A Correct, Private and Efficient Mix Network

Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan

ISRC, IT Faculty, Queensland University of Technology
2 George Street, Brisbane, QLD 4001, Australia
{k.peng, c.boyd, e.dawson, k.viswanathan}@qut.edu.au
http://www.isrc.qut.edu.au

Abstract. A mix network achieving strong correctness and privacy is proposed. The degree of correctness and privacy are precisely stated and a formal proof of correctness is given. A grouping function is employed to achieve stronger correctness and higher efficiency without compromising strong privacy. In order to further improve the efficiency of the mix network a new batch verification technique, suitable for verifying multiple proofs of knowledge, is presented together with a formal proof of soundness.

Keywords: re-encryption mix network, shuffling, batch verification

1 Introduction

Mix networks are important tools to implement anonymity and are widely employed in many cryptographic applications such as e-voting and e-auctions. Since the original proposal of Chaum [6] many mix networks have been proposed in the research literature. However, most of them are inefficient, vulnerable, or limited to some special applications. Abe [1] introduced the idea of improving efficiency by dividing a costly large-scale verification operation into a few efficient small-scale verification operations. In this paper, we use Abe's idea in a new way to design a mix network with several novel features and avoiding some shortcomings of Abe's scheme. Our final proposal is simpler and more efficient than Abe's mix network, and also more efficient than other mix networks employing verification of shuffling on each server (e.g.[8, 14, 10]), especially when a large number of values are shuffled. Unlike other schemes, the new proposal achieves correctness and privacy more clearly and precisely. Therefore, our scheme is more suitable for many applications.

We divide the explanation of the new mix network into three stages. First a prototype Mix-1 is proposed, which employs a new verification mechanism to achieve formally proved correctness. Then Mix-1 is optimised to Mix-2 by adopting a grouping function. Compared to Mix-1, Mix-2 improves efficiency, strengthens correctness, and maintains strong privacy. Finally, a formally proved batch verification technique is applied to optimize Mix-2 to Mix-3, the final protocol achieving even higher efficiency.

The remainder of this paper is structured as follows. In section 2, previous work on mix networks is introduced. In section 3, a new method of correctness verification and a new batch verification technique are proposed. In section 4, the new mix network is presented. In section 5, security and other properties of the proposed mix network are analysed. Section 6 is a conclusion.

Parameter settings in the remainder of this paper are as follows.

- Let q and $p = 2q + 1$ be large primes. G is the cyclic subgroup of Z_p^* with order q. Let g and h be generators of G. ElGamal encryption algorithm is applied on G with private key $x \in Z_q$ and public key $(g, y = g^x)$. In this paper, when an ElGamal ciphertext (a, b) is presented for decryption, $a \in G$ and $b \in G$ are not checked. If $a \in Z_p^*$ and $b \in Z_p^*$, the ciphertext is decrypted and the decryption result is only guaranteed to be in Z_p^*.
- There are n users and m servers in the mix network. The number of honest servers is ϵ. If secret sharing is performed among the servers, the threshold is t (usually $m = 2t + 1$).

2 Related Work

A mix network shuffles a number of ciphertext inputs, each from one user, to the same number of plaintext outputs, so that 1) the outputs are a permutation of the plaintexts of the inputs; 2) the permutation between the inputs and the outputs is unknown, so that the users cannot be linked to their outputs. These two properties are called *correctness* and *privacy.* A mix network achieves *robustness* if it can still work properly in abnormal situations, such as failure of one or more switching nodes. A mix network is *publicly verifiable* if its correctness can be publicly verified. A mix network is usually composed of a few servers, working in sequence. Each server gets its inputs from the previous server and randomly permutes them to a set of outputs, which are inputs to the next server.

According to the processing performed by the servers, mix networks can be classified into two types: decryption chain mix networks and re-encryption mix networks. In the former type each input is sequentially encrypted for each server by the user. Consequently failure of any server means that the input message cannot be recovered if each server keeps his private key secret as required to achieve strong privacy. Therefore decryption chain mix networks inherently lack robustness. Only re-encryption mix networks are discussed further in this paper.

Ogata *et al.* [15], introduced a basic structure for re-encryption mix networks, which was further developed in many later papers. Suppose ElGamal encryption scheme is employed with private key x and public key $(g, y = g^x)$. Several decrypting authorities share x by t-out-of-m threshold verifiable secret sharing. The m servers SV_j for $j = 1, 2, \ldots, m$ form a mix network to shuffle n encrypted inputs c_i for $i = 1, 2, \ldots, n$. Inputs to SV_j are $c_{j-1,i}$ for $i = 1, 2, \ldots, n$ while $c_{0,i} = c_i$ for $i = 1, 2, \ldots, n$. Outputs of SV_j are $c_{j,i}$ for $i = 1, 2, \ldots, n$. On server SV_j, input $c_{j-1,i} = (a_{j-1,i}, b_{j-1,i})$ is permuted to $c_{j,\pi_j(i)} = (a_{j,\pi_j(i)}, b_{j,\pi_j(i)}) = (g^{r_{j,i}} a_{j-1,i}, y^{r_{j,i}} b_{j-1,i})$ where $r_{j,i}$ is randomly chosen and π_j is a secret random permutation of $\{1, 2, \ldots, n\}$. The outputs of the mix network are $c'_i = c_{m,i}$ for $i = 1, 2, \ldots, n$. The shuffling from n inputs to n outputs on every server is denoted as $PN(n)$, correctness of which must be verified. Finally, the decrypting authorities (e.g. the servers themselves) cooperate to decrypt c'_i for $i = 1, 2, \ldots, n$.

Mix networks can be further classified into three categories according to the different correctness verification mechanisms.

- In the first category, correctness is not verified and the servers are trusted to perform the shuffling correctly. Ohkubo and Abe [16] designed an example in this category. Strong trust is necessary in such a mix network.
- Mix networks in the second category do not provide a verification of correct shuffling by each server separately. Instead, correctness of the shuffling by the whole mix network is verified after the mix network outputs the shuffled results in plaintexts. Several published schemes fall into this category [6, 17, 19, 9]. Drawbacks of this category include 1) a cheating server cannot be identified instantly; 2) in case of verification of incorrect shuffling, a mix network in the third category must be employed to perform the shuffling again; 3) some outputs may be revealed in plaintext even when the shuffling is incorrect and a re-shuffling is needed.
- In the third category [18, 13, 1, 2, 8, 15, 12, 4, 14, 10] each server verifies correctness of the previous servers' shuffling before performing its own shuffling and proves that its own shuffling is correct before sending them to the next server. Although the schemes in the first two categories are more efficient, the third category is still very useful because
  1. it overcomes the shortcomings of the first two categories;
  2. it is a necessary sub-function (to deal with the abnormal situation when cheating in the shuffling is found) in the second category.

  However, in this category, various problems exist: [13] is not publicly verifiable; the guarantee for correctness and privacy is not strong enough for many applications [12, 4]; [1, 2, 15, 18] are inefficient. Among them, three recently proposed schemes [8, 14, 10] are best. However, these three schemes are still not efficient enough for large-scale applications (e.g. national voting) as their computational cost is linear to the number of inputs.

In the third category, Abe's scheme [1] has a particularly useful feature which is an efficiency improvement on the following naive mix network. Let $\pi_{j,l}$ for $l = 1, 2, \ldots, n!$ be all the $n!$ possible permutations for π_j. A naive method to verify correctness of shuffling by SV_j is to test the following equation.

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

This verification allows correctness to be proved without breaching privacy. Zero knowledge proof of 1-out-of-$n!$ equality of logarithms can be applied to implement (1), based on the zero knowledge proof of partial knowledge by Cramer *et al* [7].

This test is very inefficient because the computational cost for both the prover and verifier on every server is $O(n \cdot n!)$ exponentiations. So Abe improved its efficiency by dividing a n-input-to-n-output mixing (denoted as $PN(n)$ in [1]) into a few 2-input-to-2-output mixing (denoted as $PN(2)$ in [1]). However, Abe's schemes are still not efficent enough for many applications. Our proposal is to design a re-encryption mix network employing correctness verification per server. The new scheme overcomes the shortcomings of Abe's schemes [1, 2], while retaining the idea that efficiency can be saved by dividing a large-scale correctness verification into several small-scale correctness verifications. It achieves higher computational efficiency than that of [8, 14, 10] in that the computational cost is independent of the number of users, but determined by the extent of correctness and privacy required by a certain application.

3 Preliminary Work

In this section we introduce the building blocks used to construct our mix network. We first propose a new method for shuffling verification in a mix network and prove that it is sufficient to guarantee validity of the shuffling. Then we present a new batch verification technology to improve efficiency of simultaneous proofs of equality of logarithms, which appear in the verification of the shuffling.

3.1 Improvement on the Naive Verification Technique

Although naive verification by Equation (1) can explicitly guarantee the correctness of SV_j's shuffling, it is too inefficient to be practical. A more efficient verification technique uses the following equation.

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

Equation (2) must be proved with zero knowledge proof of 1-out-of-n equality of logarithms. The computational cost of proof and verification of this equation is $n(4n-2)$ and $4n^2$ exponentiations respectively. The zero knowledge proof of Equation (2) by SV_j is denoted by CV *(correctness verification)* in the rest of this paper.

It is proved in Theorem 1 that CV is enough for the correctness verification.

Definition 1 *$SV_j(c_{j-1,\mu}, c_{j,\nu}) = 1$ means SV_j knows $r_{j,\nu}$ satisfying $a_{j,\nu} = g^{r_{j,\nu}} a_{j-1,\mu}$ and $b_{j,\nu} = y^{r_{j,\nu}} b_{j-1,\mu}$.*

Theorem 1. *If the shuffling by SV_j is incorrect, CV can be satisfied with a probability no more than $1/q$ without collusion of all the previous $j-1$ servers and at least two users, assuming DL problem is intractable.*

To prove Theorem 1, the following lemma is used.

Lemma 1. *If the shuffling by SV_j is incorrect and for every $c_{j-1,\mu}$ with $1 \leq \mu \leq n$ there exists some $c_{j,\nu}$ with $1 \leq \nu \leq n$ such that $SV_j(c_{j-1,\mu}, c_{j,\nu}) = 1$, then SV_j knows $\log_g a_{j-1,i'} - \log_g a_{j-1,i''}$ where $1 \leq i' < i'' \leq n$.*

Proof: If the shuffling is incorrect and for every $c_{j-1,\mu}$ for $\mu = 1, 2, \ldots, n$, there exists a $c_{j,\nu}$ with $1 \leq \nu \leq n$ satisfying $SV_j(c_{j-1,\mu}, c_{j,\nu}) = 1$, then there must be two inputs $c_{j-1,\mu 1}$ and $c_{j-1,\mu 2}$ satisfying $SV_j(c_{j-1,\mu 1}, c_{j,\tau}) = 1$ and $SV_j(c_{j-1,\mu 2}, c_{j,\tau}) = 1$ with $1 \leq \tau \leq n$. Otherwise there exists a permutation PM between the inputs and outputs such that $c_{j,\nu} = PM(c_{j-1,\mu})$ if $SV_j(c_{j-1,\mu}, c_{j,\nu})$, which is contradictory to the assumption that the shuffling is incorrect.

$SV_j(c_{j-1,\mu 1}, c_{j,\tau}) = 1$ and $SV_j(c_{j-1,\mu 2}, c_{j,\tau}) = 1$ means SV_j knows λ_1 and λ_2, so that $a_{j,\tau} = g^{\lambda_1} a_{j-1,\mu 1}$, $b_{j,\tau} = y^{\lambda_1} b_{j-1,\mu 1}$, $a_{j,\tau} = g^{\lambda_2} a_{j-1,\mu 2}$ and $b_{j,\tau} = y^{\lambda_2} b_{j-1,\mu 2}$. □

Proof of Theorem 1: As SV_j cannot get collusion of all the previous $j-1$ servers and at least two users, the inputs to SV_j are encrypted randomly from the viewpoint of SV_j and SV_j knows $\log_g a_{j-1,i}$ for at most one $c_{j-1,i} = (a_{j-1,i}, b_{j-1,i})$ where $1 \leq i \leq n$ if DL problem is intractable. So, if the shuffling by SV_j is incorrect, there exists $c_{j-1,\mu}$, so that $SV_j(c_{j-1,\mu}, c_{j,\nu}) \neq 1$ for $\nu = 1, 2, \ldots, n$. Otherwise according to Lemma 1 SV_j knows $\log_g a_{j-1,i'} - \log_g a_{j-1,i''}$ where $1 \leq i' < i'' \leq n$, which is contradictory to the above assumption. So

$$\log_g (a_{j,1}/a_{j-1,\mu}) = \log_y (b_{j,1}/b_{j-1,\mu}) \ \vee \ \log_g (a_{j,2}/a_{j-1,\mu}) = \\ \log_y (b_{j,2}/b_{j-1,\mu}) \ \vee \ldots \vee \ \log_g (a_{j,n}/a_{j-1,\mu}) = \log_y (b_{j,n}/b_{j-1,i})$$

can be proved in CV with a probability no more than $1/q$ as proof of equality of logarithms in CV implies knowledge of logarithm (without knowledge of the logarithm, SV_j can only guess the challenge and the success probability of the guess is $1/q$).

Therefore, CV can be satisfied with a probability no more than $1/q$. □

Even when SV_j colludes with all previous $j-1$ servers and at least two users, invalid shuffling of the honest users' inputs will still be discovered in CV with an overwhelmingly large probability. This conclusion is straightforward from the proof of Lemma 1. In proof of Lemma 1, it is illustrated that the only possible attack against correctness is for a malicious server to collude with two or more malicious users and all the previous servers to tamper any of these malicious users' inputs. Since an honest user will not conspire with the malicious server and will conceal the randomising factor in his encrypted input, the attack against the integrity of his input can only succeed with a negligible probability if DL is intractable. Due to space limitations, this conclusion is not proved in detail.

3.2 Batch Verification of Equality of Logarithms

A theorem for batch verification is presented in this section, which extends known batch techniques [3, 5, 11]. This technique can batch verify equality of logarithms and optimize efficiency of the verification protocol in Section 3.1. Batch verification of equality of logarithms was first mentioned in a voting scheme [18]. However, in [18], batch verification is not formally proposed or proved to be secure.

The formal description of batch verification of equality of logarithms is provided in Theorem 2, which will be formally proved.

Definition 2 $|\ |$ *is the absolute-value function from Z_p^* to G defined by*

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

Theorem 2. *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 $1 \le v \le n$ 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 $1 - 2^{-l}$.*

To prove Theorem 2, a lemma is proved first.

Lemma 2. *Suppose $y_i \in Z_p^*$ and $z_i \in Z_p^*$ for $i = 1, 2, \ldots, n$ and t_1, t_2, $\ldots$, t_{v-1}, t_{v+1}, t_{v+2}, $\ldots$, t_n are constant. If $\log_g |y_v| \neq \log_h |z_v|$ with $1 \le v \le n$ and $\log_g \prod_{i=1}^n y_i^{t_i} = log_h \prod_{i=1}^n z_i^{t_i}$, then there is only one possible solution for t_v.*

Proof: If the lemma is not correct, the following two equations are satisfied simultaneously where $\log_g |y_v| \neq \log_h |z_v|$, t_1, t_2, $\ldots$ and $t_v \neq \hat{t}_v$.

$$log_g \prod_{i=1}^{n} y_i^{t_i} = \log_h \prod_{i=1}^{n} z_i^{t_i} \tag{3}$$

$$\log_g \prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{\hat{t}_v} \prod_{i=v+1}^{n} y_i^{t_i} = \log_h \prod_{i=1}^{v-1} z_i^{t_i} \cdot z_v^{\hat{t}_v} \prod_{i=v+1}^{n} z_i^{t_i} \tag{4}$$

Without losing generality, suppose $t_v > \hat{t}_v$. (3) $-$ (4):

$$\log_g y_v^{t_v - \hat{t}_v} = \log_h z_v^{t_v - \hat{t}_v}$$

As y_v and z_v are members of Z_p^*, there are two possibilities.

1. y_v and z_v are members of G. Then $(t_v - \hat{t}_v) \log_g y_v = (t_v - \hat{t}_v) \log_h z_v \bmod q$. Note that $t_v - \hat{t}_v \neq 0 \bmod q$ because $1 \le \hat{t}_v < t_v < 2^l < q$. Therefore, $\log_g y_v = \log_h z_v \bmod q$
2. y_v or $z_v \in Z_p^* \backslash G$. Then $t_v - \hat{t}_v$ must have a factor 2 and $\frac{t_v - \hat{t}_v}{2} \log_g y_v^2 = \frac{t_v - \hat{t}_v}{2} \log_h z_v^2 \bmod q$. Note that $\frac{t_v - \hat{t}_v}{2} \neq 0 \bmod q$ because $1 \le \hat{t}_v < t_v < 2^l < q$. Therefore, $\log_g y_v^2 = \log_h z_v^2 \bmod q$. Namely $\log_g |y_v| = \log_h |z_v| \bmod q$.

In both cases, $\log_g |y_v| = \log_h |z_v|$. That is contradictory to the assumption $\log_g |y_v| \neq \log_h |z_v|$. $\square$

Proof of Theorem 2: Lemma 2 means that among the $(2^l)^n$ possible combinations of t_i for $i = 1, 2, \ldots, n$, at most $(2^l)^{n-1}$ of them can satisfy $\log_g \prod_{i=1}^n y_i^{t_i} = log_h \prod_{i=1}^n z_i^{t_i}$ when $\log_g |y_v| \neq \log_h |z_v|$. So if $\log_g |y_v| \neq \log_h |z_v|$ and t_i for $i = 1, 2, \ldots, n$ are randomly chosen, $\log_g \prod_{i=1}^n y_i^{t_i} = log_h \prod_{i=1}^n z_i^{t_i}$ is satisfied with probability no more than 2^{-l}. $\square$

4 The Proposed Mix Network

When the server SV_j performs ElGamal re-encryption and permutation π_j and Equation (2) is employed to verify the correctness of shuffling, the following properties are achieved.

1. A dishonest server SV_j can prove its incorrect shuffling to be correct with probability no more than $1/q$ without collusion of all the previous $j-1$ servers and at least two users. Even when SV_j colludes with all the previous $j-1$ servers and at least two users, invalid shuffling of honest users' inputs will still be discovered in CV with an overwhelmingly large probability.
2. Identified incorrect shuffling can be removed and the mix network can recover efficiently.
3. Computational costs for the prover and verifier of the correctness verification of a server's shuffling are $n(4n-2)$ and $4n^2$ exponentiations respectively.
4. If at least one server is honest, all the $n!$ permutation are equally possible in the mix network and if the number of malicious decrypting authorities is no more than t, privacy is achieved.

This mix network is denoted as Mix-1. However there are still some drawbacks of this solution:

- when two users conspire with the first server, correctness is not guaranteed;
- when n is large, $O(n^2)$ exponentiations is still a high cost.

To solve these problems, an idea of Abe[1, 2] is used: divide a $PN(n)$ into a few smaller shufflings, verification of whose correctness is efficient. However, switching gate $PN(2)$ is not applied in this paper to avoid complex construction of gate circuit. Instead, a simpler grouping technique is used.

4.1 Group Shuffling

On each server the n inputs are divided into groups with same size k, while re-encryption and random permutation are applied to each group. For simplicity, suppose $n = k^u$. There are $z = k^{u-1}$ groups. Usually $m \leq u$ as the number of servers is often small. The grouping function on every server is specially designed according to a general rule: if an input to the mix network is likely to be permuted to a few outputs after the shuffling of the first j servers, any two of these outputs (inputs to the $j+1^{th}$ server) cannot be divided into a same group on the $j+1^{th}$ server. This rule can provide the greatest diffusion, and thus as strong privacy as possible.

Before the shuffling, each server SV_j randomly generates $v_{j,i} \in G$ for $i = 1, 2, \ldots, n$. Inputs to the mix network c_i for $i = 1, 2, \ldots, n$ are sorted to $c_{0,i} = (a_{0,i}, b_{0,i})$ for $i = 1, 2, \ldots, n$, so that $a_{0,i} + \sum_{j=1}^{m} v_{j,i} \bmod p$ increases as i increases. On server SV_j, the shuffling is as follows.

1. Grouping
   - SV_j get inputs $c_{j-1,i}$ for $i = 1, 2, \ldots, n$ from SV_{j-1}. So far $c_{0,k^{j-1}w+1}$, $c_{0,k^{j-1}w+2}, \ldots c_{0,k^{j-1}w+k^{j-1}}$ have been shuffled to $c_{j-1,k^{j-1}w+1}, c_{j-1,k^{j-1}w+2}, \ldots c_{j-1,k^{j-1}w+k^{j-1}}$ for $w = 0, 1, \ldots, k^{u-j+1} - 1$. Denote $c_{j-1,k^{j-1}w+1}$, $c_{j-1,k^{j-1}w+2}, \ldots c_{j-1,k^{j-1}w+k^{j-1}}$ as a shuffling range $R_{j-1,w+1}$, then SV_j in fact receives k^{u-j+1} shuffling ranges $R_{j-1,1}$, $R_{j-1,2}$, $\ldots$, $R_{j-1,k^{u-j+1}}$.
   - SV_j regroups in every k successive shuffling ranges. The k inputs in the same position in every k successive shuffling ranges are regrouped into the same group. Namely, input $c_{j-1,i}$ is mapped to $c_{j,\alpha,\beta}$, which is the β^{th} element in Group α, where $\alpha = ((i-1)/k^j)k^{j-1} + ((i-1) \bmod k^{j-1}) + 1$ and $\beta = ((i-1) \bmod k^j)/k^{j-1} + 1$.
2. Re-encryption and permutation
   $c_{j,\alpha,\beta} = (a_{j,\alpha,\beta}, b_{j,\alpha,\beta})$ is permuted to $c'_{j,\alpha,\pi_{j,\alpha}(\beta)} = (a'_{j,\alpha,\pi_{j,\alpha}(\beta)}, b'_{j,\alpha,\pi_{j,\alpha}(\beta)}) = (g^{r_{j,\alpha,\beta}} a_{j,\alpha,\beta}$,
   $y^{r_{j,\alpha,\beta}} b_{j,\alpha,\beta})$ for $\alpha = 1, 2, \ldots, z$ and $\beta = 1, 2, \ldots, k$ where $r_{j,\alpha,\beta}$ is randomly chosen and $\pi_{j,\alpha}$ for $\alpha = 1, 2, \ldots, z$ are random secret permutations from $\{1, 2, \ldots, k\}$ to $\{1, 2, \ldots, k\}$.
3. De-grouping
   $c_{j,i} = c'_{j,\alpha,\beta}$ where $i = k(\alpha - 1) + \beta$.

Shuffling of SV_j is verified by SV_{j+1} before it starts its own shuffling using the following equation.

$$\begin{aligned} &\log_g (a'_{j,\alpha,1}/a_{j,\alpha,\beta}) = \log_y (b'_{j,\alpha,1}/b_{j,\alpha,\beta}) \ \vee \ \log_g (a'_{j,\alpha,2}/a_{j,\alpha,\beta}) = \\ &\log_y (b'_{j,\alpha,2}/b_{j,\alpha,\beta}) \vee \ldots \ \vee \ \log_g (a'_{j,\alpha,k}/a_{j,\alpha,\beta}) = \log_y (b'_{j,\alpha,k}/b_{j,\alpha,\beta}) \\ &\qquad \text{for } \alpha = 1, 2, \ldots, z \text{ and } \beta = 1, 2, \ldots, k \end{aligned} \tag{5}$$

Realization of verification of Equation (5) is denoted as *GCV (grouped correctness verification)*. If the verification fails, SV_{j+1} gets the outputs of SV_{j-1}, verifies them and uses them as its inputs if they are valid. If SV_{j-1}'s outputs are invalid too, he gets the outputs of the previous server until he finds a set of valid outputs as its inputs. After the shuffling of the last server, the outputs are decrypted as in Mix-1. This mix network applying group shuffling is denoted as Mix-2.

The following theorem can be proved in a way similar to the proof of theorem 1.

Theorem 3. *If the group shuffling by SV_j is incorrect, GCV can be satisfied with a probability no more than $1/q$ without collusion of all the previous $j-1$ servers and at least two users in a same group on SV_j, assuming DL problem is intractable.*

When conspiracy of all the previous servers and at least two malicious users is available, attack against correctness is more difficult than in Mix-1. As the grouping function is dependent on $v_{j,i}$ for $j = 1, 2, \dots m$ and $i = 1, 2, \dots n$, if at least one server is honest to generate them randomly, the grouping on any server is random. So if only static attack (all colluding users and servers are chosen before the attack starts) is considered and at least one server SV_j is honest to choose $v_{j,i}$ for $i = 1, 2, \dots n$ randomly, the probability that the colluding users are in the same group on any server is low. For example, even if SV_1 colludes with two users, they happen to fall in a same group with a probability $1/z$. That means although attacks involving more than one user and the first few servers against correctness is still possible, they succeed with a low probability[1]. Like in Mix-1, the probability to tamper with an honest user's input successfully is negligible if DL is intractable. Therefore, correctness property is improved.

The computational cost to produce the proof is $n(4k-2)$ exponentiations. The computational cost to verify the proof is $4nk$ exponentiations. Better efficiency is achieved compared to Mix-1.

Privacy of Mix-2 is achieved if the number of malicious decrypting authorities is no more than t. The extent of privacy is measured by two factors: diffusion of any single input and diffusion of the inputs as a whole. As stated before, in normal applications $m < u$. So, if a dishonest server reveals its shuffling, it makes no difference to the situation where this server performs re-encryption without permutation. Therefore, the only impact of this attack on the privacy of the shuffling of the whole mix network is to degrade the mix network to a mix network containing one fewer servers. The shuffling of the other servers is not affected and can still provide strong privacy protection.

- Diffusion of any single input: each input may be permuted to any of a set of k^ϵ outputs with an equal probability, where ϵ is the number of honest servers.
- Diffusion of the inputs as a whole: $(k!)^{z\epsilon}$ possible permutations from the inputs of the mix network to its outputs are equally likely.

If $m \geq u$, greater privacy is possible.

- When $\epsilon = u$, diffusion of single input may be as great as that in Mix-1 (any input to n equally likely outputs).
- When $\epsilon > u$, diffusion of the inputs as a whole may be as great as that in Mix-1 (all $n!$ possible permutations are equally likely).

However, it is only possible as it depends on the distribution of the honest servers.

[1] As k is usually small, z is large when n is large. So the probability is very low when n is very large as in a large-scale voting.

4.2 Batched Group-shuffling Mix Network

Efficiency of correctness verification of Mix-2 is better compared to that of Mix-1. However it is still costly when n is large. The batch verification technique in Section 3.2 can be employed to improve the efficiency further. If every server SV_j uses a same permutation π_j to replace $\pi_{j,\alpha}$ for $\alpha = 1, 2, \ldots, z$, according to Theorem 2 verification equation (5) can be batched as follows.

$$\begin{aligned}
& \log_g(\prod_{\alpha=1}^{z} {a'}_{j,\alpha,1}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} a_{j,\alpha,\beta}^{t_{j,\alpha}}) = \log_y(\prod_{\alpha=1}^{z} {b'}_{j,\alpha,1}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} b_{j,\alpha,\beta}^{t_{j,\alpha}}) \\
\vee \ & \log_g(\prod_{\alpha=1}^{z} {a'}_{j,\alpha,2}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} a_{j,\alpha,\beta}^{t_{j,\alpha}}) = \log_y(\prod_{\alpha=1}^{z} {b'}_{j,\alpha,2}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} b_{j,\alpha,\beta}^{t_{j,\alpha}}) \qquad (6) \\
\vee \ \ldots \ \vee \ & \log_g(\prod_{\alpha=1}^{z} {a'}_{j,\alpha,k}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} a_{j,\alpha,\beta}^{t_{j,\alpha}}) = \log_y(\prod_{\alpha=1}^{z} {b'}_{j,\alpha,k}^{t_{j,\alpha}} / \prod_{\alpha=1}^{z} b_{j,\alpha,\beta}^{t_{j,\alpha}}) \\
& \text{for } \beta = 1, 2, \ldots, k
\end{aligned}$$

where $t_{j,\alpha}$ for $\alpha = 1, 2, \ldots, z$ are random integers with length l. The verification in Equation 6 for any β is denoted as $BGCV_{j,\beta}$. If $BGCV_{j,\beta}$ holds for $\beta = 1, 2, \ldots, k$, it is denoted as $BGCV(j-1 \rightarrow j)$, which means the correction verification for SV_j is passed. $BGCV(j-1 \rightarrow j)$ is checked for $j = 1, 2, \ldots, m$ to ensure the correctness of the mix network.

This mix network is denoted as Mix-3.

Definition 3 *In Mix-3, group shuffling by SV_j is correct if for any $1 \leq \alpha \leq z$, the same permutation exists between $|D(c_{j,\alpha,\beta})|$ for $\beta = 1, 2, \ldots, k$ and $|D(c'_{j,\alpha,\beta})|$ for $\beta = 1, 2, \ldots, k$ where $D()$ denotes decryption.*

To apply equation (6), the construction of the mix network must be changed slightly as follows. After the shuffling of all the servers, the outputs of the mix network are decrypted. Every decrypted message M_i for $i = 1, 2, \ldots, n$ is checked to be in G by testing whether $M_i^q = 1$. If $M_i^q \neq 1$, an additional computation is performed: $M_i = -M_i = g_0^q M_i$.

5 Analysis

5.1 Correctness Analysis

Correctness of Mix-3 is proved in this subsection.

Definition 4 *Inputs of SV_j are divided into k vectors $V_\beta = (c_{j,1,\beta}, c_{j,2,\beta}, \ldots, c_{j,z,\beta})$ for $\beta = 1, 2, \ldots, k$ where $c_{j,\alpha,\beta} = (a_{j,\alpha,\beta}, b_{j,\alpha,\beta})$ is in $(Z_p^*)^2$. Outputs of SV_j are divided into k vectors $V'_\beta = (c'_{j,1,\beta}, c'_{j,2,\beta}, \ldots, c'_{j,z,\beta})$ for $\beta = 1, 2, \ldots, k$ where $c'_{j,\alpha,\beta} = (a'_{j,\alpha,\beta}, b'_{j,\alpha,\beta})$ is in $(Z_p^*)^2$.*

Definition 5 *$SV_j(V_\mu, V'_\nu) = 1$ means SV_j knows r_α satisfying $|a'_{j,\alpha,\nu}| = g^{r_\alpha}|a_{j,\alpha,\mu}|$ and $|b'_{j,\alpha,\nu}| = y^{r_\alpha}|b_{j,\alpha,\mu}|$ for $\alpha = 1, 2, \ldots, z$.*

Lemma 3. *If the shuffling by SV_j in Mix-3 is incorrect and for every V_μ with $1 \leq \mu \leq k$ there exists some V'_ν with $1 \leq \nu \leq k$ satisfying $SV_j(V_\mu, V'_\nu) = 1$, then SV_j knows $\log_g a_{j-1,\alpha,i'} - \log_g a_{j-1,\alpha,i''}$ for $\alpha = 1, 2, \ldots, z$ where $1 \leq i' < i'' \leq k$.*

Proof of Lemma 3 is very similar to that of Lemma 1, so is left to the reader.

Lemma 4. *$y_i \in Z_p^*$ for $i = 1, 2, \ldots, n$. $1 \leq t_i < 2^l < p$ for $i = 1, 2, \ldots, n$ where t_i are random values and l is a security parameter. If there exists v, such that $1 \leq v \leq n$ and the logarithm $\log_g |y_v|$ is not known, then $\log_g \prod_{i=1}^n y_i^{t_i}$ is known only with a probability no more than 2^{-l}.*

Proof: First we prove a statement: if there exists v, such that $1 \leq v \leq n$ and $\log_g |y_v|$ is not known, given a definite set $S = \{t_i \mid t_i < 2^l$ and $i = 1, 2, \ldots v - 1, v+1, \ldots n\}$, then $\log_g \prod_{i=1}^n y_i^{t_i}$ is known for at most one t_v.

If this statement is not correct, $\log_g (\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{t_v} \cdot \prod_{i=v+1}^n y_i^{t_i})$ and $\log_g (\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{\hat{t}_v} \cdot \prod_{i=v+1}^n y_i^{t_i})$ are known where $\log_g |y_v|$ is not known and $t_v \neq \hat{t}_v$.

So $\log_g (\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{t_v} \cdot \prod_{i=v+1}^n y_i^{t_i}) - \log_g (\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{\hat{t}_v} \cdot \prod_{i=v+1}^n y_i^{t_i})$
$= \log_g \frac{\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{t_v} \cdot \prod_{i=v+1}^n y_i^{t_i}}{\prod_{i=1}^{v-1} y_i^{t_i} \cdot y_v^{\hat{t}_v} \cdot \prod_{i=v+1}^n y_i^{t_i}} = \log_g y_v^{t_v - \hat{t}_v} = (t_v - \hat{t}_v) \log_g |y_v|$ is known.

Since $t_v - \hat{t}_v$ is public, $\log_g |y_v|$ is known. A contradiction is found, which means the statement is correct. So for every definite set $\{t_i \mid t_i < 2^l,\ i = 1, 2, \ldots, n\}$, the probability that t_v happens to be the unique possible value, so that $\log_g \prod_{i=1}^n y_i^{t_i}$ is known, is no more than 2^{-l} as there are 2^l choices for t_v. $\square$

Theorem 4. *If the shuffling by SV_j is incorrect according to Definition 3, $BGCV(j - 1 \rightarrow j)$ holds with a probability no more than $1 - (q-1)(1-2^{-l})/q$ without collusion of all the previous $j-1$ servers and at least $2z$ users with their re-encrypted inputs as $c_{j,\alpha,\rho}$ and $c_{j,\alpha,\delta}$ for $\alpha = 1, 2, \ldots, z$ where $1 \leq \rho < \delta \leq k$ if DL problem is intractable.*

Proof: The following denotations are used.
C denotes the shuffling is correct.
E_μ denotes $BGCV(j, \mu)$ holds.
Q denotes $BGCV(j - 1 \rightarrow j)$ holds.
$N1_\mu$ denotes $D(c_{j,\alpha,\mu}) \neq D(c'_{j,\alpha,\nu})$ for $1 \leq \nu \leq k$.
$N2_\mu$ denotes $D(c_{j,\alpha,\mu}) = D(c'_{j,\alpha,\nu})$, but SV_j does not know $\log_g |a'_{j,\alpha,\nu}/a_{j,\alpha,\mu}|$ for $1 \leq \nu \leq k$.

As supposed, SV_j cannot get collusion of all the previous $j-1$ servers and at least $2z$ users with their re-encrypted inputs as $c_{j,\alpha,\rho}$ and $c_{j,\alpha,\delta}$ for $\alpha = 1, 2, \ldots, z$ where $1 \leq \rho < \delta \leq k$. So for any $\log_g a_{j,\alpha,\rho}$ and $\log_g a_{j,\alpha,\delta}$ for $\alpha = 1, 2, \ldots, z$ where $1 \leq \rho < \delta \leq k$, SV_j knows at most $2z - 1$ of them and the left one is independent of these $2z - 1$ values in the viewpoint of SV_j if DL problem is intractable. According to Lemma 3, if the shuffling by server SV_j is incorrect and DL problem is intractable, there exists a vector V_μ and no V'_ν with $1 \leq \nu \leq k$ can satisfy $SV_j(V_\mu, V'_\nu) = 1$, where vector $V_\mu = (c_{j,1,\mu}, c_{j,2,\mu}, \ldots, c_{j-1,z,\mu})$ and

$V'_\nu = (c'_{j,1,\nu}, c'_{j,2,\nu}, \ldots, c'_{j,z,\nu})$. Otherwise, SV_j knows $\log_g a_{j-1,\alpha,i'} - \log_g a_{j-1,\alpha,i''}$ for $\alpha = 1, 2, \ldots, z$ where $1 \le i' < i'' \le k$, which is contradictory to the fact that for any $\log_g a_{j,\alpha,\rho}$ and $\log_g a_{j,\alpha,\delta}$ for $\alpha = 1, 2, \ldots, z$ where $1 \le \rho < \delta \le k$ SV_j knows at most $2z-1$ of them and the left one is independent of those $2z-1$ values in the viewpoint of SV_j.

$SV_j(V_\mu, V'_\nu) \neq 1$ means there exists α, such that $1 \le \alpha \le k$ and $N1_\mu \vee N2_\mu$ is true. Namely, $P(N1_\mu/\bar{C}) + P(N2_\mu/\bar{C}) = 1$.

According to Theorem 2, $P(\bar{E}_\mu, N1_\mu) \ge 1 - 2^{-l} - (1-2^{-l})/q = (q-1)(1-2^{-l})/q$.

According to Lemma 4, when $N2_\mu$ happens, $\log_g \prod_{\alpha=1}^{z} (a_{j,\alpha}/a_{j-1,\alpha,\mu})^{t_\alpha}$ is known to SV_j with a probability no more than 2^{-l}. So $P(\bar{E}_\mu, N2_\mu) \ge 1 - 2^{-l} - (1-2^{-l})/q = (q-1)(1-2^{-l})/q$.

So, $P(\bar{E}_\mu/\bar{C}) = P(N1_\mu/\bar{C})P(\bar{E}_\mu/N1_\mu) + P(N2_\mu/\bar{C})P(\bar{E}_\mu/N2_\mu) = (q-1)(1-2^{-l})/q$.

Therefore, $P(\bar{Q}/\bar{C}) = P(\bar{E_1}/\bar{C}) \vee P(\bar{E_2}/\bar{C}) \ldots \vee P(\bar{E_k}/\bar{C}) \ge P(\bar{E}_\mu/\bar{C}) = (q-1)(1-2^{-l})/q$.
Namely $P(Q/\bar{C}) \le 1 - (q-1)(1-2^{-l})/q$. □

According to Theorem 4, Mix-3 can provide correctness on every server with an overwhelmingly large probability if DL problem is intractable and on a condition that this server cannot obtain the collusion of all the previous $j-1$ servers and users with at least 2 inputs in each group on two same positions. This condition is much weaker than the conditions for correctness in Mix-1 and Mix-2 as even though collusion of $2z$ or more users is available, the probability of their inputs are in each group on two same positions is very small if at least one server SV_j is honest to choose $v_{j,i}$ for $i = 1, 2, \ldots n$ randomly. Like in Mix-1 and Mix-2, the probability to tamper with an honest user's input is negligible. If the shuffling on every server is correct, the plaintexts in the inputs to the mix network $\{m_1, m_2, \ldots, m_n\}$ and its plaintext outputs $\{M_1, M_2, \ldots, M_n\}$ have a relationship $\{m_1, m_2, \ldots, m_n\} = \{|M_1|, |M_2|, \ldots, |M_n|\}$. If $M_i^q \neq 1$, an additional computation $M_i = M_i g_0^q$ is performed to obtain correct outputs. Therefore, stronger correctness is achieved in Mix-3 than in Mix-1 and Mix-2 as less trust on the users is needed in Mix-3.

5.2 Other Properties

Shuffling by every server can be verified publicly and efficiently and a cheating server can be identified immediately. Any identified cheating server is deleted and its inputs become inputs to the next server. So abnormal situations can be dealt with efficiently and the proposed scheme is robust.

Recall that as defined in Section 1 and Section 4 there are n users and m servers in the mix network; the number of honest servers is ϵ; t-out-of-m threshold distributed decryption is used; k is the size of a group, z is the number of groups and $k^u = n$.

The computational cost for correctness proof and verification on a server in Mix-3 are $k(4k-2)$ and $4k^2$ exponentiations respectively. These costs are independent of the number of inputs and more efficient than those in Mix-2.

	Extent of Correctness	Diffusion of a single input	Diffusion of all the inputs	Is the diffusion uniform?
Abe[1]	not specified	1 among n if $\epsilon > t$	$n!$ permutations if $\epsilon > t$	No
Abe[2]	not specified	1 among n if $\epsilon > t$	$n!$ permutations if $\epsilon > t$	Yes
Furukawa[8]	not specified	1 among n	$n!$ permutations	Yes
Neff[14]	not specified	1 among n	$n!$ permutations	Yes
Groth[10]	not specified	1 among n	$n!$ permutations	Yes
Mix-1	$P(P/\bar{C}) \leq 1/q$	1 among n	$n!$ permutations	Yes
Mix-2	$P(P/\bar{C}) \leq 1/q$	1 among k^ϵ	$(k!)^{z\epsilon}$ permutations	Yes
Mix-3	$P(P/\bar{C}) \leq 1-(q-1)(1-2^{-l})/q$	1 among k^ϵ	$(k!)^\epsilon$ permutations	Yes

Table 1. Comparison of the mix networks

Privacy of Mix-3 is achieved if the number of malicious decrypting authorities is no more than t. Extent of privacy in Mix-3 is as follows when $m < u$ is assumed.

– Diffusion of any single input in Mix-3 is the same as that in Mix-2 (each input may be permuted to any of k^ϵ outputs with an equal probability).
– Diffusion of the inputs as a whole in Mix-3 is weaker: $(k!)^\epsilon$ possible permutation from the inputs of the mix network to its outputs are equally likely.

So, stronger correctness and higher efficiency in Mix-3 compared to in Mix-2 is achieved by sacrificing some privacy. By selecting appropriate k and m, a good trade-off between efficiency and privacy can be achieved. When $m \geq u$, as in the case of Mix-2, privacy may be improved in both factors if the distribution of honest servers is appropriate.

In Table 1 and Table 2, the proposed scheme is compared against the best mix networks in the third category (defined in Section 2). Note the following points

– "not specified" in Table 1 means the probability of correctness (with how much a probability the mix network is correct)is not provided.
– In [1] only $t+1$ out of the m servers take part in the shuffling.
– Re-encryption on each server cost $4(n\log_2 n - n + 1)$ exponentiations in [1] if ElGamal encryption is employed, while in other shuffling schemes this cost is usually $2n$. That is another aspect of inefficiency in [1].
– In [14], it was declared that the total computational cost of proof and verification of shuffling correctness is $8k+5$. However, the shuffling scheme in [14] is not concrete and it is commonly believed that Neff's scheme is not

so efficient as he claimed. Like Groth's analysis in [10], in this paper it is concluded that Neff's shuffling scheme costs ςn exponentiations (where ς is a small integer) and is not as efficient as [8] or [10].

In [1] only $t+1$ out of the m servers take part in the shuffling. The final version of the proposed scheme, Mix-3, achieves correctness more clearly (with a concrete extent) than the other schemes. Suppose in the proposed scheme, the decrypting authorities are chosen from the shuffling servers and the decryption key is shared among them with a t-out-of-m threshold like in most other mix networks. Then, when $\epsilon \leq t$, there is no privacy in either Abe's schemes [1, 2] or the proposed scheme as the inputs can be decrypted by $t+1$ malicious servers. When $\epsilon > t$, privacy in Mix-3 is sufficient for most applications although dependent on ϵ it may not achieve the maximum privacy as in [1]. The proposed scheme is more efficient than all the other schemes, especially when n is large. Moreover, the proposed scheme is simpler than Abe's schemes as complex gate circuit is not employed and the achieved properties are not dependent on theorems in gate circuit theory.

	Correctness proof on a server	Correctness verification on a server
Abe[1]	$12(n \log_2 n - n + 1)$	$16(n \log_2 n - n + 1)$
Furukawa[8]	$8n$	$10n$
Neff[14]	$o(n)$	$o(n)$
Groth[10][a]	$6n + 3n/\kappa + 3$	$6n + 3n/\kappa + 6$
Mix-3	$k(4k-2)$	$4k^2$

[a] κ is a parameter smaller than n.

Table 2. Comparison of computation cost in full-length exponentiations

In Table 3, an example is given to make a clearer comparison where $|q| = 1024$, $n = 10000$, $m = 5$, $t = 2$, $k = 10$, $\epsilon = 4 > t$, $\kappa = 100$ and SV_5 is assumed to be dishonest. Note that computational cost in Table 3 is in full-length exponentiations while some multiplications and short-length exponentiations are ignored as they are trivial compared to the costs of the full-length exponentiations . The results of this table clearly demonstrate enormous improvement on efficiency in the proposed scheme without losing strong correctness and privacy when there are a large number of users. In a national wide election involving millions of voters, the efficiency advantage of the proposed scheme is greater.

6 Conclusion

The proposed mix network provides strong and precise correctness and privacy. With the help of a grouping function and a batch verification technique, the mix

	Extent of Correctness	Diffusion of a single input	Diffusion of all the inputs	Cost of proof on a server	Cost of server verification
Abe[1]	not specified	1 among 10000	10000! permutations	1474537	1966050
Furukawa[8]	not specified	1 among 10000	10000! permutations	80000	100000
Groth[10]	not specified	1 among 10000	10000! permutations	60303	60306
Mix-3	$P(P/\bar{C})$ is extremely small	1 among 10000	1.734×10^{26} permutations	380	400

Table 3. Example for comparison

network is very efficient. The mix network is robust and can deal with dishonest servers efficiently.

Acknowledgements

We acknowledge the support of the Australian government through ARC Discovery Grant No. DP0345458.

References

1. M Abe. Mix-networks on permutation net-works. In *Asiacrypt 98*, pages 258–273, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science 1716.
2. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography 2001*, pages 317–324, Berlin, 2001. Springer-Verlag. Lecture Notes in Computer Science 1992.
3. M Bellare, J A Garay, and T Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EuroCrypt'98*, pages 236–250, Berlin, 1998. Springer-Verlag. Lecture Notes in Computer Science 1403.
4. Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 68–77, 2002.
5. Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In *Advances in Cryptology -ASIACRYPT 2000*, pages 58–71, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science 1976.
6. D Chaum. Untraceable electronic mail, return address and digital pseudonym. In *Communications of the ACM, 24(2)*, pages 84–88, 1981.
7. R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - Crypto '94*, pages 174–187, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.
8. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2001.
9. Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *ASIACRYPT 2002*, pages 451–465, Berlin, 2002. Springer-Verlag. Lecture Notes in Computer Science Volume 1592.

10. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography 2003*, pages 145–160, Berlin, 2003. Springer-Verlag. Lecture Notes in Computer Science Volume 2567.
11. Fumitaka Hoshino, Masayuki Abe, and Tetsutaro Kobayashi. Lenient/Strict batch verification in several groups. In *Information Security, 4th International Conference, ISC 2001*, pages 81–94, Berlin, 2001. Springer-Verlag. Lecture Notes in Computer Science Volume 2200.
12. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomizsed partial checking. In *Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002*, pages 339–353. USENIX, 2002.
13. Ari Juels and Markus Jakobsson. An optimally robust hybrid mix network. In *Proc. of the 20th annual ACM Symposium on Principles of Distributed Computation*, pages 284–292. ACM, 2001.
14. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security 2001*, pages 116–125, 2001.
15. W Ogata, K Kurosawa, K Sako, and K Takatani. Fault tolerant anonymous channel. In *Proc. of International Conference on Information and Communication Security 1997*, pages 440–444, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1334.
16. Miyako Ohkubo and Masayuki Abe. A length-invariant hybrid mix. In *ASIACRYPT 2000*, pages 178–191, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1976.
17. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology - EuroCrypt '93*, pages 248–259, Berlin, 1993. Springer-Verlag. Lecture Notes in Computer Science Volume 765.
18. K. Sako and J. Kilian. Receipt-free mix-type voting scheme–a practical solution to the implementation of a voting booth. In *EUROCRYPT '95*, pages 393–403, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science Volume 921.
19. Kapali Viswanathan, Colin Boyd, and Ed Dawson. A three phased schema for sealed bid auction system design. In *Information Security and Privacy, 5th Australasian Conference, ACISP'2000*, pages 412–426, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science 1841.