Simple And Efficient Shuffling With Provable Correctness and ZK Privacy

Kun Peng, Colin Boyd and Ed Dawson

Information Security Institute
Queensland University of Technology
{k.peng, c.boyd, e.dawson}@qut.edu.au
http://www.isrc.qut.edu.au

Abstract. A simple and efficient shuffling scheme containing two protocols is proposed. Firstly, a prototype, Protocol-1 is designed, which is based on the assumption that the shuffling party cannot find a linear relation of the shuffled messages in polynomial time. As application of Protocol-1 is limited, it is then optimised to Protocol-2, which does not need the assumption. Both protocols are simpler and more efficient than any other shuffling scheme with unlimited permutation. Moreover, they achieve provable correctness and ZK privacy.

Keywords: shuffling, permutation, correctness, privacy, zero knowledge

1 Introduction

Shuffling is a very important cryptographic primitive. In a shuffling, a party re-encrypts and shuffles a number of input ciphertexts to the same number of output ciphertexts and publicly proves the validity of his operation. Its most important application is to build up anonymous channels used in e-voting [13], anonymous email [4] and anonymous browsing [7] etc. It is also employed in other cryptographic applications like multiparty computation [17] and electronic auction [18]. Two properties must be satisfied in a shuffling. The first property is correctness, which requires the shuffling party's validity proof to guarantee that the plaintexts of the outputs are a permutation of the plaintexts of the inputs. The second property is privacy, which requires the validity proof of the shuffling to be zero knowledge.

Recently, several shuffling schemes $[1, 2, 6, 13, 8, 19, 15]$ have been proposed. Among them, [2] is a slight modification of [1]; [15] is a Paillier-encryption-based version of [6]; a similar idea is used in [13] and [8]. Except [19], all of them employ complicated proof techniques to prove correctness of the shuffling. The shuffling in [1] and [2] employs a large and complex shuffling circuit; [6] and [15] explicitly deal with a $n \times n$ matrix (n is the number of inputs); [13] and [8] employ proof of equality of product of exponents. Complexity of the proof causes several drawbacks. Firstly, correctness of the shuffling is not always strict. More precisely, in [8], if an input is shuffled to its minus ($g^q = -1 \bmod 2q + 1$

where q and $2q+1$ are primes and the order of g modulo $2q+1$ is $2q$), the proof can be accepted with a probability no smaller than 0.5. Secondly, some details of the proof (for example, the efficiency optimisation mechanism in [8]) are too complex to be easily understood or strictly analysed. Thirdly, the proofs in [6], [13] and [15] are not honest-verifier zero knowledge as pointed out in [10], [15] and [14]. So their privacy cannot be strictly and formally guaranteed. Finally, the proof is inefficient in all of them except [19]. Especially, the computational cost in [1] and [2] are linear in $n \log n$ while [13] and [8] need seven rounds of communication.

Although [19] is simple and very efficient, it has two drawbacks. Firstly, only a fraction of all the possible permutations are permitted. Secondly, it needs an assumption called linear ignorance assumption in this paper.

Definition 1 *Let $D()$ be the decryption function for an encryption scheme with plaintext space $\{0, 1, \ldots, q-1\}$. Suppose an adversary $\mathcal{A}$ is given a set of n valid ciphertexts $c_1, c_2, \ldots, c_n$. $\mathcal{A}$ succeeds if it outputs integers $l_1, l_2, \ldots, l_n$, not all zero, such that $\sum_{i=1}^{n} l_i D(c_i) = 0 \bmod q$. The* linear ignorance assumption *states that there is no efficient adversary that can succeed with non-negligible probability.*

In [19], linear ignorance assumption is used against the shuffling party, who receives some ciphertext to shuffle and acts as the adversary. It is assumed in [19] that given the ciphertexts to shuffle, the probability that the shuffling party can efficiently find a linear relation about the messages encrypted in them is negligible. When the encryption scheme is semantically secure and the distribution of $D(c_1), D(c_2), \ldots, D(c_n)$ is unknown, this assumption is reasonable. However, if some party with some information about $D(c_1), D(c_2), \ldots, D(c_n)$ collude with the shuffling party, this assumption fails.

In this paper, two correct and private shuffling protocols, denoted as Protocol-1 and Protocol-2, are proposed. Protocol-1 is a prototype and needs the linear ignorance assumption against the shuffling party. So the shuffling party's knowledge of the shuffled messages is strictly limited in Protocol-1. Therefore, Protocol-1 is not suitable for applications like e-voting, where the shuffling party (tallier) may get some information about the shuffled messages from some message providers (colluding voters). Protocol-2 is an optimization of Protocol-1. It requires slightly more computation than Protocol-1, but concretely realises linear ignorance of the shuffling party in regard to the ciphertexts to shuffle. Namely, in Protocol-2, linear ignorance of the shuffling party in regard to the ciphertexts is not an assumption but a provable fact, which is an advantage over [19] and Protocol-1. As a result, Protocol-2 does not need the linear ignorance assumption, so is suitable for a much wider range of applications than Protocol-1. Both the new shuffling protocols are honest-verifier zero knowledge and more efficient than $[1, 2, 6, 13, 8, 15]$. Moreover, neither of them limits the permutation, which is an advantage over [19].

2 The Shuffling Protocol

Let n be the number of inputs. An additive homomorphic semantically-secure encryption scheme[1] like Paillier encryption [16] is employed where $E(m, r)$ stands for encryption of message m using random integer r, $RE(c, r)$ stands for re-encryption of ciphertext c using random integer r and $D(c)$ stands for decryption of ciphertext c. Additive homomorphism of the encryption scheme implies $RE(c, r) = cE(0, r)$. Let q be the modulus of the message space, which has no small factor. Any computation in any matrix or vector is modulo q in this paper. In encryption or re-encryption the random factor r is chosen from a set Q dependent on the encryption algorithm. $|m|$ stands for the bit length of an integer m. L is a security parameter, such that 2^L is no larger than the smallest factor of q.

M' stands for the transpose matrix of a matrix M. A matrix is called a permutation matrix if there is exactly one 1 in every row and exactly one 1 in every column in this matrix while the other elements in this matrix are zeros. $ZP\ (\ x_1, x_2, \ldots, x_k \mid f_1, f_2, \ldots, f_l\ )$ stands for a ZK proof of knowledge of secret integers $x_1, x_2, \ldots, x_k$ satisfying conditions $f_1, f_2, \ldots, f_l$. $ExpCost(x)$ stands for the computational cost of an exponentiation computation with a x bit exponent. In this paper, it is assumed that $ExpCost(x)$ equals $1.5x$ multiplications. $ExpCost^n(x)$ stands the computational cost of the product of n exponentiations with x-bit exponents. Bellare *et al.* [3] showed that $ExpCost^n(x)$ is no more than $n + 0.5nx$ multiplications.

In a shuffling, ciphertexts $c_1, c_2, \ldots, c_n$ encrypting messages $m_1, m_2, \ldots, m_n$ are sent to a shuffling party, who shuffles the ciphertexts into $c'_1, c'_2, \ldots, c'_n$ and has to prove that $D(c'_1), D(c'_2), \ldots, D(c'_n)$ is a permutation of $D(c_1), D(c_2), \ldots, D(c_n)$. Batch verification techniques in [17] indicate that if

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

can be satisfied with a non-negligible probability where $s_1, s_2, \ldots, s_n$ are randomly chosen and $\pi()$ is a permutation, the shuffling is correct and $D(c'_i) = D(c_{\pi(i)})$ for $i = 1, 2, \ldots, n$. However, direct verification of Equation (1) requires knowledge of $\pi()$. To protect privacy of the shuffling, $\pi()$ must not appear in the verification. Groth's shuffling scheme [8] shows that to prove Equation (1) without revealing $\pi()$ is complicated and inefficient. In the new shuffling scheme a much simpler method is employed. Firstly, it is proved that the shuffling party knows $t_1, t_2, \ldots, t_n$ such that

$$\sum_{i=1}^{n} s_i D(c_i) = \sum_{i=1}^{n} t_i D(c'_i) \bmod q \tag{2}$$

[1] An encryption algorithm with encryption function $E()$ is additive homomorphic if $E(m_1)E(m_2) = E(m_1 + m_2)$ for any messages m_1 and m_2. An encryption algorithm is semantically-secure if given a ciphertext c and two messages m_1 and m_2, such that $c = E(m_i)$ where $i = 1$ or 2, there is no polynomial algorithm to find out i.

where it is not required to prove that $t_1, t_2, \ldots, t_n$ are a permutation of $s_1, s_2, \ldots, s_n$. This proof does not reveal the permutation, but is not strong enough to guarantee validity of the shuffling. Actually, Equation (2) only implies that under the linear ignorance assumption against the shuffling party there exists a matrix M such that $(D(c'_1), D(c'_2), \ldots, D(c'_n)) \cdot M = (D(c_1), D(c_2), \ldots, D(c_n))$. As M need not be a permutation matrix, this proof only guarantees that $D(c_1), D(c_2), \ldots, D(c_n)$ is a linear combination of $D(c'_1), D(c'_2), \ldots, D(c'_n)$ under the linear ignorance assumption against the shuffling party. However, repeating this proof in a non-linear manner can guarantee M is a permutation matrix under the linear ignorance assumption against the shuffling party. In Protocol-1, given random integers s_i and s'_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$, the shuffling party has to prove that he knows secret integers t_i and t'_i from Z_q for $i = 1, 2, \ldots n$, such that

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

$$\sum_{i=1}^{n} s'_i D(c_i) = \sum_{i=1}^{n} t'_i D(c'_i) \bmod q$$

$$\sum_{i=1}^{n} s_i s'_i D(c_i) = \sum_{i=1}^{n} t_i t'_i D(c'_i) \bmod q$$

Note that $s_i s'_i$ and $t_i t'_i$ in the third equation breaks the linear relation among the three equations. Under the linear ignorance assumption against the shuffling party, the three equations above can guarantee correctness of the shuffling with an overwhelmingly large probability. In Protocol-2, every input to be shuffled is randomly distributed into two inputs, each in one of two input sets. Then the two sets of inputs are shuffled separately using the same permutation. As the distribution is random, the input messages in both shufflings are random and are unknown even to the original message providers. So it is impossible for the shuffling party to find any linear relation of the input messages in either shuffling as the employed encryption algorithm is semantically secure. As the two shufflings are identical, their outputs can be combined to be the final shuffled outputs.

2.1 Protocol-1

In Protocol-1, it is assumed that the shuffling party cannot find a linear relation of $m_1, m_2, \ldots, m_n$ in polynomial time. Protocol-1 is as follows.

1. The shuffling party randomly chooses $\pi()$, a permutation of $\{1, 2, \ldots, n\}$, and integers r_i from Q for $i = 1, 2, \ldots n$. He then outputs $c'_i = RE(c_{\pi(i)}, r_i)$ for $i = 1, 2, \ldots n$ while concealing $\pi()$.
2. A verifier randomly chooses and publishes s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$. The shuffling party chooses r'_i from Q for $i = 1, 2, \ldots n$ and

publishes $c''_i = c'^{t_i}_i E(0, r'_i)$ for $i = 1, 2, \ldots n$ where $t_i = s_{\pi(i)}$. The shuffling party publishes ZK proof

$$ZP\ (\ t_i, r'_i \mid c''_i = c'^{t_i}_i E(0, r'_i)\ ) \qquad \text{for } i = 1, 2, \ldots n \tag{3}$$

and

$$ZP\ (\ r_i, t_i, r'_i \text{ for } i = 1, 2, \ldots, n \mid \prod_{i=1}^{n} c_i^{s_i} \prod_{i=1}^{n} (E(0, r_i))^{t_i} E(0, r'_i) = \prod_{i=1}^{n} c''_i\ ) \tag{4}$$

3. The verifier randomly chooses and publishes s'_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$. The shuffling party sets $t'_i = s'_{\pi(i)}$ for $i = 1, 2, \ldots n$ and publishes ZK proof

$$ZP\ (\ r_i, t_i, r'_i, t'_i \text{ for } i = 1, 2, \ldots n \mid \prod_{i=1}^{n} c_i^{s'_i} \prod_{i=1}^{n} (E(0, r_i))^{t'_i} = \prod_{i=1}^{n} c'^{t'_i}_i,$$
$$\prod_{i=1}^{n} c_i^{s_i s'_i} \prod_{i=1}^{n} (E(0, r_i))^{t_i t'_i} (E(0, r'_i))^{t'_i} = \prod_{i=1}^{n} c''^{t'_i}_i\ ) \tag{5}$$

If the shuffling party is honest and sets $t_i = s_{\pi(i)}$ and $t'_i = s'_{\pi(i)}$, he can pass the verification as $\sum_{i=1}^{n} t_i D(c'_i) = \sum_{i=1}^{n} s_{\pi(i)} D(c_{\pi(i)}) = \sum_{i=1}^{n} s_i D(c_i)$; $\sum_{i=1}^{n} t'_i D(c'_i) = \sum_{i=1}^{n} s'_{\pi(i)} D(c_{\pi(i)}) = \sum_{i=1}^{n} s'_i D(c_i)$ and $\sum_{i=1}^{n} t_i t'_i D(c'_i) = \sum_{i=1}^{n} s_{\pi(i)} s'_{\pi(i)} D(c_{\pi(i)}) = \sum_{i=1}^{n} s_i s'_i D(c_i)$. Theorem 1 shows that if the shuffling party can pass the verification with a non-negligible probability, his shuffling is correct.

Theorem 1. *If the verifier chooses his challenges s_i and s'_i randomly and the shuffling party in Protocol-1 can provide ZK proofs (3), (4) and (5) with a probability larger than 2^{-L}, there exists a $n \times n$ permutation matrix M such that $(m'_1, m'_2, \ldots, m'_n)M = (m_1, m_2, \ldots, m_n)$ under the linear ignorance assumption against the shuffling party.*

To prove Theorem 1, the following lemmas are proved first.

Lemma 1. *If given random integers s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots, n$, a party can find in polynomial time integers t_i from Z_q for $i = 1, 2, \ldots, n$ with a probability larger than 2^{-L}, such that $\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q$, then he can find in polynomial time a matrix M such that $(m'_1, m'_2, \ldots, m'_n)M = (m_1, m_2, \ldots, m_n)$.*

Proof: Given any integer k in $\{1, 2, \ldots, n\}$ there must exist integers $s_1, s_2, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n$ in $\{0, 1, \ldots, 2^L - 1\}$ and two different integers s_k and $\hat{s}_k$ in $\{0, 1, \ldots, 2^L - 1\}$ such that given $s_1, s_2, \ldots, s_n$ and $\hat{s}_k$, the party can find in polynomial time t_i and $\hat{t}_i$ from Z_q for $i = 1, 2, \ldots, n$ to satisfy the following two equations.

$$\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q \tag{6}$$

$$(\sum_{i=1}^{k-1} s_i m_i)\hat{s}_k m_k \sum_{i=k+1}^{n} s_i m_i = \sum_{i=1}^{n} \hat{t}_i m'_i \bmod q \tag{7}$$

Otherwise, for any $s_1, s_2, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n$ there is at most one s_k to satisfy equation $\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q$. This deduction implies that among the 2^{nL} possible combinations of $s_1, s_2, \ldots, s_n$, the party can find in polynomial time t_i for $i = 1, 2, \ldots, n$ to satisfy $\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q$ for at most $2^{(n-1)L}$ combinations. This conclusion leads to a contradiction: given random integers s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots, n$ the party can find in polynomial time t_i for $i = 1, 2, \ldots, n$ to satisfy $\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q$ with a probability no larger than 2^{-L}.

Subtracting (7) from (6) yields

$$(s_k - \hat{s}_k) m_k = \sum_{i=1}^{n} (t_i - \hat{t}_i) m'_i \bmod q$$

Note that $s_k \in \{0, 1, \ldots, 2^L - 1\}$, $\hat{s}_k \in \{0, 1, \ldots, 2^L - 1\}$, $s_k \neq \hat{s}_k$ and 2^L is no larger than the smallest factor of q. So $s_k - \hat{s}_k \neq 0 \bmod q$. Namely, given a non-zero integer $s_k - \hat{s}_k$, the party can find in polynomial time $t_i - \hat{t}_i$ for $i = 1, 2, \ldots, n$ such that $(s_k - \hat{s}_k) m_k = \sum_{i=1}^{n} (t_i - \hat{t}_i) m'_i \bmod q$. So, for any k in $\{1, 2, \ldots, n\}$ the party knows a vector $V_k = (\ (t_1 - \hat{t}_1)/(s_k - \hat{s}_k),\ (t_2 - \hat{t}_2)/(s_k - \hat{s}_k),\ \ldots,\ (t_n - \hat{t}_n)/(s_k - \hat{s}_k)\ )'$ such that $m_k = (m'_1, m'_2, \ldots, m'_n) V_k$. Therefore, the party can find in polynomial time a matrix M such that $(m_1, m_2, \ldots, m_n) = (m'_1, m'_2, \ldots, m'_n) M$ where $M = (V_1, V_2, \ldots, V_n)$. □

Lemma 2. *If a party can find in polynomial time a $n \times n$ singular matrix M such that $(m'_1, m'_2, \ldots, m'_n) M = (m_1, m_2, \ldots, m_n)$ where $(m_1, m_2, \ldots, m_n)$ and $(m'_1, m'_2, \ldots, m'_n)$ are two vectors, then he can find in polynomial time a linear relation about $m_1, m_2, \ldots, m_n$.*

Proof: Suppose $M = (V_1, V_2, \ldots, V_n)$. Then $m_i = (m'_1, m'_2, \ldots, m'_n) V_i$.

As M is singular and the party can find in polynomial time M, he can find in polynomial time integers $l_1, l_2, \ldots, l_n$ and k such that $\sum_{i=1}^{n} l_i V_i = (0, 0, \ldots, 0)$ where $1 \leq k \leq n$ and $l_k \neq 0 \bmod q$. So

$$\sum_{i=1}^{n} l_i m_i = \sum_{i=1}^{n} l_i (m'_1, m'_2, \ldots, m'_n) V_i = (m'_1, m'_2, \ldots, m'_n) \sum_{i=1}^{n} l_i V_i = 0$$

Namely, the party can find in polynomial time $l_1, l_2, \ldots, l_n$ to satisfy $\sum_{i=1}^{n} l_i m_i = 0$ where $1 \leq k \leq n$ and $l_k \neq 0 \bmod q$. □

Lemma 3. *If a party can find a $n \times n$ non-singular matrix M and integers $l_1, l_2, \ldots, l_n$ and k in polynomial time such that $(m'_1, m'_2, \ldots, m'_n) =$*

$(m_1, m_2, \ldots, m_n)M$, $\sum_{i=1}^n l_i m'_i = 0$, $1 \le k \le n$ and $l_k \ne 0 \bmod q$ where $(m_1, m_2, \ldots, m_n)$ and $(m'_1, m'_2, \ldots, m'_n)$ are two vectors, then he can find a linear relation about $m_1, m_2, \ldots, m_n$ in polynomial time.

Proof: As $(m'_1, m'_2, \ldots, m'_n) = (m_1, m_2, \ldots, m_n)M$ and $\sum_{i=1}^n l_i m'_i = 0$,

$$\sum_{i=1}^{n} l_i (m_1, m_2, \ldots, m_n) V_i = 0 \quad \text{where } M = (V_1, V_2, \ldots, V_n)$$

So

$$(m_1, m_2, \ldots, m_n) \sum_{i=1}^{n} l_i V_i = 0$$

Note that $\sum_{i=1}^n l_i V_i \ne (0, 0, \ldots, 0)$ as M is non-singular, $1 \le k \le n$ and $l_k \ne 0 \bmod q$. Therefore, the party can find a linear relation about $m_1, m_2, \ldots, m_n$ in polynomial time. □

Lemma 4. *If given random integers s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots, n$, a party can find a $n \times n$ non-singular matrix M and integers t_i from Z_q for $i = 1, 2, \ldots, n$ in polynomial time such that $(m_1, m_2, \ldots, m_n) = (m'_1, m'_2, \ldots, m'_n)M$ and $\sum_{i=1}^n s_i m_i = \sum_{i=1}^n t_i m'_i \bmod q$ where $(m_1, m_2, \ldots, m_n)$ and $(m'_1, m'_2, \ldots, m'_n)$ are two vectors, then $(s_1, s_2, \ldots, s_n)M = (t_1, t_2, \ldots, t_n)$ under the linear ignorance assumption against the shuffling party.*

Proof:

$$(m_1, m_2, \ldots, m_n) = (m'_1, m'_2, \ldots, m'_n)M$$

implies

$$m_i = (m'_1, m'_2, \ldots, m'_n)V_i \qquad \text{for } i = 1, 2, \ldots, n$$

where $M = (V_1, V_2, \ldots, V_n)$.

So

$$\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q$$

implies

$$(m'_1, m'_2, \ldots, m'_n) \sum_{i=1}^{n} s_i V_i = (m'_1, m'_2, \ldots, m'_n) \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$$

So given random integers s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots, n$, the party can find matrix $M = (V_1, V_2, \ldots, V_n)$ and integers t_i from Z_q for $i = 1, 2, \ldots, n$ in polynomial time such that

$$(m'_1, m'_2, \ldots, m'_n)(\sum_{i=1}^{n} s_i V_i - \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}) = 0 \tag{8}$$

As M is non-singular,

$$(m'_1, m'_2, \ldots, m'_n) = (m_1, m_2, \ldots, m_n)M^{-1}$$

So

$$\sum_{i=1}^{n} s_i V_i - \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

otherwise according to Lemma 3 the party can find a linear relation about $m_1, m_2, \ldots, m_n$ in polynomial time, which is contradictory to the linear ignorance assumption against the shuffling party. So

$$\sum_{i=1}^{n} s_i V_i = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix} \text{ and thus } M' \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$$

Namely,

$$(s_1, s_2, \ldots, s_n)M = (t_1, t_2, \ldots, t_n)$$

□

Lemma 5. *If* $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ *with a probability larger than* 2^{-L} *for random integers* $s_1, s_2, \ldots, s_n$ *from* $\{0, 1, 2, \ldots, 2^L - 1\}$*, then* $y_i = 0 \bmod q$ *for* $i = 1, 2, \ldots, n$*.*

Proof: Given any integer k in $\{1, 2, \ldots, n\}$, there must exist integers $s_1, s_2, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n$ in $\{0, 1, \ldots, 2^L - 1\}$ and two different integers s_k and $\hat{s}_k$ in $\{0, 1, \ldots, 2^L - 1\}$ such that the following two equations are correct.

$$\sum_{i=1}^{n} y_i s_i = 0 \bmod q \tag{9}$$

$$(\sum_{i=1}^{k-1} y_i s_i) + y_k \hat{s}_k + \sum_{i=k+1}^{n} y_i s_i = 0 \bmod q \tag{10}$$

Otherwise, for any $s_1, s_2, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n$ there is at most one s_k to satisfy equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$. This deduction implies among the 2^{nL} possible combinations of $s_1, s_2, \ldots, s_n$, equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ is correct for at most $2^{(n-1)L}$ combinations. This conclusion leads to a contradiction: given random integers s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots, n$, equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ is correct with a probability no larger than 2^{-L}.

Subtracting (10) from (9) yields

$$y_k(s_k - \hat{s}_k) = 0 \bmod q$$

Note that $GCD(s_k - \hat{s}_k, q) = 1$ as 2^L is no larger than the smallest factor of q, $s_k \neq \hat{s}_k$ and s_k,$\hat{s}_k$ are L-bit integers. So, $y_k = 0 \bmod q$. Note that k can be any integer in $\{1, 2, \ldots, n\}$. Therefore $y_i = 0 \bmod q$ for $i = 1, 2, \ldots, n$. $\square$

Proof of Theorem 1:
According to additive homomorphism of the employed encryption algorithm, ZK proofs (3), (4) and (5) guarantee that the shuffling party can find integers t_i and t'_i for $i = 1, 2, \ldots, n$ to satisfy

$$\sum_{i=1}^{n} s_i m_i = \sum_{i=1}^{n} t_i m'_i \bmod q \tag{11}$$

$$\sum_{i=1}^{n} s'_i m_i = \sum_{i=1}^{n} t'_i m'_i \bmod q \tag{12}$$

$$\sum_{i=1}^{n} s_i s'_i m_i = \sum_{i=1}^{n} t_i t'_i m'_i \bmod q \tag{13}$$

where $m'_i = D(c'_i)$ and s_i and s'_i for $i = 1, 2, \ldots, n$ are randomly chosen by the verifier.

According to Lemma 1, the shuffling party knows a matrix M such that

$$(m'_1, m'_2, \ldots, m'_n)M = (m_1, m_2, \ldots, m_n) \tag{14}$$

According to Lemma 2, M is non-singular under the linear ignorance assumption against the shuffling party.

According to Lemma 4, Equations (14) together with Equations (11), (12) and (13) implies

$$(s_1, s_2, \ldots, s_n)M = (t_1, t_2, \ldots, t_n) \tag{15}$$

$$(s'_1, s'_2, \ldots, s'_n)M = (t'_1, t'_2, \ldots, t'_n) \tag{16}$$

$$(s_1 s'_1, s_2 s'_2, \ldots, s_n s'_n)M = (t_1 t'_1, t_2 t'_2, \ldots, t_n t'_n) \tag{17}$$

under the linear ignorance assumption against the shuffling party.

Equation (15), Equation (16) and Equation (17) respectively imply

$$(s_1, s_2, \ldots, s_n)V_1 = t_1 \tag{18}$$

$$(s'_1, s'_2, \ldots, s'_n)V_1 = t'_1 \tag{19}$$

$$(s_1 s'_1, s_2 s'_2, \ldots, s_n s'_n)V_1 = t_1 t'_1 \tag{20}$$

where $M = (V_1, V_2, \ldots, V_n)$.

Equation (18) and Equation (19) imply

$$(s'_1, s'_2, \ldots, s'_n)V_1 (s_1, s_2, \ldots, s_n)V_1 = t_1 t'_1 \tag{21}$$

Equation (20) and Equation (21) imply

$$(s'_1, s'_2, \ldots, s'_n)V_1(s_1, s_2, \ldots, s_n)V_1 = (s_1 s'_1, s_2 s'_2, \ldots, s_n s'_n)V_1$$

So

$$(s'_1, s'_2, \ldots, s'_n)V_1(s_1, s_2, \ldots, s_n)V_1 = (s'_1, s'_2, \ldots, s'_n)\begin{pmatrix} v_{1,1}s_1 \\ v_{1,2}s_2 \\ \vdots \\ v_{1,n}s_n \end{pmatrix}$$

$$\text{where } V_1 = \begin{pmatrix} v_{1,1} \\ v_{1,2} \\ \vdots \\ v_{1,n} \end{pmatrix}$$

under the linear ignorance assumption against the shuffling party.

Note that $s'_1, s'_2, \ldots, s'_n$ are randomly chosen by the verifier. So according to Lemma 5,

$$V_1(s_1, s_2, \ldots, s_n)V_1 = \begin{pmatrix} v_{1,1}s_1 \\ v_{1,2}s_2 \\ \vdots \\ v_{1,n}s_n \end{pmatrix}$$

under the linear ignorance assumption against the shuffling party. So

$$(s_1, s_2, \ldots, s_n)V_1 v_{1,i} = v_{1,i}s_i \text{ for } i = 1, 2, \ldots, n$$

under the linear ignorance assumption against the shuffling party.

Note that $V_1 \neq (0, 0, \ldots, 0)$ as M is non-singular. So there must exist integer k such that $1 \leq k \leq n$ and $v_{i,k} \neq 0 \bmod q$. So

$$(s_1, s_2, \ldots, s_n)V_1 = s_k$$

under the linear ignorance assumption against the shuffling party. Namely,

$$s_1 v_{1,1} + s_2 v_{1,2} + \ldots + s_n v_{1,n} = s_k \bmod q$$

and thus

$$s_1 v_{1,1} + s_2 v_{1,2} + \ldots + s_{k-1} v_{1,k-1} + (s_k - 1)v_{1,k} + s_{k+1} v_{1,k+1} + \ldots + s_n v_{1,n} = 0 \bmod q$$

under the linear ignorance assumption against the shuffling party.

Note that $s_1, s_2, \ldots, s_n$ are randomly chosen by the verifier. So according to Lemma 5, $v_{1,1} = v_{1,2} = \ldots = v_{1,k-1} = v_{1,k+1} = \ldots = v_{1,n} = 0$ and $v_{1,k} = 1$ under the linear ignorance assumption against the shuffling party. Namely, V_1 contains one 1 and $n - 1$ 0s under the linear ignorance assumption against the shuffling party.

For the same reason, V_i contains one 1 and $n-1$ 0s for $i = 2, 3, \ldots, n$ under the linear ignorance assumption against the shuffling party. Note that M is non-singular. Therefore, M is a permutation matrix under the linear ignorance assumption against the shuffling party. □

In some applications of shuffling like [17], only semantically encrypted ciphertexts $c_1, c_2, \ldots, c_n$ are given to the shuffling party while no information about $m_1, m_2, \ldots, m_n$ is known. So the linear ignorance assumption against the shuffling party (the shuffling party cannot find a linear relation about $m_1, m_2, \ldots, m_n$ in polynomial time) is satisfied. Therefore, the shuffling by Protocol-1 is correct in these applications according to Theorem 1.

2.2 Protocol-2

In Protocol-1, the linear ignorance assumption is necessary. That means Protocol-1 cannot guarantee correctness of the shuffling if someone with knowledge of any shuffled message colludes with the shuffling party. For example, when the shuffling is used to shuffle the votes in e-voting, some voters may collude with the shuffling party and reveal their votes. Then the shuffling party can tamper with some votes without being detected. So Protocol-1 is upgraded to Protocol-2, which can guarantee the linear ignorance and thus correctness of the shuffling without any assumption. The upgrade is simple. The input ciphertexts $c_1, c_2, \ldots, c_n$ are divided into two groups of random ciphertexts $d_1, d_2, \ldots, d_n$ and $e_1, e_2, \ldots, e_n$ such that $c_i = e_i d_i$ for $i = 1, 2, \ldots, n$. Then Protocol-1 can be applied to shuffle $d_1, d_2, \ldots, d_n$ and $e_1, e_2, \ldots, e_n$ using an identical permutation. After the shuffling, the two groups of outputs are combined to recover the re-encrypted permutation of $c_1, c_2, \ldots, c_n$. Protocol-2 is as follows.

1. The shuffling party calculates $d_i = h(c_i)$ for $i = 1, 2, \ldots, n$ where $h()$ is a random oracle query implemented by a hash function from the ciphertext space of the employed encryption algorithm to the same ciphertext space. Thus two groups of ciphertexts d_i for $i = 1, 2, \ldots, n$ and $e_i = c_i / d_i$ for $i = 1, 2, \ldots, n$ are obtained.
2. The shuffling party randomly chooses $\pi()$, a permutation of $\{0, 1, \ldots, n\}$ and integers r_i and u_i from Q for $i = 1, 2, \ldots n$. He then outputs $d'_i = RE(d_{\pi(i)}, r_i)$ and $e'_i = RE(e_{\pi(i)}, u_i)$ for $i = 1, 2, \ldots n$ while concealing $\pi()$.
3. The verifier randomly chooses and publishes s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$. The shuffling party chooses r'_i from Q for $i = 1, 2, \ldots n$ and publishes $d''_i = {d'}_i^{t_i} E(0, r'_i)$ for $i = 1, 2, \ldots n$ where $t_i = s_{\pi(i)}$. The shuffling party publishes ZK proof

$$ZP\ (\ t_i, r'_i \mid d''_i = {d'}_i^{t_i} E(0, r'_i)\ ) \qquad \text{for } i = 1, 2, \ldots n \tag{22}$$

and

$$ZP\ (\ r_i, u_i, t_i, r'_i \text{ for } i = 1, 2, \ldots, n \mid$$

$$\prod_{i=1}^{n} d_i^{s_i} \prod_{i=1}^{n}(E(0,r_i))^{t_i} E(0,r_i') = \prod_{i=1}^{n} d_i'', \quad (23)$$

$$\prod_{i=1}^{n} e_i^{s_i} \prod_{i=1}^{n}(E(0,u_i))^{t_i} = \prod_{i=1}^{n} e'^{t_i}_i \;)$$

4. The verifier randomly chooses and publishes s_i' from $\{0,1,\ldots,2^L-1\}$ for $i=1,2,\ldots n$. The shuffling party sets $t_i' = s'_{\pi(i)}$ for $i=1,2,\ldots n$ and publishes ZK proof

$$ZP\;(\; r_i, t_i, r_i', t_i' \text{ for } i=1,2,\ldots n \mid \prod_{i=1}^{n} d_i^{s_i'} \prod_{i=1}^{n}(E(0,r_i))^{t_i'} = \prod_{i=1}^{n} d'^{t_i'}_i,$$

$$\prod_{i=1}^{n} d_i^{s_i s_i'} \prod_{i=1}^{n}(E(0,r_i))^{t_i t_i'}(E(0,r_i'))^{t_i'} = \prod_{i=1}^{n} d''^{t_i'}_i \;) \quad (24)$$

5. If the proofs above are verified to be valid, the outputs of the shuffling are $c_i' = d_i' e_i'$ for $i=1,2,\ldots n$.

Just like in Protocol-1, if the shuffling party is honest and sets $t_i = s_{\pi(i)}$ and $t_i' = s'_{\pi(i)}$, he can pass the verification in Protocol-2. Theorem 2 shows that if the shuffling party can pass the verification in Protocol-2 with a non-negligible probability, his shuffling is correct even without the linear ignorance assumption.

Theorem 2. *If the verifier chooses his challenges s_i and s_i' randomly and the shuffling party in Protocol-2 can provide ZK proofs (22), (23) and (24) with a probability larger than 2^{-L}, then there is an identical permutation from $D(d_1), D(d_2), \ldots, D(d_n)$ to $D(d_1'), D(d_2'), \ldots, D(d_n')$ and from $D(e_1), D(e_2), \ldots, D(e_n)$ to $D(e_1'), D(e_2'), \ldots, D(e_n')$.*

Proof: According to additive homomorphism of the employed encryption, ZK proofs (22), (23) and (24) guarantee that the shuffling party can find integers t_i and t_i' for $i=1,2,\ldots,n$ to satisfy

$$\sum_{i=1}^{n} s_i D(d_i) = \sum_{i=1}^{n} t_i D(d_i') \bmod q \quad (25)$$

$$\sum_{i=1}^{n} s_i D(e_i) = \sum_{i=1}^{n} t_i D(e_i') \bmod q \quad (26)$$

$$\sum_{i=1}^{n} s_i' D(d_i) = \sum_{i=1}^{n} t_i' D(d_i') \bmod q \quad (27)$$

$$\sum_{i=1}^{n} s_i s_i' D(d_i) = \sum_{i=1}^{n} t_i t_i' D(d_i') \bmod q \quad (28)$$

where s_i and s_i' for $i=1,2,\ldots,n$ are randomly chosen by the verifier.

Note that $d_1, d_2, \ldots, d_n$ are produced by the hash function $h()$, which is regarded as a random oracle. So to find a linear relation about $D(d_1), D(d_2), \ldots, D(d_n)$ is equivalent to repeatedly querying a random oracle for a vector of n random ciphertexts and then finding a linear relation on the plaintexts corrresponding to one of these vectors. This is infeasible as the employed encryption algorithm is semantically secure. So the probability that the shuffling party can find any linear relation about $D(d_1), D(d_2), \ldots, D(d_n)$ is negligible. For the same reason, the probability that the shuffling party can find any linear relation about $D(e_1), D(e_2), \ldots, D(e_n)$ is negligible.

According to Theorem 1, Equations (25), (27) and (28) imply that there exists a permutation matrix M such that

$$(D(d'_1), D(d'_2), \ldots, D(d'_n))M = (D(d_1), D(d_2), \ldots, D(d_n))$$

So according to Lemma 4,

$$(s_1, s_2, \ldots, s_n)M = (t_1, t_2, \ldots, t_n) \tag{29}$$

According to Lemma 1 and Lemma 4, Equation (26) implies that there exists a matrix $\hat{M}$ such that

$$(D(e'_1), D(e'_2), \ldots, D(e'_n))\hat{M} = (D(e_1), D(e_2), \ldots, D(e_n))$$

and

$$(s_1, s_2, \ldots, s_n)\hat{M} = (t_1, t_2, \ldots, t_n) \tag{30}$$

Subtracting (30) from (29) yields

$$(s_1, s_2, \ldots, s_n)(M - \hat{M}) = (0, 0, \ldots, 0)$$

According to Lemma 5, every column vector in matrix $M - \hat{M}$ contains n zeros. So $M = \hat{M}$. Therefore there is an identical permutation (matrix) from $D(d_1), D(d_2), \ldots, D(d_n)$ to $D(d'_1), D(d'_2), \ldots, D(d'_n)$ and from $D(e_1), D(e_2), \ldots, D(e_n)$ to $D(e'_1), D(e'_2), \ldots, D(e'_n)$. □

According to Theorem 2, $D(d_1)D(e_1), D(d_2)D(e_2), \ldots, D(d_n)D(e_n)$ is permuted to $D(d'_1)D(e'_1), D(d'_2)D(e'_2), \ldots, D(d'_n)D(e'_n)$. Namely, $D(c'_1), D(c'_2), \ldots, D(c'_n)$ is a permutation of $D(c_1), D(c_2), \ldots, D(c_n)$ even in the absence of the linear ignorance assumption.

3 Implementation and Cost

The additive homomorphic semantically secure encryption employed in Protocol-1 may be the modified ElGamal encryption [11, 12] or Paillier encryption [16]. The implementation details and computational cost are slightly different with different encryption schemes. For example, the following Paillier encryption algorithm can be employed. $N = p_1 p_2$, $p_1 = 2p'_1 + 1$, $p_2 = 2p'_2 + 1$ where p_1, p_2,

p'_1 and p'_2 are large primes and $GCD(N, p'_1p'_2) = 1$. Integers a, b are randomly chosen from Z_N^* and $g = (1+N)^a + b^N \bmod N$. The public key consists of N and g. The private key is $\beta p'_1 p'_2$ where β is randomly chosen from Z_N^*. A message $m \in Z_N$ is encrypted to $c = g^m r^N \bmod N^2$ where r is randomly chosen from Z_N^*. The modulus of the message space is N. If Paillier encryption is employed, Protocol-1 can be implemented as follows.

1. The shuffling party randomly chooses integers r_i from Z_N^* for $i = 1, 2, \ldots n$. He then outputs $c'_i = c_{\pi(i)} r_i^N \bmod N^2$ for $i = 1, 2, \ldots n$.
2. After the verifier publishes s_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$, the shuffling party chooses r'_i from Z_N^* for $i = 1, 2, \ldots n$ and publishes $c''_i = {c'}_i^{t_i} {r'}_i^N \bmod N^2$ for $i = 1, 2, \ldots n$ where $t_i = s_{\pi(i)}$. The shuffling party publishes ZK proof
$$ZP\ (\ t_i, r'_i \mid c''_i = {c'}_i^{t_i} {r'}_i^N \bmod N^2\ ) \qquad \text{for } i = 1, 2, \ldots n \tag{31}$$
and
$$ZP\ (\ R_1 \mid R_1^N = C_1 \bmod N^2\ ) \tag{32}$$
where $R_1 = \prod_{i=1}^n r_i^{t_i} r'_i \bmod N^2$ and $C_1 = \prod_{i=1}^n c''_i / \prod_{i=1}^n c_i^{s_i} \bmod N^2$.
3. After the verifier publishes s'_i from $\{0, 1, \ldots, 2^L - 1\}$ for $i = 1, 2, \ldots n$, the shuffling party sets $t'_i = s'_{\pi(i)}$ for $i = 1, 2, \ldots n$ and publishes ZK proof
$$\begin{aligned} ZP\ (\ R_2, R_3, t'_i \text{ for } i = 1, 2, \ldots n \mid C_2 R_2^N = \textstyle\prod_{i=1}^n {c'}_i^{t'_i} \bmod N^2, \\ C_3 R_3^N = \textstyle\prod_{i=1}^n {c''}_i^{t'_i} \bmod N^2\ ) \end{aligned} \tag{33}$$
where $R_2 = \prod_{i=1}^n r_i^{t'_i} \bmod N^2$, $R_3 = \prod_{i=1}^n r_i^{t_i t'_i} {r'}_i^{t'_i} \bmod N^2$, $C_2 = \prod_{i=1}^n c_i^{s'_i} \bmod N^2$ and $C_3 = \prod_{i=1}^n c_i^{s_i s'_i} \bmod N^2$.

Non-interactive implementation of ZK proof (31), (32) and (33) can be implemented as follows.

1. The shuffling party randomly chooses $W_1 \in Z_N^*$, $W_2 \in Z_N^*$, $W_3 \in Z_N^*$, $v_i \in Z_N$ for $i = 1, 2, \ldots, n$, $v'_i \in Z_N$ for $i = 1, 2, \ldots, n$ and $x_i \in Z_N^*$ for $i = 1, 2, \ldots, n$. He calculates $a_i = {c'}_i^{v_i} x_i^N \bmod N^2$ for $i = 1, 2, \ldots, n$, $f = W_1^N \bmod N^2$, $a = (\prod_{i=1}^n {c'}_i^{v'_i})/W_2^N \bmod N^2$ and $b = (\prod_{i=1}^n ({c''}^{v'_i})/W_3^N \bmod N^2$.
2. The shuffling party calculates $c = H(f, a, b, a_1, a_2, \ldots, a_n)$ where $H()$ is a random oracle query implemented by a hash function with a 128-bit output.
3. The shuffling party calculates $z_1 = W_1 R_1^c \bmod N^2$, $z_2 = W_2 / R_2^c \bmod N^2$, $z_3 = W_3 / R_3^c \bmod N^2$, $\alpha_i = x_i r_i^c \bmod N^2$ for $i = 1, 2, \ldots, n$, $\gamma_i = v_i + c t_i \bmod N$ for $i = 1, 2, \ldots, n$ and $\gamma'_i = c t'_i - v'_i \bmod N$ for $i = 1, 2, \ldots, n$.
4. The shuffling party publishes $z_1, z_2, z_3, \alpha_1, \alpha_2, \ldots, \alpha_n, \gamma_1, \gamma_2, \ldots, \gamma_n, \gamma'_1, \gamma'_2, \ldots, \gamma'_n$. Anyone can verifiy that
$$\begin{aligned} c = H(\ z_1^N / C_1^c,\ C_2^c / (z_2^N \textstyle\prod_{i=1}^n {c'}_i^{\gamma'_i}),\ C_3^c / (b z_3^N \textstyle\prod_{i=1}^n {c''}_i^{\gamma'_i}), \\ {c'}_i^{\gamma_i} \alpha_i^N / {c''_i}^c \text{ for } i = 1, 2, \ldots, n\ ) \end{aligned} \tag{34}$$

This implementation is a combination of ZK proof of knowledge of logarithm [20], ZK proof of equality of logarithms [5] and ZK proof of knowledge of root [9]. All the three proof techniques are correct and specially sound, so this implementation guarantees Equations (3), (4) and (5). All of the three proof techniques are honest-verifier zero knowledge. So if the hash function can be regarded as a random oracle query, this implementation is zero knowledge. Therefore, ZK privacy is achieved in Protocol-1. In this implementation, the computational cost of shuffling is n full length exponentiations[2]; the cost of proof is $3nExpCost(|N|) + 2ExpCost^n(|N|) + nExpCost(L) + 3ExpCost^n(L) + ExpCost^n(2L) + (n + 3)ExpCost(128) + 3$, which is approximately equal to $11n/3 + 8nL/(3|N|) + 128(n + 3)/|N| + 3$ full length exponentiations.

ZK proofs (22), (23) and (24) in Protocol-2 can be implemented similarly. When Paillier encryption is employed, the computational cost of shuffling is $2n$ full length exponentiations; the cost of proof is approximately equal to $11n/3 + 11nL/(3|N|)+128(n+4)/|N|+3$ full length exponentiations. It is well known [11, 12] that ElGamal encryption can be modified to be additive homomorphic. If the additional DL search in the decryption function caused by the modification is not an efficiency concern (e.g. when the messages are in a known small set), the modified ElGamal encryption can also be applied to our shuffling. An ElGamal-based shuffling only uses ZK proof of knowledge of logarithm [20] and ZK proof of equality of logarithms [5]. Note that in the ElGamal-based shuffling each output ciphertext must be verified to be in the ciphertext space. When a prime p is the multiplication modulus, the ciphertext space is the cyclic subgroup G with order q where q is a prime and $p = 2q+1$. If an output is in $Z_p^* - G$, Proofs (3), (4), (5) cannot guarantee correctness of the shuffling. The implementation and cost of the ElGamal-based shuffling are similar to those of Paillier-based shuffling in both Protocol-1 and Protocol-2.

In summary, both protocols can be efficiently implemented with either Paillier encryption or ElGamal encryption to achieve correctness and privacy in the shuffling.

4 Conclusion

Two new shuffling protocols are proposed in this paper. The first protocol is a prototype and based on an assumption. The second one removes the assumption and can be applied to more applications. Both protocols are simple and efficient, and achieve all the desired properties of shuffling. In Tables 1, the new shuffling protocols based on Paillier encryption are compared against the existing shuffling protocols. It is demonstrated in Table 1 that Protocol-2 is the only shuffling scheme with strict correctness, unlimited permutation, ZK privacy and without the linear ignorance assumption. In Table 1 the computational cost is counted in terms of full-length exponentiations (with 1024-bit exponent) where $L = 20$.

[2] An exponentiation is called full length if the exponent can be as long as the order of the base.

Table 1. Comparison of computation cost in full-length exponentiations

	Correctness	Permutation	Privacy	Linear ignor- -ance assumption	Computation cost (shuffling and proof)	Communication Rounds
[1, 2]	strict	unlimited	ZK	unnecessary	$\geq 16(n \log_2 n - 2n + 2)$	3
[6, 15]	strict	unlimited	not ZK	unnecessary	$10n$	3
[13]	strict	unlimited	not ZK	unnecessary	$12n$	7
[8][a]	not strict	unlimited	ZK	unnecessary	$8n + 3n/\kappa + 3$	7
[19][b]	strict	limited	ZK	necessary	$2n + k(4k - 2)$	3
Protocol-1	strict	unlimited	ZK	necessary	$n + \frac{369}{96}n + \frac{27}{8} < 5n$	3
Protocol-2	strict	unlimited	ZK	unnecessary	$2n + \frac{3077}{768}n + 3.5 \approx 6n$	3

[a] κ is a chosen parameter.
[b] k is a small parameter determined by the flexibility of permutation and strength of privacy.

It is demonstrated that the new shuffling protocols are more efficient than the existing shuffling schemes except [19], which is not a complete shuffling.

Acknowledgements

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

References

1. M Abe. Mix-networks on permutation net-works. In *ASIACRYPT '98*, volume 1716 of *Lecture Notes in Computer Science*, pages 258–273, Berlin, 1999. Springer-Verlag.
2. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 317–324, Berlin, 2001. Springer-Verlag.
3. M Bellare, J A Garay, and T Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250, Berlin, 1998. Springer-Verlag.
4. D Chaum. Untraceable electronic mail, return address and digital pseudonym. *Communications of the ACM, 24(2)*, pages 84–88, 1981.
5. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1992. Springer-Verlag.
6. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, Berlin, 2001. Springer.
7. Eran Gabber, Phillip B. Gibbons, Yossi Matias, and Alain Mayer. How to make personalized web browsing simple, secure, and anonymous. In *Proceedings of Financial Cryptography 1997*, volume 1318 of *Lecture Notes in Computer Science*, pages 17–31, Berlin, 1997. Springer.

8. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160, Berlin, 2003. Springer-Verlag.
9. L. C. Guillou and J. J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Berlin, 1989. Springer-Verlag.
10. J.Furukawa, H.Miyauchi, K.Mori, S.Obana, and K.Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Proceedings of Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30, Berlin, 2002. Springer.
11. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *JW-ISC 2000*, pages 101–108, 2000.
12. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Information Security and Cryptology, ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 389–406, Berlin, 2002. Springer-Verlag.
13. 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.
14. Lan Nguyen and Rei Safavi-Naini. An efficient verifiable shuffle with perfect zero-knowledge proof system. In *Cryptographic Algorithms and their Uses 2004*, pages 40–56, 2004.
15. Lan Nguyen, Rei Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In *Applied Cryptography and Network Security, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 61–75, Berlin, 2004. Springer-Verlag.
16. P Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, 1999. Springer-Verlag.
17. Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. An efficient and verifiable solution to the millionaire problem. In *Pre-Proceedings of ICISC 2004*, pages 315–330, 2004.
18. Kun Peng, Colin Boyd, Edward Dawson, and Kapali Viswanathan. Efficient implementation of relative bid privacy in sealed-bid auction. In *The 4th International Workshop on Information Security Applications, WISA2003*, volume 2908 of *Lecture Notes in Computer Science*, pages 244–256, Berlin, 2003. Springer-Verlag.
19. Kun Peng, Colin Boyd, Edward Dawson, and Kapali Viswanathan. A correct, private and efficient mix network. In *2004 International Workshop on Practice and Theory in Public Key Cryptography*, pages 439–454, Berlin, 2004. Springer-Verlag.
20. C Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology, 4, 1991*, pages 161–174, 1991.