has weaker correctness.

7 Conclusion

The new mix-network obtains the best trade-off between correctness, privacy and high efficiency. Extended Binary Mixing Gates (**EBMG**s) is applied to achieve high efficiency. Permutation-based proof of mixing correctness is employed to guarantee high level of correctness. Although the number of possible permutations is reduced as a result of batch verification, privacy of the mix-network (especially privacy of any single input) is still strong.

The proposed batch re-encryption technique can be implemented in other re-encryption chain mix-network schemes to increase their performance.

8 Acknowledgements

We acknowledge the support of the Australian government through ARC Discovery 2002, Grant No: DP0211390; ARC Discovery 2003, Grant No: DP0345458; and ARC Linkage International fellowship 2003, Grant No: LX0346868.

References

1. Masayuki Abe. Mix-networks on permutations networks. In *Advances in Cryptology—ASIACRYPT 99*, pages 258–273, 1999.
2. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography, 4th International Workshop—PKC 01*, pages 317–324, 2001.
3. Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology—EUROCRYPT 98*, pages 236–250, 1998.
4. Dan Boneh and Philipe Golle. Almost entirely correct mixing with applications to voting. In *ACM Conference on Computer and Communications Security—CCS 02*, pages 68–77, 2002.
5. Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In *Advances in Cryptology—ASIACRYPT 00*, pages 58–71, 2000.
6. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.
7. David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology—CRYPTO 92*, pages 89–105, 1993.
8. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO 94*, pages 174–187, 1994.
9. Ivan Damgaård and Mats Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography—PKC 01*, pages 119–136, 2001.
10. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437. IEEE, 1987.
11. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology—CRYPTO 01*, pages 368–387, 2001.
12. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT 99*, pages 295–310, 1999.

13. Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology—ASIACRYPT 02*, pages 451–465, 2002.
14. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography, 6th International Workshop—PKC 03*, pages 145–160, 2003.
15. Markus Jakobsson and Ari Juels. An optimally robust hybrid mix network. In *20th ACM Symposium on Principles of Distributed Computing—PODC 01*, pages 284–292, 2001.
16. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium*, pages 339–353, 2002.
17. C. Andrew Neff. Verifiable, secret shuffles of elgamal encrypted data for secure multi-authority elections. In *8th ACM Conference on Computer and Communications Security—CCS 01*, pages 116–125, 2001.
18. Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In *Information Security and Cryptology, ICISC 1997*, pages 440–444, 1997.
19. Miyako Ohkubo and Masayuki Abe. A length-invariant hybrid mix. In *Advances in Cryptology—ASIACRYPT 00*, pages 178–191, 2000.
20. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology—EUROCRYPT 93*, pages 248–259, 1993.
21. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology—EUROCRYPT 91*, pages 522–526, 1991.
22. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO 91*, pages 129–140, 1992.
23. Kun Peng, Colin Boyd, Edward Dawson, and Kapali Viswanathan. Efficient implementation of relative bid privacy in sealed-bid auction. In *The 4th International Workshop on Information Security Applications, WISA2003*, Berlin, 2003. Springer-Verlag.
24. Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. A correct, private, and efficient mix network. In *Public Key Cryptography, 7th International Workshop—PKC 04*, pages 439–454, 2004.
25. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *Advances in Cryptology—EUROCRYPT 95*, pages 393–403, 1995.
26. Kapali Viswanathan, Colin Boyd, and Ed Dawson. A three phased schema for sealed bid auction system design. In *Advances in Cryptology—ACISP 00*, pages 412–426, 2000.