Attack and Correction: How to Design a Secure and Efficient Mix Network

Kun Peng*

Abstract—Shuffling is an effective method to build a publicly verifiable mix network to implement verifiable anonymous channels that can be used for important cryptographic applications like electronic voting and electronic cash. One shuffling scheme by Groth is claimed to be secure and efficient. However, its soundness has not been formally proven. An attack against the soundness of this shuffling scheme is presented in this paper. Such an attack compromises the soundness of the mix network based on it. Two new shuffling protocols are designed on the basis of Groth's shuffling and batch verification techniques. The first new protocol is not completely sound, but is formally analyzed in regards to soundness, so it can be applied to build a mix network with formally proven soundness. The second new protocol is completely sound, so is more convenient to apply. Formal analysis in this paper guarantees that both new shuffling protocols can be employed to build mix networks with formally provable soundness. Both protocols prevent the attack against soundness in Groth's scheme. Both new shuffling protocols are very efficient as batch-verification-based efficiency-improving mechanisms have been adopted. The second protocol is even simpler and more elegant than the first one as it is based on a novel batch cryptographic technique.

Keywords—Mix Network, Correction

1. Introduction

A mix network [6] is a very important cryptographic tool. It is a normal method for implementing verifiable anonymous channels, which is essential for anonymous communication applications like anonymous email, anonymous browsing, electronic cash, and especially electronic voting. A mix network shuffles a set of ciphertexts, so that they cannot be traced back to their original sources. A mix network is usually composed of multiple shuffling nodes, each of which re-encrypts (or partially decrypts) and re-orders the ciphertexts so that they cannot be traced if at least one shuffling node conceals the permutation it uses in its shuffling. Thus, shuffling is an essential primitive for verifiable anonymous communication applications. As anonymous communication applications like e-voting must be publicly verifiable without compromising privacy, very often the employed mix network and thus shuffling must be publicly verifiable through zero knowledge proof techniques. Namely, each shuffling node must publicly prove correctness of its shuffling without revealing the permutation it uses. A few publicly verifiable shuffling schemes [1, 2, 8, 11-13, 16, 18, 20, 22] have been published in recent years. To

Manuscript received October 11, 2011; accepted December 27, 2011.
Corresponding Author: Kun Peng
* Institute for Inforcomm Research, Singapore (dr.kun.peng@gmail.com)

achieve verifiability conveniently and efficiently, these schemes adopt re-encryption (and not partial decryption) in their shuffling. From now on in this paper, shuffling refers to verifiable shuffling.

A shuffling scheme by Groth [13] is among the most efficient shuffling schemes without limit on permutation. However, the soundness of Groth's shuffling has not been formally proven, so it still remains dubious. It is illustrated in this paper that Groth's shuffling cannot guarantee that the messages encrypted in the shuffled ciphertexts are not tampered with. A malicious shuffling node can perform an incorrect shuffling (tampering with any encrypted message) and pass the verification with a probability at least as large as 0.5. In this paper, such an attack is presented in order to breach the soundness of Groth's shuffling and thus the soundness of the mix network that is based on it. It is then pointed out that the vulnerability of Groth's shuffling lies in the mis-use of batch verification technique. A formal batch verification theory is applied to analyse Groth's shuffling and the attack.

In this paper, Groth's shuffling [13] is modified to prevent the discovered attack. Firstly, the ElGamal encryption employed in Groth's shuffling is modified and the original parameter setting and shuffling operation employed in Groth's shuffling are adapted. Although the modified El-Gamal-based shuffling protocol is still not completely sound, it is formally analysed in regards to soundness so that 1) a sound and verifiable mix network can be built up on it; 2) unlike other mix network schemes employing shuffling without verifiable soundness (e.g. [10]), the new mix network can always output correctly shuffled result and never needs rewinding. Then, a new shuffling protocol employing Paillier encryption is designed as an optimisation, which is completely sound and thus can be directly employed to build a sound mix network. The soundness of both of the new mix network schemes are formally proven with the help of the formal batch verification theory. The new schemes make good use of batch verification, such that they not only prevent the presented attack, but they also avoid the informal assumption about a random oracle in [18]. The new Paillier-based shuffling is even more advanced than the new ElGamal-based shuffling as it is completely sound and based on a novel batch proof-and-verification technique.

Contributions to this paper include the discovery and analysis of the attack against the soundness of Groth's shuffling and the mix network that is based on it, the proposal of batch verification techniques with formally provable security, the application of formal batch verification theory to shuffling verification, and fixing Groth's shuffling to achieve provable soundness and better performance.

2. The Background Mix Network and Shuffling

An anonymous channel is an important tool that is frequently employed in various secure network and communication applications. A verifiable anonymous channel is usually implemented through a mix network, which was first proposed by Chaum [6]. A mix network receives a number of ciphertexts and outputs the same number of plaintexts, so that the output plaintexts are an unknown permutation of the messages encrypted in the input ciphertexts. A mix network is widely employed in applications like anonymous emails [6], anonymous browsing [9], electronic auctions [19], and especially e-voting [16]. A mix network is usually composed of multiple shuffling nodes, who in turn shuffle the ciphertexts. As explained in Section 1, this paper is only interested in verifiable shuffling, whose correctness can be verified. In verifiable shuffling, each node re-encrypts and permutes its input ciphertexts from the last node to the same number of output ciphertexts encrypting the same set of messages. Multiple instances of shuffling together with an appropriate decryption function can build up a mix network, such that the permutation in the mix network is not revealed if at least one shuffling node conceals its permutation. The following properties must be satisfied in a shuffling:

• Correctness: if the shuffling node follows the shuffling protocol, the plaintexts encrypted in the output ciphertexts are a permutation of the plaintexts encrypted in the input ciphertexts.
• Public verifiability: the shuffling node can publicly prove that it does not deviate from the shuffling protocol.
• Soundness: a successfully verified proof of a shuffling node guarantees that it follows the shuffling protocol.
• Privacy: the permutation used by the node is not revealed.

The four properties above can guarantee the following properties of a mix network using that shuffling.

• Correctness: if all the shuffling nodes follow the shuffling protocol, the output plaintexts are a permutation of the plaintexts encrypted in the input ciphertexts.
• Public verifiability: the shuffling nodes in the mix network can publicly prove that they do not deviate from the shuffling protocol.
• Soundness: the successfully verified proof of the nodes guarantees that all of the shuffling nodes follow the shuffling protocol.
• Privacy: the permutation between the inputs and the outputs in the mix network is not revealed.

In recent years, several shuffling schemes [1, 2, 8, 11-13, 16, 18, 20, 22] claiming to achieve the properties above have been proposed. Among them, [2] is a slight modification of [1], while [13] is a generalization and improvement of [16]. Various shuffling and verification techniques are used in these shuffling schemes. The shuffling in [1] and [2] employs multiple small shuffling gates to form a shuffling circuit. The shuffling in [8] bases the shuffling on a permutation matrix. The shuffling in [16] and [13] employs the proof of the equality of the product of exponents. The shuffling in [20] employs the batch verification theory to improve the efficiency of shuffling verification. The shuffling in [18] is inspired by [13], but employs different proof techniques. The idea in [13] and [18] is further explored in [22, 12]. The shuffling in [11] sacrifices computational efficiency to pursue higher communicational efficiency.

Although the shuffling in [20] is simple and very efficient, it has two drawbacks. First, the shuffling node is not completely free to choose a random permutation and only a fraction of all the possible permutations are permitted. Second, it assumes the nodes have no knowledge of the messages encrypted in the shuffled ciphertexts, which is called the ignorance assumption in [18]. These two drawbacks limit the application of the shuffling in [20]. [13] is one of the most efficient shuffling schemes without a limit on permutation. Although not explicitly mentioned, the idea of batch verification is also employed in [13]. However, unlike [20, 13] did not base batch verification on a formally provable theory. As Groth did not give a formal proof of soundness for his scheme, doubt still remains about its reliability. When the soundness of [18] is proven, a random oracle is employed to avoid an ignorant assumption. A hash function is used to implement the random oracle, which brings an informal factor into the proof.

3. GROTH'S SHUFFLING AND AN ATTACK

Groth's shuffling is recalled in this section and an attack breaking its soundness is described. It is illustrated that this attack can succeed with a probability of at least 0.5. The origin of the vulnerability and principle of the attack are also analysed.

3.1 Groth's Shuffling

In Groth's shuffling, input ciphertexts $c_1, c_2, \dots, c_n$ are shuffled to output ciphertexts $c'_1, c'_2, \dots, c'_n$, such that $D(c'_1), D(c'_2), \dots, D(c'_n)$ is a permutation of $D(c_1), D(c_2), \dots, D(c_n)$.

Groth [13] suggested verifying the correctness of the shuffling by testing:

$$\sum_{i=1}^{n} t_i D(c_i) = \sum_{i=1}^{n} t_{\pi(i)} D(c'_i) \bmod q \tag{1}$$

where $\pi()$ is a permutation of $\{1,2,\dots,n\}$ and t_i for $i = 1,2,\dots,n$ are randomly chosen. Unfortunately, Equation (1) cannot be proved or verified explicitly since $\pi()$ must be kept secret. Groth proposed a technique to publicly compute $c = \prod_{i=1}^{n} c_i^{t_i}$ and $c' = \prod_{i=1}^{n} c_i'^{t_{\pi(i)}}$ without revealing $\pi()$. Then $D(c'/c) = 1$ can be proved and verified. For simplicity, his prototype protocol instead of his final optimised protocol is described here. The latter only optimises some calculation details and both protocols share the same method and same vulnerability to attack. Groth suggested using a homomorphic encryption algorithm like the ElGamal encryption. The analysis in [13] is based on the following ElGamal encryption. Let $\hat{p}$ be a prime, $\hat{q}$ a factor of $\hat{p}-1$ and cyclic group $\hat{G}$ with the order $\hat{q}$ as a subgroup of $Z_{\hat{p}}^*$. Let $\hat{g}_0$ be a generator of $Z_{\hat{p}}^*$, and $\hat{g}_1$ and $\hat{g}_2$ be the generators of $\hat{G}$. The private key $\hat{X}$ is chosen from $Z_{\hat{q}}$ and the public key $(\hat{g}_1, \hat{Y} = \hat{g}_1^{\hat{X}})$ is published. The message m is encrypted into $(\hat{g}_1^r, m\hat{Y}^r)$ where r is randomly chosen from $Z_{\hat{q}}$. The re-encryption function $RE(c, r')$ re-encrypts the ciphertext $c = (a,b)$ to $c' = (a', b') = (\hat{g}_1^{r'} a, \hat{Y}^{r'} b)$. A ciphertext $c = (a,b)$ is decrypted into $b/a^{\hat{X}}$. $ZP(x_1, x_2, \dots, x_k \mid y_1, y_2, \dots, y_l)$ denotes the ZK proof of knowledge of integers $x_1, x_2, \dots, x_k$ satisfying conditions $y_1, y_2, \dots, y_l$. Groth's prototype protocol (including verification of its validity) is as follows where the detailed implementation of the ZK proofs is omitted:

1. The shuffling node publishes $s_i \in Z_{\hat{q}}$ and $S_i = \hat{g}_1^{s_{\pi(i)}} \hat{g}_2^{r_i}$ for $i = 1,2,\dots,n$ where $\pi()$ is a permutation of $\{1,2,\dots,n\}$ and r_i is randomly chosen from $Z_{\hat{q}}$. It outputs a statis-

tical ZK proof that s_i for $i = 1,2,\ldots,n$ are permuted and committed in S_i for $i = 1,2,\ldots,n$ where the permutation $\pi()$ is kept secret.

2. The shuffling node gets $c_i = (a_i, b_i)$ for $i = 1,2,\ldots,n$ and shuffles them to $c'_i = RE(c_{\pi(i)}, w_i) = (a'_i, b'_i)$ for $i = 1,2,\ldots,n$ where w_i is randomly chosen from $Z_{\hat{q}}$.

3. Random integers $t_i \in Z_{\hat{q}}$ for $i = 1,2,\ldots,n$ are chosen by the verifier.

4. The shuffling node publishes $T_i = \hat{g}_1^{t_{\pi(i)}} \hat{g}_2^{r'_i}$ for $i = 1,2,\ldots,n$ where r'_i is randomly chosen from $Z_{\hat{q}}$. It also provides a ZK proof that the same permutation used to shuffle and commit s_i for $i = 1,2,\ldots,n$ in S_i for $i = 1,2,\ldots,n$ is used to shuffle and commit t_i for $i = 1,2,\ldots,n$ in T_i for $i = 1,2,\ldots,n$ while the permutation $\pi()$ is kept secret.

5. The shuffling node calculates $a' = \hat{g}_1^{\gamma} \prod_{i=1}^{n} a_i'^{t_{\pi(i)}}$ and $b' = \hat{Y}^{\gamma} \prod_{i=1}^{n} b_i'^{t_{\pi(i)}}$ where γ is randomly chosen from $Z_{\hat{q}}$ and provides a ZK proof $ZP\,(\gamma, t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(n)}, r'_1, r'_2, \ldots,$

$$r'_n \mid a' = \hat{g}_1^{\gamma} \prod_{i=1}^{n} a_i'^{t_{\pi(i)}}, b' = \hat{Y}^{\gamma} \prod_{i=1}^{n} b_i'^{t_{\pi(i)}} \quad ,$$

$$T_i = \hat{g}_1^{t_{\pi(i)}} \hat{g}_2^{r'_i} \; for\ i = 1,2,\ldots,n\,), \tag{2}$$

which can be implemented using a ZK proof of knowledge for logarithm [21] and a ZK proof of equality for logarithms [7].

6. The shuffling node proves

$$\log_{\hat{g}_1}(a'/\prod_{i=1}^{n} a_i^{t_i}) = \log_{\hat{Y}}(b'/\prod_{i=1}^{n} b_i^{t_i}) \tag{3}$$

using the proof of equality of logarithms [7]. The verifier checks the proof without knowledge of $\pi()$.

3.2 Attack against Groth's Shuffling

Groth did not provide his batch technique with a formal proof and analysis, so its soundness remains dubious. Therefore, it is not surprising that the following attack can break its soundness. Although Proof (??) and Equation (3) guarantee

$$\log_{\hat{g}_1}(\prod_{i=1}^{n} a_i'^{t_{\pi(i)}}/\prod_{i=1}^{n} a_i^{t_i}) = \log_{\hat{Y}}(\prod_{i=1}^{n} b_i'^{t_{\pi(i)}}/\prod_{i=1}^{n} b_i^{t_i}) \tag{4}$$

they are not enough to guarantee the correctness of the shuffling. If the shuffling node shuffles $c_{\pi(i)}$ to $c'_i = (a'_i, \hat{g}_0^{(\hat{p}-1)/2} b'_i)$ where $(a'_i, b'_i) = RE(c_{\pi(i)}, w_i)$ for some i in $\{1,2,\ldots,n\}$ while shuffling all the other inputs honestly, then when $t_{\pi(i)}$ is, even this shuffling can pass Groth's verification. If the verifier always chooses an odd t_i for $i = 1,2,\ldots,n$, a dishonest server can still pass the verification by shuffling two inputs $c_{\pi(i)}$ and $c_{\pi(j)}$ to $c'_i = (a'_i, \hat{g}_0^{(\hat{p}-1)/2} b'_i)$ and $c'_j = (a'_j, \hat{g}_0^{(\hat{p}-1)/2} b'_j)$ and shuffling all the other inputs honestly. This attack succeeds with a probability of no less than 0.5. When $(\hat{p}-1)/\hat{q}$ has factors other than 2, the probability of a successful attack is even greater.

This attack shows that Groth's shuffling is not sound and cannot guarantee that the plaintexts encrypted in the output ciphertexts are a permutation of the plaintexts encrypted in the input ciphertexts. As a result, a mix network based on Groth's shuffling is not sound and cannot guarantee that the output plaintexts are a permutation of the plaintexts encrypted in the input ciphertexts.

3.3 Origin or the Vulnerability and Principle of the Attack

The verification of Equation (4) is actually a batch verification. Batch verification is a cryptographic technique that was first analyzed by Bellare et al. [4] and then developed further [5, 15, 3, 20]. Batch verification is a technique to verify a batch of similar claims with a single verification, so that the cost is lower than when the claims are verified separately. A shuffling scheme [20] successfully improves the efficiency of shuffling verification while maintaining an overwhelmingly large probability. Although batch verification is not mentioned in [13], it is implicitly employed. However, Groth did not use batch verification to improve the efficiency of separate proofs and verifications, but employed batch verification to hide the permutation, which will be revealed in separate proofs and verifications. Equation (4) is equivalent to:

$$\log_{\hat{g}_1} \prod_{i=1}^{n} (a'_{\pi^{-1}(i)}/a_i)^{t_i} = \log_{\hat{Y}} \prod_{i=1}^{n} (b'_{\pi^{-1}(i)}/b_i)^{t_i} \tag{5}$$

So from the viewpoint of batch shuffling verification (see [20] for details), Groth actually based his shuffling verification on Hypothesis 1.

Hypothesis 1 Suppose $y_i \in Z_{\hat{p}}$, $z_i \in Z_{\hat{p}}$ and $t_i \in Z_{\hat{q}}$ for $i = 1,2,\ldots,n$. If $\log_{\hat{g}_1} \prod_{i=1}^{n} y_i^{t_i} = \log_{\hat{Y}} \prod_{i=1}^{n} z_i^{t_i}$, then $\log_{\hat{g}_1} y_i = \log_{\hat{Y}} z_i$ for $i = 1,2,\ldots,n$ except for a negligible probability.

Unfortunately, when $\log_{\hat{g}_1} y_j = \log_{\hat{Y}} \hat{g}_0^{(\hat{p}-1)/2} z_j$ and $i \le j \le n$, the verification equation $\log_{\hat{g}_1} \prod_{i=1}^{n} y_i^{t_i} = \log_{\hat{Y}} \prod_{i=1}^{n} z_i^{t_i}$ can still be satisfied when t_j is even. No matter how the t_i for $i = 1,2,\ldots,n$ are chosen, the verification equation can still be satisfied when there is a $\log_{\hat{g}_1} y_j = \log_{\hat{Y}} \hat{g}_0^{(\hat{p}-1)/2} z_j$ with a probability of at least 0.5. When $(\hat{p}-1)/\hat{q}$ has factors other than 2, the probability is even greater. So Hypothesis 1 is incorrect and the satisfaction of $\log_{\hat{g}_1} \prod_{i=1}^{n} y_i^{t_i} = \log_{\hat{Y}} \prod_{i=1}^{n} z_i^{t_i}$ is not enough for the batch verification of $\log_{\hat{g}_1} y_i = \log_{\hat{Y}} z_i$ for

$i = 1,2,\ldots,n$. That is why the attack in Section 3.2 is feasible.

The essence of the attack in Section 3.2 is that a ciphertext encrypting a message in $\hat{G}$ can be incorrectly shuffled to a ciphertext encrypting a message in $Z_p^* - \hat{G}$ without being detected with a non-negligible probability. Obviously, if a_i, b_i, a'_i and b'_i for $i = 1,2,\ldots,n$ are verified to be in $\hat{G}$, the attack can be prevented. Although Groth assumed that a_i and b_i are in $\hat{G}$ and the shuffling node acts honestly, neither assumption is verified in his scheme. The membership test of ciphertexts is neither mentioned in this protocol nor counted in the cost estimation of his scheme. Even if the membership test is performed to avoid the attack, the high efficiency of the shuffling scheme is compromised, as additional $4n$ exponentiations are needed. Moreover, when the membership test detects an attack, Groth did not propose a method to fix the incorrect output. If the shuffling has to be rewound, both the efficiency and robustness of the shuffling are compromised.

4. Protocol 1: ElGamal Based Shuffling and a Mix Network

Our idea is that, although complete soundness is difficult to implement in Groth's shuffling, the parameter setting and shuffling operation in Groth's shuffling can be re-designed, such that the modified shuffling can be employed to build a sound mix network. The new shuffling does not employ a membership test and is very efficient. Multiple instances of the new shuffling are used to build a new mix network, which employs a final membership test on the output plaintexts to guarantee the soundness of the mix network. Any incorrect shuffling can be detected by the final membership test. The detected mistake can be easily fixed without rewinding any shuffling. So although soundness is not guaranteed in any shuffling, it is achieved in the mix network. As the membership test is performed once in the whole mix network instead of once per shuffling node, the additional computational cost for a membership test is trivial. The parameter setting in Groth's shuffling is adapted in the new shuffling, so that the batch verification technique with formally provable security can be applied to the shuffling verification. The following ElGamal setting is used: G_1 is a cyclic subgroup of Z_p^* with order q where $p-1=2q$ and p, q are large primes. Let g_1, g_2 be generators of G_1 and g_0 be a generator of Z_p^*. The private key X is chosen from Z_q and the public key $(g_1, Y = g_1^X)$ is published. The message space is G_1. The message m is encrypted into (g_1^r, mY^r) where r is randomly chosen from Z_q. A ciphertext $c=(a,b)$ is decrypted into $\log_{g_1} b/a^X$. The absolute-value function from Z_p^* to G_1 defined by

$$|\sigma| = \begin{cases} \sigma & \text{if } \sigma \in G_1 \\ -\sigma \bmod p & \text{if } \sigma \in Z_p^* \setminus G_1 \text{ where } -1 = g_0^q \end{cases}$$

Note that the setting $p-1=2q$ guarantees that $p-1$ has no other factor than 2 and q. This change makes it easier to analyse and handle the attack in Section 3.2. Theorem 4 is based on this new setting.

Theorem 1 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 $2^L < q$. Let t_i for $i = 1,2,\dots,n$ be random integers smaller than 2^L. If there exists integer v, such that $1 \le v \le n$ and $\log_{g_1} | y_v | \ne \log_{g_2} | z_v |$, then $\log_{g_1} | \prod_{i=1}^{n} y_i^{t_i} | \ne \log_{g_2} | \prod_{i=1}^{n} z_i^{t_i} |$ with a probability no less than $1 - 2^{-L}$.

Theorem 1 has been proven in [20]. It supports the following shuffling protocol where, unless specified, all multiplications are with modulus p.

1. The shuffling node publishes $s_i \in Z_q$ and $S_i = g_1^{s_{\pi(i)}} g_2^{r_i}$ for $i = 1,2,\dots,n$ where $\pi()$ is a permutation of $\{1,2,\dots,n\}$ and r_i is randomly chosen from Z_q. The shuffling node also provides a ZK proof where s_i for $i = 1,2,\dots,n$ are shuffled and committed in S_i for $i = 1,2,\dots,n$ while the permutation $\pi()$ is kept secret. Details of the ZK proof technique were described in [13], which is statistical zero knowledge.

2. The shuffling node obtains $c_i = (a_i, b_i)$ for $i = 1,2,\dots,n$ and shuffles c_i to $c'_i = (a'_i, b'_i) = (\pm a_{\pi(i)} g_1^{w_i}, \pm b_{\pi(i)} y^{w_i})$ where w_i is randomly chosen from Z_q. This is a loose shuffling as the shuffling node has two choices for both a'_i and b'_i.

3. Random integers $t_i \in \{0,1,\dots,2^L - 1\}$ for $i = 1,2,\dots,n$ are chosen by the verifier (e.g., the next shuffling node or the other shuffling nodes together).

4. The shuffling node publishes $T_i = g_1^{t_{\pi(i)}} g_2^{r'_i}$ for $i = 1,2,\dots,n$ where r'_i is randomly chosen from Z_q. The shuffling node also provides a ZK proof where the same permutation used to shuffle and commit s_i for $i = 1,2,\dots,n$ in S_i for $i = 1,2,\dots,n$ is used to shuffle and commit t_i for $i = 1,2,\dots,n$ in T_i for $i = 1,2,\dots,n$, which can be implemented using a ZK proof of knowledge for logarithm [21] and a ZK proof of the equality of the logarithms [7]. The permutation $\pi()$ is not revealed in the proof.

5. The shuffling node calculates $a' = g_1^{\gamma} \prod_{i=1}^{n} a_i'^{t_{\pi(i)}}$ and $b' = Y^{\gamma} \prod_{i=1}^{n} b_i'^{t_{\pi(i)}}$ where γ is randomly chosen from Z_q and provides a ZK proof

$$ZP\,(\gamma, t_{\pi(1)}, t_{\pi(2)}, \dots, t_{\pi(n)}, r'_1, r'_2, \dots, r'_n \mid a' = g_1^{\gamma} \prod_{i=1}^{n} a_i'^{t_{\pi(i)}}, b' = Y^{\gamma} \prod_{i=1}^{n} b_i'^{t_{\pi(i)}},$$
$$T_i = g_1^{t_{\pi(i)}} g_2^{r'_i} \ for\ i = 1,2,\dots,n\,) \quad (6)$$

which can be implemented using a ZK proof of knowledge for logarithm [21] and a ZK proof of the equality of the logarithms [7].

6. The last step of the batch verification of correctness of the shuffling is a ZK proof of equality of logarithms [7]:

$$\log_{g_1} | a'/\prod_{i=1}^{n} a_i^{t_i} | = \log_Y | b'/\prod_{i=1}^{n} b_i^{t_i} | \tag{7}$$

Proof (??) and Equation (7) guarantee

$$\log_{g_1} | \prod_{i=1}^{n} a_i'^{t_{\pi(i)}} / \prod_{i=1}^{n} a_i^{t_i} | = \log_Y | \prod_{i=1}^{n} b_i'^{t_{\pi(i)}} / \prod_{i=1}^{n} b_i^{t_i} | \tag{8}$$

According to Theorem 4, satisfaction of Equation (8) guarantees

$$\log_{g_1} | a_i'/a_{\pi(i)} | = \log_Y | b_i'/b_{\pi(i)} | \; for \; i = 1,2,\ldots,n$$

while $\pi()$ is not revealed.

This new shuffling is publicly verifiable. As the verifier-honest ZK proof and statistical ZK proof are employed in the shuffling verification, this new shuffling is private. Unfortunately, although the batch shuffling verification in Equation (7) together with Proof (6) guarantees that the shuffling node does not deviate from the new loose shuffling protocol, the shuffling protocol is not strictly correct or sound as it is loose and does not satisfy the definitions of correctness or soundness of shuffling. Especially, the shuffling verification may be passed when $(a_i', b_i') = (-a_{\pi(i)} g_1^{w_i}, -b_{\pi(i)} y^{w_i})$. However, the following correct, sound, private, and publicly verifiable mix network can be built up on the base of this new shuffling:

1. Ciphertext $c_1, c_2, \ldots, c_n$ are submitted to the mix network.

2. Each shuffling node shuffles the ciphertexts in turn using the shuffling above.

3. The last shuffling outputs ciphertexts $c_1', c_2', \ldots, c_n'$.

4. The ciphertexts $c_1', c_2', \ldots, c_n'$ are decrypted into $m_1, m_2, \ldots, m_n$.

5. Each m_i is verified to be in G_1 for $i = 1,2,\ldots,n$. If any m_i is not in G_1, it is corrected into $-m_i$.

6. After the final correction, $m_1, m_2, \ldots, m_n$ are output.

As the new shuffling is private and publicly verifiable, this new mix network is also private and publicly verifiable. Moreover, this new mix network is correct and sound. Given (a_i, b_i) for $i = 1,2,\ldots,n$, each shuffling in the new mix network outputs (a_i', b_i') for $i = 1,2,\ldots,n$ such that $\log_{g_1} | a_i'/a_{\pi(i)} | = \log_Y | b_i'/b_{\pi(i)} | \; for \; i = 1,2,\ldots,n$ where $\pi()$ is a permutation. So, the mix network outputs $m_1, m_2, \ldots, m_n$ such that $m_i = \pm D(c_{\phi(i)})$ for $i = 1,2,\ldots,n$ where $\phi()$ is the permutation, which combines the permutations of all the shuffling nodes. Therefore, the final membership test and correction can guarantee $m_i = D(c_{\phi(i)})$ for $i = 1,2,\ldots,n$.

Namely the new mix network is correct and if no shuffling nodes deviate from the shuffling protocol, it is then sound. The guarantee of the shuffling nodes' honesty is illustrated in Theorem 2.

Theorem 2 The probability that a dishonest shuffling node does not strictly follow the shuffling protocol in Section 4, but passes the verification of Equation (7) is negligible. Proof: Let A_1 be the event where a shuffling node strictly follows the shuffling protocol; A_2 be the event where Equation (7) is correct; A_3 be the event where the shuffling passes the verification of Equation (7); and $P(A)$ denote the probability of event A.

$$\begin{aligned}
P(A_3/\overline{A_1}) &= P((A_3 \wedge A_2)/\overline{A_1}) + P((A_3 \wedge \overline{A_2})/\overline{A_1}) \\
&= P(A_3 \wedge A_2 \wedge \overline{A_1})/P(\overline{A_1}) + P(A_3 \wedge \overline{A_2} \wedge \overline{A_1})/P(\overline{A_1}) \\
&= P(\overline{A_1} \wedge A_2)P(A_3/\overline{A_1} \wedge A_2)/P(\overline{A_1}) + P(A_3 \wedge \overline{A_2} \wedge \overline{A_1})P(\overline{A_2} \wedge \overline{A_1})/(P(\overline{A_1})P(\overline{A_2} \wedge \overline{A_1})) \\
&= P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(\overline{A_2}/\overline{A_1})P(A_3 \wedge \overline{A_2} \wedge \overline{A_1})/P(\overline{A_2} \wedge \overline{A_1}) \\
&= P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(\overline{A_2}/\overline{A_1})P(A_3 \wedge \overline{A_2} \wedge \overline{A_1})/(P(\overline{A_2})P(\overline{A_1}/\overline{A_2}))
\end{aligned}$$

$P(\overline{A_1}/\overline{A_2}) = 1$ as $P(A_2/A_1) = 1$. So

$$\begin{aligned}
P(A_3/\overline{A_1}) &= P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(\overline{A_2}/\overline{A_1})P(A_3 \wedge \overline{A_2} \wedge \overline{A_1})/P(\overline{A_2}) \\
&\le P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(\overline{A_2}/\overline{A_1})P(A_3 \wedge \overline{A_2})/P(\overline{A_2}) \\
&\le P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(\overline{A_2}/\overline{A_1})P(A_3/\overline{A_2}) \\
&\le P(A_2/\overline{A_1})P(A_3/\overline{A_1} \wedge A_2) + P(A_3/\overline{A_2})
\end{aligned}$$

According to Theorem 4, $P(A_2/\overline{A_1}) \le 2^{-L}$. As Equation (7) is proven using a standard Chaum-Pedersen proof of equality of logarithms, $P(A_3/\overline{A_1} \wedge A_2) = 1$ and $P(A_3/\overline{A_2}) < 2^{-L'}$ where L' is the bit length of the challenge in the Chaum-Pedersen proof of equality of logarithms (e.g. 128). So

$$P(A_3/\overline{A_1}) \le P(A_2/\overline{A_1}) + P(A_3/\overline{A_2}) = 2^{-L} + 2^{-L'}$$

Theorem 2 guarantees that the shuffling nodes do not deviate from the shuffling instruction and $\log_{g_1} | a_i'/a_{\pi(i)} | = \log_Y | b_i'/b_{\pi(i)} | \; for \; i = 1,2,\ldots,n$ is satisfied in every shuffling in the mix network with an overwhelmingly large probability. The guarantee is strong enough with a short t_i (e.g. 30 or 40 bits long). So, in the mix network $m_i = \pm D(c_{\pi(i)})$ for $i = 1,2,\ldots,n$ with some permutation of $\pi()$ with an overwhelmingly large probability. Therefore, the mix network is sound. Theorem 1 and Theorem 2 illustrate that the attack against Groth's shuffling can be handled and a correct and sound mix network can be obtained. Although an additional final check and correction is needed in the new mix network, it is still more efficient than a mix network based on Groth's shuffling due to the usage of a short t_i and a better trade-off between soundness and efficiency. Note that our proof and verification technique is different from the aggregate shuffling verification technique proposed in [10], which fails in instant verification and the mistakes found by the final verification cannot be fixed without rewinding the mix network. Passing each shuffling verification in the new mix network guarantees a correct final result and any mistake found by the final check can be corrected without rewinding any shuffling.

5. Protocol 2: Paillier Based Shuffling and the Mix Network

In this section, the shuffling scheme in Protocol 1 is extended to the Paillier encryption. Unlike the shuffling in Protocol 1, this new shuffling is correct and sound. When the Paillier encryption algorithm is employed, batch-shuffling verification is strictly based on Theorem 5. Before Theorem 5 is presented, a parameter setting of the Paillier encryption [17] is introduced. $N = p_1 q_1$, $p_1 = 2p' + 1$, $q_1 = 2q' + 1$ where p_1, q_1, p' and q' are large primes and $GCD(N, p'q') = 1$. G_2 is the cyclic subgroup containing all the quadratic residues in $Z^*_{N^2}$. Unless specified, all multiplicative computations in this section take place in $Z^*_{N^2}$ with modulus N^2. L is a security parameter such that $2^L < min(p', q')$. $\Pr[A \mid \mu_1, \mu_2, \ldots, \mu_n \in S \mid \nu_1, \nu_2, \ldots, \nu_m \mid F(\mu_1, \mu_2, \ldots, \mu_n, \nu_1, \nu_2, \ldots, \nu_m)]$ denotes the probability distributing over S with variables $\mu_1, \mu_2, \ldots, \mu_n$ that given input $\mu_1, \mu_2, \ldots, \mu_n, \nu_1, \nu_2, \ldots, \nu_m$ polynomial-time algorithm A calculates $F(\mu_1, \mu_2, \ldots, \mu_n, \nu_1, \nu_2, \ldots, \nu_m)$.

Theorem 3 If $\Pr[A \mid t_1, t_2, \ldots, t_n \in_R \{0,1,\ldots,2^L - 1\}^n \mid y_1, y_2, \ldots, y_n \mid (\prod_{i=1}^n y_i^{t_i})^{1/N}] > 2^{-L}$, then by querying A, a polynomial-time algorithm B with inputs $y_1, y_2, \ldots, y_n$ can be built to calculate $y_i^{1/N}$ for any integer i in $\{1,2,\ldots,n\}$.

Proof: $\Pr[A \mid t_1, t_2, \ldots, t_n \in_R \{0,1,\ldots,2^L - 1\}^n \mid y_1, y_2, \ldots, y_n \mid (\prod_{i=1}^n y_i^{t_i})^{1/N}] > 2^{-L}$ implies that for any given integer v in $\{1,2,\ldots,n\}$ there must exist integers $t_1, t_2, \ldots, t_n$ and t'_v in $\{0,1,\ldots,2^L - 1\}$ such that

$$A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_n) \to (\prod_{i=1}^n y_i^{t_i})^{1/N} \; A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_{v-1}, t'_v, t_{v+1}, \ldots, t_n)$$
$$\to ((\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{t'_v} \prod_{i=v+1}^{n} y_i^{t_i})^{1/N}$$

Otherwise, for any valid $(t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_n)$, there is at most one t_v in $\{1,2,\ldots,n\}$ to satisfy $A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_n) \to (\prod_{i=1}^n y_i^{t_i})^{1/N}$. This implies that among the 2^{nL} possible choices for $\{t_1, t_2, \ldots, t_n\}$ (a combination of $2^{(n-1)L}$ possible choices for $\{t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_n\}$ and 2^L possible choices for t_v) there are at most $2^{(n-1)L}$ choices to satisfy $A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_n) \to (\prod_{i=1}^n y_i^{t_i})^{1/N}$, which is a contradiction to the assumption that $\Pr[A \mid t_1, t_2, \ldots, t_n \in_R \{0,1,\ldots,2^L - 1\}^n \mid y_1, y_2, \ldots, y_n \mid (\prod_{i=1}^n y_i^{t_i})^{1/N}] > 2^{-L}$. So algorithm B can be designed as follows:

1. Perform a brute-force query to A with all possible input $t_1, t_2, \ldots, t_n$ until meeting the two correct instances of computation:

$$A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_n) \rightarrow (\prod_{i=1}^{n} y_i^{t_i})^{1/N} \tag{9}$$

$$A(y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_{v-1}, t'_v, t_{v+1}, \ldots, t_n) \rightarrow ((\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{t'_v} \prod_{i=v+1}^{n} y_i^{t_i})^{1/N} \tag{10}$$

2. The calculation result of (9) divided by the calculation result of (10) yields $(y_v^{t_v - \hat{t}_v})^{1/N}$.

3. Let $(y_v^{t_v - \hat{t}_v})^{1/N} = \omega$. Then $\omega^N = y_v^{t_v - \hat{t}_v}$. According to the Euclidean algorithm, there exists integers α and β, such that $\beta(t_v - \hat{t}_v) = \alpha N + GCD(N, t_v - \hat{t}_v)$. So calculate α and β using the Euclidean algorithm and output $\omega^\beta / y_v^\alpha$.

Note the following two facts:

- n is constant and L is a security parameter with a small constant value (e.g., 30), so the brute-force query for ω has a complexity of $O(2^{nL})$ and is polynomial time. Although $O(2^{nL})$ involves the exponentiation of parameters and may be a high cost, especially when n is large, it is independent of the length of N, which indicates the difficulty of finding the N^{th} root modulo N^2. So we can say that $O(2^{nL})$ is a polynomial cost compared to the hard problem to solve in Theorem 3.
- The prover can calculate α and β satisfying $\beta(t_v - \hat{t}_v) = \alpha N + GCD(N, t_v - \hat{t}_v)$ in polynomial time from N and $t_v - \hat{t}$ using Euclidean algorithm. $GCD(N, t_v - \hat{t}_v) = 1$ as $t_v - \hat{t}_v < 2^L < min(p_1, q_1)$. So $y_v^{\beta(t_v - \hat{t}_v)} = y_v^{\alpha N} y_v$. Namely, $y_v = y_v^{\beta(t_v - \hat{t}_v)} / y_v^{\alpha N} =$ $(y_v^{(t_v - \hat{t}_v)})^\beta / y_v^{\alpha N} = \omega^{N\beta} / (y_v^\alpha)^N = (\omega^\beta / y_v^\alpha)^N$. So $y_v^{1/N} = \omega^\beta / y_v^\alpha$.

Therefore, B is a polynomial-time algorithm to calculate $y_v^{1/N}$. As v can be any integer in $\{1, 2, \ldots, n\}$, algorithm B can be built to calculate $y_i^{1/N}$ for any i in $\{1, 2, \ldots, n\}$.

According to Theorem 3, the verification of the knowledge of $y_i^{1/N}$ for $i = 1, 2, \ldots, n$ can be batched to the verification of the knowledge of $(\prod_{i=1}^{n} y_i^{t_i})^{1/N}$. If the prover does not know $y_i^{1/N}$ for any $i \in \{1, 2, \ldots, n\}$, then the prover can compute $(\prod_{i=1}^{n} y_i^{t_i})^{1/N}$ with only negligible probability.

With the support of Theorem 3, the batch ZK proof-verification technique can be employed to design a Paillier-based shuffling. Suppose the Paillier ciphertexts c_i for $i = 1, 2, \ldots, n$ are shuffled to c'_i for $i = 1, 2, \ldots, n$. The shuffling is as follows:

1. The shuffling node publishes $s_i \in Z_N$ and $S_i = g_3^{s_{\pi(i)}} r_i^N$ for $i = 1, 2, \ldots, n$ where $\pi()$ is a permutation of $\{1, 2, \ldots, n\}$ and r_i is randomly chosen from Z_N^*. The shuffling node also provides a proof that s_i for $i = 1, 2, \ldots, n$ are shuffled and committed in S_i

while the permutation $\pi()$ is kept secret. Details of the ZK proof technique were described in [13], which is statistical zero knowledge.

2. The shuffling node obtains c_i for $i = 1,2,\ldots,n$ and shuffles them to $c'_i = c_{\pi(i)} w_i^N$ where w_i is randomly chosen from Z_N^*.

3. Random integers $t_i \in \{0,1,\ldots,2^L - 1\}$ for $i = 1,2,\ldots,n$ are chosen by the verifier (e.g., the next shuffling node or the other shuffling nodes together).

4. The shuffling node publishes $T_i = g_3^{t_{\pi(i)}} r_i'^N$ for $i = 1,2,\ldots,n$ where r'_i is randomly chosen from Z_N^*. The shuffling node also provides a proof that the same permutation used to shuffle and commit s_i for $i = 1,2,\ldots,n$ in S_i for $i = 1,2,\ldots,n$ is used to shuffle and commit t_i for $i = 1,2,\ldots,n$ in T_i for $i = 1,2,\ldots,n$, which can be implemented using the ZK proof of knowledge of root [14] and the ZK proof of equality of logarithms [7]. The permutation $\pi()$ is not revealed in the proof.

5. The shuffling node calculates $c' = \gamma^N \prod_{i=1}^{n} c_i'^{t_{\pi(i)}}$ where γ is randomly chosen from Z_N^* and provides a ZK proof

$$ZP\,(\,\gamma, t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(n)}, r'_1, r'_2, \ldots, r'_n \mid c' = \gamma^N \prod_{i=1}^{n} c_i'^{t_{\pi(i)}} T_i = g_3^{t_{\pi(i)}} r_i'^N \;\; for\ i = 1,2,\ldots,n\,) \quad (11)$$

which can be implemented using the proof of knowledge of root [14] and the proof of the equality of logarithms in [7].

6. The last step of the batch verification of correctness of the shuffling is a ZK proof of knowledge of the root proposed in [14]:

$$(c'/\prod_{i=1}^{n} c_i^{t_i})^{1/N}. \quad (12)$$

Proof (??) and Proof (12) guarantee the knowledge of:

$$(\prod_{i=1}^{n} c_i'^{t_{\pi(i)}} / \prod_{i=1}^{n} c_i^{t_i})^{1/N}. \quad (13)$$

According to Theorem 5, the proof of knowledge of (13) implies the knowledge of $(c'_i / c_{\pi(i)})^{1/N}$ for $i = 1,2,\ldots,n$ while $\pi()$ is not revealed.

Note that unlike the ElGamal-based shuffling protocol, the Paillier-based shuffling protocol is correct. Namely, if the shuffling party does not deviate from the protocol, the plaintexts encrypted in the output ciphertexts is a permutation of the plaintexts encrypted in the input ciphertexts. The soundness of the Paillier-based shuffling protocol is illustrated in Theorem 4.

Theorem 4 The probability that a dishonest shuffling node does not follow Step 2 strictly in its shuffling, but passes the verification of Equation (7) is negligible.

Theorem 4 can be proved like Theorem 2. Due to space limit, the proof is not repeated. Theorem 4 guarantees that if the shuffling verification is passed the plaintexts encrypted in the output ciphertexts is a permutation of the plaintexts encrypted in the input ciphertexts with an overwhelmingly large probability. Namely, the new Paillier-based shuffling is sound. So a sound mix network can be built up using the new Paillier-based shuffling without any extra operation like a final membership test and providing the correction needed in the ElGamal-based shuffling protocol. Therefore, multiple layers of Paillier-based shuffling and a final decryption form a correct, sound, private, and publicly verifiable mix network. Like the new ElGamal-based shuffling, the new Paillier-based shuffling protocol is more efficient than Groth's shuffling as a shorter t_i is used to obtain a better trade-off between soundness and efficiency.

6. Comparison

In Table 1, the properties of the existing shuffling schemes are compared. Privacy and public verifiability are not included in the table as all the schemes in the table are private and publicly verifiable. It is assumed that the optimised computation of modulo exponentiation and multiplication in [13] is employed to improve efficiency in the new shuffling protocols, which are optimisations of [13] in soundness and efficiency. As the efficiency optimisation mechanism in [13] does not affect soundness, it can be employed without causing any concern. As shorter exponents are used in the new shuffling protocols, they are more efficient than [13]. In the comparison, it is assumed that $L = 20$ and $L' = 128$. Assume that an exponentiation with an x-

Table 1. Comparison of properties

	Correctness	Soundness	Permutation	Drawbacks	Computation (full length exponentiation)
[1, 2]	Yes	Yes	Unlimited	Low efficiency	$\geq 16(n\log_2 n - 2n + 2)$
[8]	Yes	Yes	Unlimited		$10\ n$
[16]	Yes	Yes	Unlimited		$12\ n$
[13][1]	Yes	No	Unlimited	Vulnerable to attacks	$8n + 3n/\kappa + 3$
[20][1]	Yes	Yes	Limited	Assumption of ignorance	$2n + k(4k - 2)$
[18]	Yes	Yes	Unlimited	Informal random oracle	$6n + 3.5$
[22][1]	Yes	Yes	Unlimited		$\geq 7n$
[12]	Yes	Yes	Unlimited		$4n$
[11]	Yes	Yes	Unlimited		$7n$
New shuffling[1] ElGamal-based	Support correct mix	Support sound mix	Unlimited		average $4n + 1.5n/\kappa + n/m$
New shuffling Paillier-based	Yes	Yes	unlimited		$3n + n/\kappa$

bit exponent costs $1.5x$ multiplications and a product of y exponentiations with x-bit exponents costs $y+0.5yx$ multiplications as according to [4]. According to this assumption, short exponentiations are counted in terms of full length exponentiation. The number of full-length exponentiations in all the computations (including re-encryption and proof of validity) is counted in Table 1, which illustrates that the new shuffling protocols satisfy all the requirements and are very efficient.

7. Conclusion

Groth's shuffling [13] is not sound and is vulnerable to attacks. Incorrect shuffling can pass the verification in Groth's shuffling with a probability of no less than 0.5. The attack exists in Groth's shuffling because the batch verification technique is improperly used. Two new shuffling protocols without any limitation to permutation are proposed, based on ElGamal encryption and Paillier encryption respectively. Both shuffling protocols are based on batch verification techniques with formally provable security and are formally analysed in regard to soundness. They can be employed to build correct, sound, private, efficient, and publicly verifiable mix networks. As formal batch verification techniques allow for a better trade-off between soundness and efficiency, these two new shuffling protocols are very efficient. They prevent an attack against Groth's shuffling, which is detected in this paper. The new Paillier-based shuffling is even more advanced than the new ElGamal-based shuffling as it is completely sound and based on a novel batch proof-and-verification technique.

References

[1] M Abe. *"Mix-networks on permutation net-works,"* In *ASIACRYPT* '98, pp.258-273.

[2] M Abe and F Hoshino. "*Remarks on mix-network based on permutation networks,"* In *PKC* '01, pp.317-324.

[3] R Aditya, K Peng, C Boyd, and E Dawson. "*Batch verification for equality of discrete logarithms and threshold decryptions,"* In *ACNS* '04, pp.494-508.

[4] M Bellare, J A Garay, and T Rabin. "*Fast batch verification for modular exponentiation and digital signatures*," In *EUROCRYPT* '98, pp.236-250.

[5] C Boyd and C Pavlovski. "*Attacking and repairing batch verification schemes*," In *ASIACRYPT* '00, pp.58-71.

[6] D Chaum. "*Untraceable electronic mail, return address and digital pseudonym*," Communications of the *ACM,* 24(2), 1981, pp.84-88.

[7] D Chaum and T Pedersen. "*Wallet databases with observers,"* In *CRYPTO* '92, pp.89-105.

[8] J Furukawa and K Sako. "*An efficient scheme for proving a shuffle*," In *CRYPTO* '01, pp.368-387.

[9] E Gabber, P Gibbons, Y Matias, and A Mayer. *"How to make personalized web browsing simple, secure, and anonymous,"* In *FC* '97, pp.17-31.

[10] P Golle, S Zhong, D Boneh, M Jakobsson, and A Juels. *"Optimistic mixing for exit-polls," In ASIACRYPT* '02, pp.451-465.

[11] J Groth and Y Ishai. *"Sub-linear zero-knowledge argument for correctness of a shuffle,"* In *EUROCRYPT* '08, pp.379-396.

[12] J Groth and S Lu. "*Verifiable shuffle of large size ciphertexts*," In *PKC* '07, pp.377-392.

[13] J Groth. "*A verifiable secret shuffle of homomorphic encryptions*," In *Public Key Cryptography* 2003, pp.145-160.

[14] L Guillou and J Quisquater. "*A ``paradoxical" identity-based signature scheme resulting from zero-*

knowledge," In Shafi Goldwasser, editor, *CRYPTO* '88, pp.216-231.
[15] F Hoshino, M Abe, and T Kobayashi. "*Lenient/Strict batch verification in several groups*," In *ISC* '01, pp.81-94.
[16] C Neff. "*A verifiable secret shuffle and its application to e-voting*," In *ACM CCS* '01, pp.116-125.
[17] P Paillier. "*Public key cryptosystem based on composite degree residuosity classes*,"In *EUROCRYPT* '99, pp.223-238.
[18] K Peng, C Boyd, and E Dawson. "*Simple and efficient shuffling with provable correctness and ZK privacy*," In *CRYPTO* '05, pp.188-204.
[19] K Peng, C Boyd, E Dawson, and K Viswanathan. "*Efficient implementation of relative bid privacy in sealed-bid auction*," In *WISA* '03, pp.244-256.
[20] K Peng, C Boyd, E Dawson, and K Viswanathan. "*A correct, private and efficient mix network,"* In *PKC* '04, pp.439-454.
[21] C Schnorr. "*Efficient signature generation by smart cards*," Journal of *Cryptology,* 4, 1991, pp.161-174.
[22] D Wikstrom. "*A sender verifiable mix-net and a new proof of a shuffle*," In A*SIACRYPT* '05, pp.273-292.

Kun Peng

Dr. Kun Peng received his Bachelor's degree in Software Engineering and his Master degree's in Computer Security from Huazhong University of Science and Technology in China. He obtained his PhD in information security from the Information Security Institute at the Queensland University of Technology in Australia in 2004. His main research interest is in applied public key cryptology. His main research interests include applied cryptology, network security, and secure e-commerce, and e-government. He is now a scientist at the Institute for Infocomm Research in Singapore.