Noname manuscript No.
(will be inserted by the editor)

Verifiable Shuffles: A Formal Model and a Paillier-based 3-Round Construction with Provable Security [⋆]

Lan Nguyen[1], **Rei Safavi-Naini**[1], **Kaoru Kurosawa**[2]

[1] School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
e-mail: `{ldn01,rei}@uow.edu.au`
[2] Department of Computer and Information Sciences
Ibaraki University 4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan
e-mail: `kurosawa@cis.ibaraki.ac.jp`

The date of receipt and acceptance will be inserted by the editor

Abstract We propose a formal model for security of verifiable shuffles and a new verifiable shuffle system based on the Paillier encryption scheme, and prove its security in the proposed model. The model is general, so it can be extended to verifiable shuffle decryption and provides a direction for provable security of mix-nets.

Key words privacy, verifiable shuffles, formal security model, mix-nets, Paillier public-key system.

1 Introduction

A *shuffle* takes an input list of ciphertexts and outputs a permuted and re-encrypted version of the input list. Re-encryption of a ciphertext can be defined for encryption systems such as the El Gamal and Paillier encryption systems, and allows generation of ciphertexts c' from a given ciphertext c such that both ciphertexts correspond to the same plaintext m under the same public key.

The main application (motivation for the study) of shuffles is to construct *mix-nets*, a cryptographic system introduced by Chaum [8] for providing communication unlinkability and anonymity. Mix-nets are among the most widely used systems for providing communication privacy, and have found applications in anonymous email systems [8], Web browsing

[⋆] This paper is the extended version of the paper [35] presented at ACNS '04.

[15], electronic voting [40,32,26], anonymous payment systems [21,9], location privacy for mobile networks [28] and mobile IP [9], secure multiparty computation [24] and privacy in advertisements [27].

A mix-net consists of a number of mix-centres that collectively permute and decrypt the mix-net's input list. Shuffles are used to implement mix-centres. A basic shuffle permutes its input list of ciphertexts through re-encryption. Mix-centres may also partially decrypt the list [2], hence called *shuffle decryption*. Mix-nets that use shuffle decryption could be more efficient but in case a mix-centre fails, they need more computation to recover [14].

The main security objective of shuffle systems is to provide *unlinkability* between its input elements and its output elements for outsiders, and so effectively keeping the permutation secret. It is referred to as *shuffle privacy*. Another important property of shuffles is *verifiability*: that is providing a proof that the output is correctly constructed. Verifiability of shuffles is used to provide *robustness* for the mix-net: that is ensuring that the mix-net works correctly even if a number of mix-servers are malicious. This is an important property of mix-nets and so verifiability of shuffles has received much attention. Shuffles must be efficient and the cost is measured in terms of computation and communication that is required to provide privacy for n users.

In this paper, we focus on *verifiable shuffles*. Privacy of shuffles has traditionally been equated to the zero-knowledge property of the proof system used for verifying correctness. Recently a number of efficient constructions for verifiable shuffles have been proposed [1–3,23,37]. In Crypto'01, Furukawa and Sako [12] gave a characterisation of permutation matrices in terms of two equations that can be efficiently proved, hence proposing an efficient (3 round proof system) verifiable shuffle scheme. However, in a subsequent paper [13], they noted that the proof system was not correctly proved to be zero-knowledge. They however gave a definition of privacy for shuffles and showed that their shuffle scheme satisfied this definition. The definition requires that the verifier cannot learn anything about the 'relation' between the shuffle's output and its input using the transcript of the proof system. Neff [32,33] and later Groth [19] proposed shuffles that provide zero-knowledge property for their proofs. However, like the Furukawa-Sako scheme, the zero-knowledgeness of Neff's protocol has not been correctly proved and still remains an open problem [36].

As noted above, the notion of privacy varies among shuffles and no formal model for verifiable shuffles has been suggested so far. Such a formalisation could also be important for formalising security of mix-nets. Recently proposed attacks [4,34,46] against mix-nets clearly demonstrate the need for such a model.

The *first contribution* of this paper is to give a formal model for shuffles that allows us to have a unified approach for assessment of shuffle systems. Our definition of shuffle privacy is motivated by observing the similarity between a shuffle hiding the permutation, and an encryption system hiding the input message. We consider adaptive attacks by an active adversary that uses a *chosen permutation attack (CPA$_S$)* (similar to chosen plaintext) and a *chosen transcript attack (CTA$_S$)* (similar to chosen ciphertext). A difference between this model and the model of a traditional encryption system is that in this case the adversary does not only specify distribution of the challenge permutation (i.e. plaintext) but also another input, the list of input ciphertexts. We allow the adversary to choose this input ciphertext list adaptively and also know the corresponding plaintext list. Using this approach, notions of privacy can be defined in line with semantic security and indistinguishability. We prove that these two notions of privacy are equivalent and can be interchangeably used. The definition of verifiability is based on the notion of completeness and soundness of the proof system. This is the first complete model for shuffle security with an active adversary and under CPA$_S$ and CTA$_S$. The model can be extended to verifiable shuffle decryption. Our future work is to extend it to a formal model of mix-nets, and so providing a unified framework for security evaluation of these systems.

A *second contribution* of this paper is proposing a new 3-round verifiable shuffle scheme based on the Paillier encryption system [39] and using the shuffle scheme to construct a robust mix-net. The Paillier encryption system provides semantic security against adaptive chosen plaintext attacks (CPA) in the standard model and similar to the El Gamal cryptosystem, it is possible to define a re-encryption algorithm for it. The shuffle uses Furukawa-Sako's approach for characterisation of permutation matrices but has computations over a composite modulus which complicates security proofs (We have to prove Theorem 5 and Theorem 8). We prove in our proposed model that the shuffle provides verifiability and privacy against chosen permutation attacks, but not chosen transcript attacks. Similar to Groth's scheme, our shuffle scheme's security relies on assumptions (Computational Composite Residuosity and Decisional Composite Residuosity) different from assumptions (Discrete Log and Decisional Diffie-Hellman) underlying security of the Furukawa-Sako and Neff schemes.

Related Works

Furukawa et al. [13] gave a definition of privacy for security evaluation of their shuffle but this definition is for a passive adversary and does not model corruption of input messages by senders or a-priori partial information about the permutation. Wikstrom [45] proposed a notion of privacy for non-verifiable shuffles for an adversary who has access to an input and an output of the shuffle. None of the above formally defines verifiability of shuffles. Furukawa [14] provided a formal security model for shuffle decryption systems. In [4], a formalisation of mix-nets and their security requirements was proposed.

An efficient verifiable shuffle construction was given by Neff [32]. This construction is based on a generalisation of Chaum-Pedersen knowledge proof of equality of discrete logarithms and uses the fact that a polynomial of degree n has at most n roots. An improved version of this scheme is given later [33]. Based on Neff's method, Groth [19] proposed a very efficient scheme that uses homomorphic commitments. The input ciphertexts in this scheme can be encrypted by any homomorphic cryptosystem. The above schemes guarantee privacy using the zero-knowledge property of the proof system. A recent direction in designing mix-nets has been to trade off some privacy or correctness for efficiency [6,18,26].

The organization of the paper is as follows. In section 2, we recall some background on public-key encryption schemes and shuffles. Section 3 provides our formal definitions of verifiable shuffles and its security requirements. Section 4 gives a verifiable shuffle scheme based on the Paillier public-key system, its security proofs and efficiency analysis. The next section constructs a robust mix-net from the verifiable shuffle scheme. Section 6 concludes the paper.

2 Background

2.1 Notations and Terminology

Let *lcm* and *gcd* stand for 'least common multiple' and 'greatest common divisor', respectively. For a set $\mathbf{S}$, $|\mathbf{S}|$ denotes the number of elements in the set and "$x \leftarrow \mathbf{S}$" denotes an element x uniformly chosen from $\mathbf{S}$. $\{Element|Conditions\}$ denotes the set of $Element$s satisfying the $Conditions$. Let "$\mathsf{Pr}[Predicate]$" denote the probability that $Predicate$ is true. For a list L, $|L|$ denotes the size of the list, $L[i]$ denotes the i^{th} element of the list and $\pi(L)$ denotes the list of elements in L permuted by a permutation π. Let S_n denote the set of all permutations on $\{1, ..., n\}$. Let $poly(n)$ refer to some fixed but unspecified polynomial and U_n denote a random variable uniformly distributed over $\{0,1\}^n$. Let PT denote *polynomial-time*, PPT denote *probabilistic* PT and DPT denote *deterministic* PT.

An algorithm $\mathcal{A}$ can simply be viewed as a machine that takes as input a string x, performs some operations and outputs a string y. We write $y \leftarrow \mathcal{A}(x)$ and denote $C_{\mathcal{A}}^{x,y}$ the probabilistic input (sequence of internal random coin tosses) of $\mathcal{A}$. For example, in subsection 2.3 below, if Paillier ciphertext $g = r^N(1+mN)$, then $C_E^{(N,m),g} = r$. We can abuse this notation for algorithms in subsection 2.2 by writing $C_{E_{pk}}^{m,c}$ instead of $C_E^{(pk,m),c}$ for an encryption algorithm E and similar for a decryption algorithm D_{sk} and a re-encryption algorithm R_{pk}. We use $C_{\mathcal{A}}^{L_x,L_y}$ to denote the list of probabilistic inputs of $\mathcal{A}$ where the i^{th} element of the list is the probabilistic input that takes the i^{th} element of the list L_x to the i^{th} element of the list L_y. The set of possible outputs of $\mathcal{A}$ on input x is denoted by $[\mathcal{A}(x)]$. For a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$, if for every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$, it holds that $f(l) < l^{-\alpha}$, then f is said to be *negligible*. A problem is said to be *computationally difficult* if for every PT algorithm, the probability that the PT algorithm can solve the problem is a negligible function.

The adversary is modelled by an *oracle machine* which is a Turing machine with additional tapes and states allowing access to some *oracles* that provide answers to queries of defined types. An interactive proof system $(\mathcal{P}, \mathcal{V})$ consists of two parties: a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Each party can be modelled by an *interactive machine*, which is a Turing machine with additional tapes and states allowing joint communication and computation with another interactive machine. Formal descriptions of oracle machines and interactive machines can be found in [16]. For a proof system $(\mathcal{P}, \mathcal{V})$, the verifier's view $View_{\mathcal{V}}^{\mathcal{P}}(x)$ denotes all that $\mathcal{V}$ can see from the execution of the proof system on input x (in other words, all input tapes of the verifier).

2.2 Public-key Encryption Schemes

2.2.1 Syntax A public-key encryption scheme consists of a *key generation* algorithm G, an *encryption* algorithm E and a *decryption* algorithm D. It is denoted by (G, E, D).

- Key generation: The PPT algorithm G on input 1^l outputs (pk, sk) where pk is the public key, sk is the secret key and l is a security parameter. It is denoted by $(pk, sk) \leftarrow G(1^l)$.
- Encryption: The PPT algorithm E takes as input the public key pk and a plaintext m and outputs a ciphertext c. It is denoted by $c \leftarrow E(pk, m)$ or $c \leftarrow E_{pk}(m)$.
- Decryption: The DPT algorithm D takes as input the secret key sk and a ciphertext c and outputs a plaintext such that if $c \leftarrow E_{pk}(m)$ then $D_{sk}(c) = m$, where $D_{sk}(c)$ (or $D(sk, c)$) denotes the output of D on input sk and c.

A public-key encryption scheme, such as the El Gamal and Paillier schemes, may have a *re-encryption* algorithm. Following the definition in [45], this means there is a PPT algorithm R that takes as input the public key pk and a ciphertext and outputs another ciphertext such that for every plaintext m and its ciphertexts c and c':

$$Pr[c' = R_{pk}(c)] = Pr[c' = E_{pk}(m)] \tag{1}$$

where $R_{pk}(c)$ (or $R(pk, c)$) denotes the output of R on input pk and c. A public-key encryption scheme with a re-encryption algorithm is denoted by (G, E, D, R).

2.2.2 Security We briefly recall definitions and notions of security used in this paper and more details can be found in [17]. There are two equivalent notions of security for encryption against *chosen plaintext attacks*, *semantic security* (SS-CPA) and *indistinguishability* (IND-CPA). A chosen plaintext attack means that the adversary can obtain ciphertexts corresponding to plaintexts that he adaptively chooses. Semantic security intuitively means that whatever the adversary is able to compute about the plaintext from a challenge ciphertext, can also be computed without the ciphertext. Indistinguishability means that it is computationally infeasible to distinguish encryptions of two plaintexts of the same length.

There are also two equivalent definitions of encryption security against *chosen ciphertext attacks*, *semantic security* (SS-CCA) and *indistinguishability* (IND-CCA). A chosen ciphertext attack means that the adversary can obtain plaintexts corresponding to ciphertexts that he adaptively chooses, even after the challenge ciphertext is given. Another type of security requirement is *non-malleability* which means that given a ciphertext, it is computationally infeasible to generate a different ciphertext such that the corresponding plaintexts are related in a known manner. It has been proved [17] that non-malleability against chosen ciphertext attacks (NM-CCA) is equivalent to SS-CCA and IND-CCA.

2.3 Paillier Public-key System

Key generation: Let $N = pq$, where p and q are large primes, and $\lambda = lcm(p-1, q-1)$. The public key is $pk = N$ and the secret key is $sk = \lambda$. Hereafter, unless stated otherwise, we assume all modular computations are in modulo N^2.
Encryption: Plaintext $m \in \mathbb{Z}_N$ can be encrypted by choosing $r \leftarrow \mathbb{Z}_N^*$ and computing the ciphertext $g = r^N(1 + mN)$.
Re-encryption: A Paillier ciphertext g for a plaintext m can be re-encrypted as $g' = r'^N g$ for the same plaintext m, where $r' \leftarrow \mathbb{Z}_N^*$. The re-encryption algorithm satisfies the condition (1) above.
Decryption: Ciphertext $g \in \mathbb{Z}_{N^2}^*$ can be decrypted as $m = L(g^\lambda \bmod N^2)/\lambda \bmod N$, where the function L takes its input from the set $\{u \in \mathbb{Z}_{N^2} | u = 1 \bmod N\}$ and is defined as $L(u) = (u-1)/N$.
Computational Composite Residuosity (CCR) Assumption: Suppose $z \leftarrow \mathbb{Z}_{N^2}^*$ is given, the Computational Composite Residuosity problem is to find $x \in \mathbb{Z}_N$ such that there exists $r \in \mathbb{Z}_N^*$ satisfying $z = r^N(1 + xN) \bmod N^2$. The CCR assumption states that the CCR problem is computationally difficult.
Decisional Composite Residuosity (DCR) Assumption: A number $z \in \mathbb{Z}_{N^2}^*$ is said to be a w^{th} *residue mod* N^2 if there exists a number $y \in \mathbb{Z}_{N^2}^*$ such that $z = y^w$. Let $\mathcal{W}_N$ denote the set of N^{th} residues modulo N^2. The Decisional Composite Residuosity problem is to distinguish between an element uniformly chosen from the set $\mathcal{W}_N$ and an element uniformly chosen from the set $\mathbb{Z}_{N^2}^*$. The DCR assumption states that the DCR problem is computationally difficult.
Security: Theorem 1 states security of the Paillier scheme and its proof can be found in [39].

Theorem 1 *The Paillier encryption scheme provides SS-CPA if and only if the DCR assumption holds.*

NM-CCA robust threshold encryption scheme: Using the twin encryption paradigm [31], the Shamir secret sharing scheme [43], the proof of equality of discrete logs and a simulation-sound proof of equality of plaintexts, Fouque and Pointcheval [11] proposed an NM-CCA robust threshold encryption scheme based on the Paillier public-key system that is proved secure in the random oracle model. We use this encryption system to construct a robust mix-net.

2.4 Furukawa-Sako Shuffle

Furukawa and Sako [12] proposed an efficient verifiable shuffle scheme based on the El Gamal public-key system. In their scheme, a permutation is represented as a permutation matrix, which is defined in Definition 1. Their proof system is based on Theorem 2, which states two conditions to achieve a permutation matrix.

Definition 1 *A matrix $(A_{ij})_{n\times n}$ is a permutation matrix modulo k if it satisfies the following for some permutation π*

$$A_{ij} = \begin{cases} 1 \ mod\ k \ if\ \pi(i) = j \\ 0 \ mod\ k\ otherwise \end{cases}$$

Theorem 2 *A matrix $(A_{ij})_{n\times n}$ is a permutation matrix modulo q, where q is a prime, if and only if for all i, j and k, both*

$$\sum_{l=1}^{n} A_{li}A_{lj} = \begin{cases} 1 \ mod\ q \ if\ i = j \\ 0 \ mod\ q\ otherwise \end{cases}$$

$$\sum_{l=1}^{n} A_{li}A_{lj}A_{lk} = \begin{cases} 1 \ mod\ q \ if\ i = j = k \\ 0 \ mod\ q\ otherwise \end{cases}$$

hold.

3 Security of Verifiable Shuffles

3.1 Syntax of Shuffles

First, we define a language to describe that a list of ciphertexts is a permuted and re-encrypted version of another ciphertext list.

Definition 2 *Suppose $\mathcal{RP} = (G, E, D, R)$ is a public key encryption scheme with a re-encryption algorithm. Define a language $\mathcal{L}_{\mathcal{RP}}$ of tuples (pk, L_1, L_2) such that pk is a public key generated by G and L_2 is a permutation of re-encryptions of ciphertexts in L_1 produced by R_{pk}. The witness $w(pk, L_1, L_2)$ includes the permutation and the list of probabilistic inputs of R_{pk}.*

$$\begin{aligned} \mathcal{L}_{\mathcal{RP}} &= \{(pk, L_1, L_2) \mid (|L_1| = |L_2|) \wedge \\ &\quad (\exists \pi \in S_{|L_1|}, \forall i \in \{1, ..., |L_1|\} : L_2[\pi(i)] \in [R_{pk}(L_1[i])])\} \\ w(pk, L_1, L_2) &= (\pi, C_{R_{pk}}^{\pi(L_1), L_2}) \end{aligned}$$

A shuffle takes a list of ciphertexts and outputs a permuted list of their re-encryptions. If being verifiable, it then runs a proof system for the language $\mathcal{L}_{\mathcal{RP}}$ to prove that the output is really a permutation of the re-encryptions of the input ciphertexts. This can be formally defined as follows.

Definition 3 *A shuffle is a pair, $(\mathcal{RP}, S)$, such that:*

- *$\mathcal{RP}$ is a public-key encryption scheme with a re-encryption algorithm (G, E, D, R). Suppose the key generation algorithm G generates a pair (pk, sk).*
- *The PPT algorithm S takes as input a public key pk, a list of n input ciphertexts L_{in} and a random permutation $\pi \in S_n$, and outputs a list of n output ciphertexts L_{out}. S performs* correctly *if L_{out} is a list of re-encryptions of ciphertexts in L_{in} permuted by π.*

Definition 4 *A verifiable shuffle is a tuple, $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$, such that:*

- *$\mathcal{RP}$ and S are defined as in Definition 3.*
- *The proof system $(\mathcal{P}, \mathcal{V})$ takes input pk, L_{in} and L_{out} from S and proves that $(pk, L_{in}, L_{out}) \in \mathcal{L}_{\mathcal{RP}}$. The private input to $\mathcal{P}$ includes only the witness $w(pk, L_{in}, L_{out})$ and does not include the private key sk.*

3.2 Security Definitions

There are 2 security requirements. Privacy requires an honest shuffle to protect its secret permutation whereas verifiability requires that any attempt by a malicious shuffle to produce an incorrect output must be detectable.

We assume an honest verifier for the proof system $(\mathcal{P}, \mathcal{V})$.

3.2.1 Verifiability The proof system proves that the shuffle output is a permutation of re-encryptions of the input ciphertexts. In other words, it is a proof system for the language $\mathcal{L}_{\mathcal{RP}}$. The proof system should satisfy two conditions, *completeness* and *soundness*. The completeness condition states that for all $x \in \mathcal{L}_{\mathcal{RP}}$, the proof system accepts with overwhelming probability. The soundness condition means that for all $x \notin \mathcal{L}_{\mathcal{RP}}$, the proof system accepts with negligible probability. In both definitions of completeness and soundness, we capture the non-uniform capability of the adversary by using a (nonuniform) auxiliary input t.

In the soundness definition, A can be cooperative with B and passes its internal state information δ to B. The private input y of the prover includes information about the lists of plaintexts $L_{in}^{(p)}$ and $L_{out}^{(p)}$ and the corresponding probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$ and $C_{E_{pk}}^{L_{out}^{(p)}, L_{out}}$.

Definition 5 *A shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is verifiable if its proof system $(\mathcal{P}, \mathcal{V})$ has a polynomial-time $\mathcal{V}$ and satisfies two conditions:*

– *Completeness: For every PPT algorithm A and every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$\mathsf{Pr}\left[\begin{array}{l}\langle \mathcal{P}(y), \mathcal{V}\rangle(pk, L_{in}, L_{out}) = 1 \text{ given } (pk, L_{in}, L_{out}) \in \mathcal{L}_{\mathcal{RP}} \\ \text{where } \ (pk, sk) \leftarrow G(1^l), \\ \qquad\quad (L_{in}, L_{out}) \leftarrow A(pk, t), \\ \qquad\quad y \leftarrow w(pk, L_{in}, L_{out}) \end{array}\right]$$

$$> 1 - l^{-\alpha}$$

The probability is taken over the internal coin tosses of G, A, $\mathcal{P}$ and $\mathcal{V}$.

– *Soundness: For every interactive machine $\mathcal{B}$, every PPT algorithm A and every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$\mathsf{Pr}\left[\begin{array}{l}\langle \mathcal{B}(y), \mathcal{V}\rangle(pk, L_{in}, L_{out}) = 1 \text{ given } (pk, L_{in}, L_{out}) \notin \mathcal{L}_{\mathcal{RP}} \\ \text{where } \ (pk, sk) \leftarrow G(1^l), \\ \qquad\quad (\pi, L_{in}, L_{out}, \delta) \leftarrow A(pk, t), \\ \qquad\quad y \leftarrow (\delta, \pi, L^{(p)}_{in}, C^{L^{(p)}_{in}, L_{in}}_{E_{pk}}, L^{(p)}_{out}, C^{L^{(p)}_{out}, L_{out}}_{E_{pk}}) \end{array}\right] < l^{-\alpha}$$

The probability is taken over the internal coin tosses of G, A, $\mathcal{B}$ and $\mathcal{V}$.

We also define *strong-soundness*, where the private input y of the prover includes the private key sk. This condition intuitively requires that even if a malicious shuffle knows the private key, it can not produce an incorrect output without being detected by the proof system. A shuffle, which provides completeness and strong-soundness, is said to provide *strong-verifiability.* In the strong-soundness definition, B may also know sk as A can include sk in its state information δ.

Definition 6 *A shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides strong-verifiability if its proof system $(\mathcal{P}, \mathcal{V})$ has a polynomial-time $\mathcal{V}$ and satisfies two conditions:*

– *Completeness: As in Definition 5*
– *Strong-soundness: For every interactive machine $\mathcal{B}$, every PPT algorithm A and every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$\mathsf{Pr}\left[\begin{array}{l}\langle \mathcal{B}(y), \mathcal{V}\rangle(pk, L_{in}, L_{out}) = 1 \text{ given } (pk, L_{in}, L_{out}) \notin \mathcal{L}_{\mathcal{RP}} \\ \text{where } \ (pk, sk) \leftarrow G(1^l), \\ \qquad\quad (\pi, L_{in}, L_{out}, \delta) \leftarrow A(pk, sk, t), \\ \qquad\quad y \leftarrow (\delta, \pi, L^{(p)}_{in}, C^{L^{(p)}_{in}, L_{in}}_{E_{pk}}, L^{(p)}_{out}, C^{L^{(p)}_{out}, L_{out}}_{E_{pk}}) \end{array}\right] < l^{-\alpha}$$

The probability is taken over the internal coin tosses of G, A, $\mathcal{B}$ and $\mathcal{V}$.

3.2.2 Privacy The shuffle can be considered as a public key transformation that hides the permutation through re-encryption. This can be viewed as 'encryption' of permutation through the process of re-encryption hence notions of 'concealment' of plaintexts in encryption systems can be used to model privacy. We consider 2 types of *adaptive attacks* by active adversaries. *Chosen permutation attack (CPA_S)* is similar to chosen plaintext attacks and the adversary can obtain transcripts of the shuffle executions corresponding to permutations that the adversary adaptively chooses. *Chosen transcript attack (CTA_S)* is similar to chosen ciphertext attacks and the adversary can obtain permutations that correspond to valid shuffle transcripts that the adversary adaptively chooses. The transcript of a verifiable shuffle's execution consists of the lists of input ciphertexts and output ciphertexts and the verifier's view of the proof system. An adaptive attack has 4 steps.

1. *Key generation:* A trusted party generates the keys $(pk, sk) \leftarrow G(1^l)$. The adversary is given $(1^l, pk)$. (sk is used for decryption and is not given to the shuffle.)
2. *Oracle queries:* The adversary adaptively uses the information obtained so far to make queries to some oracles. Different sets of oracles determine different types of attacks (CPA_S and CTA_S). After making a number of such queries, the adversary moves to the next stage.
3. *Challenge generation:* Using the information obtained so far, the adversary specifies a *challenge template*, according to which an *actual challenge* will be generated.
4. *Additional oracle queries:* Based on the information obtained so far, the adversary makes additional queries as in Step 2 and then, produces an output and halts.

The adversary's strategy consists of two stages, each represented by a PPT oracle machine, and corresponding to its action before and after generation of the actual challenge. The first part, denoted by A_1, captures the adversary's behavior during Step 2 and 3. A_1 is given the public key pk, and its output is a pair (τ, δ), where τ is the challenge template generated at the beginning of Step 3 and δ is the state information passed to the second part of the adversary. The second part of the adversary, denoted by A_2, captures the adversary's behavior during Step 4. A_2 is given the state information δ and the actual challenge o generated in Step 3, and produces the adversary's output v. We let each oracle machine to have a (nonuniform) auxiliary input t. This is to capture the nonuniform power of the adversary. It suffices to give t to only the first machine as A_1 can pass this input to the second machine as part of the state information δ. A similar argument shows that it suffices to provide the public key pk only to A_1. We write $(\tau, \delta) \leftarrow A_1^{Oracles}(pk, t)$, and $v \leftarrow A_2^{Oracles}(\delta, o)$, where *Oracles* specify oracles that are available to the adversary.

Notions of Privacy: We consider two notions of privacy. *Semantic privacy* formalizes the intuition that whatever is computable about the permutation from a shuffle execution's transcript must be also computable without the transcript. In formalising this notion under CPA$_S$ and CTA$_S$, we consider the following challenge templates. The challenge template includes a PPT algorithm Π_n, two polynomially bounded functions h_n and f_n, a list of n ciphertexts L_{in}, the list of n corresponding plaintexts $L_{in}^{(p)}$ and the list of probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)},L_{in}}$. Π_n specifies a distribution on the set S_n (of all permutations on $\{1,...,n\}$): it takes a $poly(l)$-bit input (l is the security parameter) and outputs a permutation $\pi \in S_n$. The information regarding the permutation that the adversary tries to obtain is captured by f_n, whereas the a-priori partial information about the permutation is captured by h_n. The actual challenge includes the list of output ciphertexts L_{out}, the verifier's view of the proof system $View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out})$, the list of n input ciphertexts L_{in}, the list of n corresponding plaintexts $L_{in}^{(p)}$, the list of probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)},L_{in}}$ and the partial information $h_n(\pi)$. The inclusion of $L_{in}^{(p)}$ and $C_{E_{pk}}^{L_{in}^{(p)},L_{in}}$ models the fact that the adversary can somehow know all the plaintexts of the input ciphertexts to the shuffle. The adversary's goal is to guess $f_n(\pi)$.

The second notion of privacy is *indistinguishability* and means that it is infeasible to distinguish transcripts of two shuffle executions that correspond to two permutations of the same size. In the definitions of IND-CPA$_S$ and IND-CTA$_S$, the challenge template consists of a pair of permutations $\pi_{(1)}, \pi_{(2)} \in S_n$, a list of n ciphertexts L_{in}, the list of n corresponding plaintexts $L_{in}^{(p)}$ and the list of probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)},L_{in}}$. The actual challenge is the transcript of the shuffle execution corresponding to one of the permutations and consists of the list of output ciphertexts L_{out}, the verifier's view of the proof system $View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out})$, the lists of input ciphertexts L_{in} and the corresponding plaintexts $L_{in}^{(p)}$, and the probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)},L_{in}}$ of the input ciphertexts. The adversary's goal is to distinguish the two possible cases.

Attacks: We consider two attacks.

(Chosen permutation attack) The adversary can do the following experiment for many times. It takes a permutation and a list of input ciphertexts and simulates a ciphertext list output by the algorithm S and corresponding to the input list, and the verifier's view of the proof system $(\mathcal{P}, \mathcal{V})$ when the shuffle interacts with an honest verifier. The adversary does not have access to an oracle. We provide the definitions of SP-CPA$_S$ and IND-CPA$_S$ as follows.

Definition 7 *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to provide* semantic privacy under chosen permutation attacks (SP-CPA$_S$) *if for every pair of PPT algorithms, A_1 and A_2, there exists a pair of PPT algorithms, A_1' and A_2', such that the following two conditions hold:*

1. *For every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$Pr\left[\begin{array}{l} v = f_n(\pi) \text{ where} \\ \qquad\qquad (pk, sk) \leftarrow G(1^l), \\ \qquad\qquad ((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1(pk, t), \\ \qquad\qquad L_{out} \leftarrow S(pk, L_{in}, \pi) \text{ where } \pi \leftarrow \Pi_n(U_{poly(l)}), \\ \qquad\qquad o \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, \\ \qquad\qquad C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi)), \\ \qquad\qquad v \leftarrow A_2(\delta, o) \end{array}\right]$$

$$< Pr\left[\begin{array}{l} v = f_n(\pi) \text{ where} \\ \qquad\qquad ((\Pi_n, h_n, f_n), \delta) \leftarrow A_1'(1^l, t), \\ \qquad\qquad \pi \leftarrow \Pi_n(U_{poly(l)}), \\ \qquad\qquad v \leftarrow A_2'(\delta, h_n(\pi)) \end{array}\right] + l^{-\alpha}$$

2. *For every l and t above, the parts (Π_n, h_n, f_n) generated from $A_1(pk, t)$ and $A_1'(1^l, t)$ are identically distributed.*

Definition 8 *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to provide indistinguishability under chosen permutation attacks (IND-CPA$_S$) if for every pair of PPT algorithms, A_1 and A_2, for every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$|p_{l,t}^{(1)} - p_{l,t}^{(2)}| < l^{-\alpha}$$

where

$$p_{l,t}^{(i)} \triangleq Pr\left[\begin{array}{l} v = 1 \text{ where} \\ \qquad (pk, sk) \leftarrow G(1^l), \\ \qquad ((\pi_{(1)}, \pi_{(2)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1(pk, t), \\ \qquad L_{out} \leftarrow S(pk, L_{in}, \pi_{(i)}), \\ \qquad o_{(i)} \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \\ \qquad v \leftarrow A_2(\delta, o_{(i)}) \end{array}\right]$$

where $\pi_{(1)}, \pi_{(2)} \in S_n$.

The following theorem shows the equivalence of SP-CPA$_S$ and IND-CPA$_S$. The proof is similar to the proof of the equivalence between SS-CPA and IND-CPA [17] and is shown in Appendix A.

Theorem 3 *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides SP-CPA$_S$ if and only if it provides IND-CPA$_S$.*

(Chosen transcript attack) In this attack, in addition to the experiment in chosen permutation attacks, the adversary also has access to an oracle O_T, that takes a transcript of a shuffle execution and returns the corresponding permutation if the transcript is valid, and an error symbol, otherwise. The adversary is adaptive and queries are chosen by taking the results of all previous queries into account. We assume that in step 4, the adversary can not use the transcript in the actual challenge as a query to O_T.

We note that for shuffles without proof systems, i.e. not providing verifiability, the adversary with access to O_T can always learn the permutation. This is because the shuffle transcript consists of an input ciphertext list and an output ciphertext list and the adversary can use re-encryption to generate another input ciphertext list and another output ciphertext list that he can present to O_T and obtain the permutation. For verifiable shuffles, the attack can be prevented by using the verifiability proof systems. For example, informally, by adding proofs of knowledge to the proof system, construction of new valid transcripts from old ones could be prevented. Definitions of SP-CTA$_S$ and IND-CTA$_S$ are given as follows.

Definition 9 *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to provide* semantic privacy under chosen transcript attacks (SP-CTA$_S$) *if for every pair of PPT oracle machines, A_1 and A_2, there exists a pair of PPT algorithms, A'_1 and A'_2, such that the following two conditions hold:*

1. *For every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$Pr\left[\begin{array}{l} v = f_n(\pi) \text{ where} \\ \qquad (pk, sk) \leftarrow G(1^l), \\ \qquad ((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1^{O_T}(pk, t), \\ \qquad L_{out} \leftarrow S(pk, L_{in}, \pi) \text{ where } \pi \leftarrow \Pi_n(U_{poly(l)}), \\ \qquad o \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, \\ \qquad C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi)), \\ \qquad v \leftarrow A_2^{O_T}(\delta, o) \\ \qquad (A_2 \text{ must not send } (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}) \\ \qquad \text{to } O_T) \end{array}\right]$$

$$< Pr\left[\begin{array}{l} v = f_n(\pi) \text{ where} \\ \qquad ((\Pi_n, h_n, f_n), \delta) \leftarrow A'_1(1^l, t), \\ \qquad \pi \leftarrow \Pi_n(U_{poly(l)}), \\ \qquad v \leftarrow A'_2(\delta, h_n(\pi)) \end{array}\right] + l^{-\alpha}$$

2. *For every l and t above, the parts (Π_n, h_n, f_n) generated from $A_1^{O_T}(pk, t)$ and $A'_1(1^l, t)$ are identically distributed.*

Definition 10 *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to provide* indistinguishability under chosen transcript attacks (IND-CTA$_S$) *if for every pair of PPT oracle machines, A_1 and A_2, for every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$|p_{l,t}^{(1)} - p_{l,t}^{(2)}| < l^{-\alpha}$$

where

$$p_{l,t}^{(i)} \triangleq Pr\left[\begin{array}{l} v = 1 \text{ where} \\ \quad (pk, sk) \leftarrow G(1^l), \\ \quad ((\pi_{(1)}, \pi_{(2)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1^{O_T}(pk, t), \\ \quad L_{out} \leftarrow S(pk, L_{in}, \pi_{(i)}), \\ \quad o_{(i)} \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \\ \quad v \leftarrow A_2^{O_T}(\delta, o_{(i)}) \\ \quad (A_2 \text{ must not send } (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}) \\ \quad \text{to } O_T) \end{array}\right]$$

where $\pi_{(1)}, \pi_{(2)} \in S_n$.

The following theorem shows the equivalence of SP-CTA$_S$ and IND-CTA$_S$. The proof is similar to the proof of the equivalence between SS-CCA and IND-CCA [17] and is shown in Appendix A.

Theorem 4 *A verifiable shuffle* $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ *provides SP-CTA$_S$ if and only if it provides IND-CTA$_S$.*

4 Paillier-based Verifiable Shuffle scheme

4.1 Description

The verifiable shuffle scheme is a tuple $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$, where the public-key encryption scheme with a re-encryption algorithm $\mathcal{RP}$ is the Paillier scheme. The public key is $pk = N$, where $N = pq$ with primes p and q, and the secret key is $sk = \lambda$, where $\lambda = lcm(p-1, q-1)$. The algorithm S takes pk, a list of Paillier ciphertexts $g_1, ..., g_n \in \mathbb{Z}_{N^2}^*$ and a permutation π and outputs another list of Paillier ciphertexts $g'_1, ..., g'_n \in \mathbb{Z}_{N^2}^*$. The proof system $(\mathcal{P}, \mathcal{V})$, which is described below, must prove the existence of a permutation π and $r_1, ..., r_n \in \mathbb{Z}_N^*$ such that $g'_i = r_i^N g_{\pi^{-1}(i)} \bmod N^2$, $i = 1, ..., n$.

4.1.1 Outline of the proof system The proof system is based on ideas underlying the Furukawa-Sako proof system [12]. A permutation is also represented as a permutation matrix, which is defined in Definition 1.

Representing the permutation π used in the shuffle as a permutation matrix, the shuffle's proof system, which proves the correctness of the shuffle, must show the knowledge of a permutation matrix modulo N $(A_{ij})_{n\times n}$ and $\{r_i | i = 1, ..., n\}$ satisfying the following relationship between input and output items:

$$g'_i = r_i^N \prod_{j=1}^{n} g_j^{A_{ji}}, \ i = 1, ..., n \tag{2}$$

Theorem 5 states conditions to achieve a permutation matrix modulo N. Then the proof system needs to prove the knowledge of a matrix $(A_{ij})_{n\times n}$

and $\{r_i | i = 1, ..., n\}$ satisfying equation (2) and the conditions on the matrix, as stated in Theorem 5. Theorem 5 is similar to Theorem 2 for a permutation matrix modulo prime in the Furukawa-Sako scheme. As Theorem 5 is for a permutation matrix modulo composite N, it has one more condition "$gcd(A_{ij}, N)$ is different from p and q". The theorem's proof is shown in Appendix B.

Theorem 5 *A matrix $(A_{ij})_{n\times n}$ is a permutation matrix modulo N, where $N = pq$ with primes p and q, if for all i, j and k, $gcd(A_{ij}, N)$ is different from p and q and both of the following equations hold:*

$$\sum_{l=1}^{n} A_{li}A_{lj} = \begin{cases} 1 \ mod\ N \ if\ i = j \\ 0 \ mod\ N \ otherwise \end{cases} \quad (3)$$

$$\sum_{l=1}^{n} A_{li}A_{lj}A_{lk} = \begin{cases} 1 \ mod\ N \ if\ i = j = k \\ 0 \ mod\ N \ otherwise \end{cases} \quad (4)$$

In the proof system, based on the CCR assumption, it is computationally difficult for the prover to compute p and q. Hence, for any matrix $(A_{ij})_{n\times n}$ the prover can generate, "$gcd(A_{ij}, N)$ is different from p and q". Therefore, based on Theorem 5, the proof system needs to prove the following statements:

- Given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as equation (2) using $\{r_i\}$ and a matrix that satisfies equation (3).
- Given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as equation (2) using $\{r_i\}$ and a matrix that satisfies equation (4).
- The matrix and $\{r_i\}$ in the above two statements are the same.

4.2 Proof System

The proof system $(\mathcal{P}, \mathcal{V})$ proves that the prover $\mathcal{P}$ knows permutation π and $r_1, ..., r_n \in \mathbb{Z}_N^*$ such that $g'_i = r_i^N g_{\pi^{-1}(i)}$, $i = 1, ..., n$. The input to the proof system is $N, \{g_i\}, \{g'_i\}$, $i = 1, ..., n$. Suppose there is a publicly known set, $\{\tilde{g}_i\}_{i=1}^n$, of elements uniformly generated from $\mathbb{Z}_{N^2}^*$. Let the permutation π be represented by a permutation matrix modulo N $(A_{ij})_{n\times n}$. The protocol is as follows:

1. $\mathcal{P}$ generates: $\alpha_i \leftarrow \mathbb{Z}_N$ and $\alpha, \tilde{r}_i, \tilde{\alpha}, \delta_i, \rho, \rho_i, \tau, \tau_i \leftarrow \mathbb{Z}_N^*$, $i = 1, ..., n$.
2. $\mathcal{P}$ computes:

$$\tilde{g_i}' = \tilde{r_i}^N \prod_{j=1}^{n} \tilde{g_j}^{A_{ji}}, \ i = 1, ..., n$$

$$\tilde{g}' = \tilde{\alpha}^N \prod_{j=1}^{n} \tilde{g_j}^{\alpha_j}; \ g' = \alpha^N \prod_{j=1}^{n} g_j^{\alpha_j}$$

$$\dot{t_i} = \delta_i^N(1 + N\sum_{j=1}^n 3\alpha_j A_{ji});\ \dot{v_i} = \rho_i^N(1 + N\sum_{j=1}^n 3\alpha_j^2 A_{ji}),\ i = 1, ..., n$$

$$\dot{w_i} = \tau_i^N(1 + N\sum_{j=1}^n 2\alpha_j A_{ji}),\ i = 1, ..., n$$

$$\dot{v} = \rho^N(1 + N\sum_{j=1}^n \alpha_j^3);\ \dot{w} = \tau^N(1 + N\sum_{j=1}^n \alpha_j^2)$$

3. $\mathcal{P} \longrightarrow \mathcal{V}$: $\{\tilde{g_i}'\}, \tilde{g}', g', \{\dot{t_i}\}, \{\dot{v_i}\}, \dot{v}, \{\dot{w_i}\}, \dot{w},\ i = 1, ..., n$
4. $\mathcal{P} \longleftarrow \mathcal{V}$: challenge $\{c_i\}$, $c_i \leftarrow \mathbb{Z}_N$, $i = 1, ..., n$
5. $\mathcal{P} \longrightarrow \mathcal{V}$: the following responses

$$s_i = \sum_{j=1}^n A_{ij}c_j + \alpha_i \text{ mod N},\ i = 1, ..., n \quad (5)$$

$$\tilde{s} = \tilde{\alpha}\prod_{i=1}^n \tilde{r_i}^{c_i}\tilde{g_i}^{d_i} \text{ mod } N$$

$$s = \alpha\prod_{i=1}^n r_i^{c_i}g_i^{d_i} \text{ mod } N;\ u = \rho\prod_{i=1}^n \rho_i^{c_i}\delta_i^{c_i^2} \text{ mod } N;\ v = \tau\prod_{i=1}^n \tau_i^{c_i} \text{ mod } N$$

where $d_i = (\sum_{j=1}^n A_{ij}c_j + \alpha_i - s_i)/N$, $i = 1, ..., n$ (so d_i can only be 0 or 1)
6. $\mathcal{V}$ verifies:

$$\tilde{s}^N\prod_{j=1}^n \tilde{g_j}^{s_j} = \tilde{g}'\prod_{j=1}^n \tilde{g_j}'^{c_j} \quad (6)$$

$$s^N\prod_{j=1}^n g_j^{s_j} = g'\prod_{j=1}^n g_j'^{c_j} \quad (7)$$

$$u^N(1 + N\sum_{j=1}^n (s_j^3 - c_j^3)) = \dot{v}\prod_{j=1}^n \dot{v_j}^{c_j}\dot{t_j}^{c_j^2} \quad (8)$$

$$v^N(1 + N\sum_{j=1}^n (s_j^2 - c_j^2)) = \dot{w}\prod_{j=1}^n \dot{w_j}^{c_j} \quad (9)$$

4.3 Security

Theorem 6 and Theorem 7 show that the proposed shuffle provides Verifiability under the Computational Composite Residuosity (CCR) assumption, and SP-CPA$_S$ under the Decisional Composite Residuosity (DCR) assumption. Proofs of these theorems are shown in Appendix B.

Theorem 6 *The shuffle achieves Verifiability if the CCR assumption holds.*

Theorem 7 *The shuffle achieves SP-CPA$_S$ if the DCR assumption holds.*

4.4 Comparison

Most of other shuffle schemes rely on the Discrete Log and Decisional Diffie-Hellman assumptions and often lack security proofs, whereas our shuffle scheme relies on the Computational Composite Residuosity and Decisional Composite Residuosity assumptions with provable security. The proposed proof system has the round efficiency of the Furukawa-Sako protocol (3 rounds) whereas both Neff's system and Groth's system require 7 rounds. Compared to the Furukawa-Sako and Groth proof systems, our proposed proof system has a more efficient set-up. In both those systems for El Gamal ciphertexts, a set of subgroup elements is used. Construction of these elements in general is computationally expensive [33]. Our proof system also relies on a set ($\{\tilde{g_1}, ..., \tilde{g_n}\}$) of elements in $Z^*_{N^2}$ that are just uniformly generated.

Our proof system requires $18n$ exponentiations in modulo N^2, whereas the Neff, Furukawa-Sako and Groth systems require $23n$, $18n$ and $12n$ exponentiations in a modulo prime p, respectively. However, at a comparable level of security, where the sizes of N and p are the same and the Neff, Furukawa-Sako and Groth systems have exponents whose size is k times smaller than p, an exponentiation in our system takes $4k$ times longer than an exponentiation the other systems. So the computation cost of our system is much more expensive. For the same reason, verifiable decryption of output ciphertexts of our shuffle system is more expensive and a trusted party is needed to generate keys N and λ.

5 A Robust Mix-net based on the Paillier Public-key System

5.1 Overview

The main motivation for analysing and constructing verifiable shuffles is constructing *robust mix-nets* that consist of the following polynomially bounded participants. *Users* send ciphertexts to the mix-net. Each *mix-server* (also mix-centre) is implemented as a verifiable shuffle. It takes as input a list of ciphertexts and outputs a permuted list of re-encrypted ciphertexts to the next mix server. *Decryption servers* collaboratively decrypt the list of ciphertexts output by the last mix-server. A *verifier* verifies correctness of the mix-net's operation. The verifier can be replaced by a collaboration of all servers, where each verification result is decided by a majority vote of all servers and each random challenge is jointly generated by all servers. All communication is assumed accessible by all servers.

Robustness ensures that the probability of producing correct output is close to 1. If mix-centres are implemented as verifiable shuffles, correctness of their outputs can be verified and in the case that a proof does not succeed, the corresponding mix-centre is excluded from the mix-net operation. Inputs to a mix-net must be encrypted by an NM-CCA encryption scheme [20]. Otherwise, an adversary can trace an input ciphertext *ci* by creating another input ciphertext ci' whose plaintext is related to ci's plaintext in a known manner and checking the mix-net's output for plaintexts that satisfy the relationship. An example of this attack is shown in [41] against the mix-net in [40]. It is also desirable to distribute the decryption ability, so that a minimum number of decryption servers, the *threshold*, is needed to decrypt the ciphertexts. The decryption process should also be *robust* that means the corrupted decryption servers should not be able to prevent uncorrupted ones from correctly decrypting the ciphertexts. In short, an *NM-CCA robust threshold* encryption scheme is required. Our mix-net uses the NM-CCA robust threshold version of the Paillier encryption scheme [11].

5.2 Description

(Set up) There are t mix servers, $VS_1, \cdots, VS_t$, one or more decryption servers and a verifier $\mathcal{V}$. Each mix server is implemented by as a verifiable shuffle proposed in section 4, that shuffles its input list and proves the correctness of its operation. If the proof succeeds, the output of the shuffle will be used as the input of the next shuffle. Otherwise, the next shuffle uses the input of the previous shuffles. And all verifiable shuffles of the mix servers share the same public key.

The input ciphertexts to the mix-net are encrypted by the NM-CCA robust threshold version of the Paillier encryption scheme [11]. A ciphertext encrypted using this scheme has the form (e, aux), where e is the normal Paillier ciphertext and aux allows the ciphertext to be NM-CCA-robust-threshold.

(Operations)

1. The key generation algorithm of the NM-CCA robust threshold encryption scheme generates a public key and a set of secret keys and each decryption server is given a secret key. After that, several mix sessions can be executed and each mix session can be described as follows.
2. Users send the first mix-server ciphertexts encrypted by the public key of the mix-net. An input ciphertext (e, aux) needs to pass NM-CCA test by the verifier before the sub-ciphertext e is taken to the first mix-server. Suppose $L_0 = (c_1, \cdots, c_n)$ is a list of those sub-ciphertexts taken to the first mix-server.
3. Each VS_i in turn computes a randomly permuted and re-encrypted list $L_i = (a'_{\tau_i(1)}, \cdots, a'_{\tau_i(n)})$ from $L_{i-1} = (a_1, \cdots, a_n)$, where a'_i is a re-encryption of a_i and τ_i is a secret random permutation on $\{1, \cdots, n\}$, and then outputs L_i. VS_i runs the proof system $(\mathcal{P}, \mathcal{V})$ to prove that L_i is a permutation of re-encryptions of elements in L_{i-1}.
   In case that the proof does not succeed, VS_i is excluded from the mix-net operation. If $i \neq t$, the mix-centre VS_{i+1} that receives the output of the corrupted mix-centre VS_i, will instead use VS_i's input list as its input, effectively disregarding VS_i. If $i = t$, VS_i's input list will be sent to the decryption servers.

4. The decryption servers jointly decrypt ciphertexts, which are sent from the mix-centres, in a robust way and output a list of messages $L_{out} = (m_{\phi(1)}, \cdots, m_{\phi(n)})$, where $\phi = \tau_t \cdots \tau_1$ and m_i is a plaintext of c_i.

5.3 Security

Extending our formal security model for verifiable shuffles to a formal security model for robust mix-nets would require significant efforts. First, verifiable shuffles are mainly used for one type of robust mix-nets, where each mix server behaves as a verifiable shuffle, whereas there are other types of mix-nets. There are mix-nets where each input element is a multiple-layer encryption of a message and each mix-server decrypts one layer of encryption. And there are hybrid mix-nets, which use both asymmetric and symmetric encryptions. Second, in the type of mix-nets where verifiable shuffles are applicable, there are multiple mix-servers and decryption servers. Corruption among these servers would be much more complicated than just a single malicious shuffle. A formal security model for mix-nets has been proposed [4] but no robust mix-net scheme has been proved secure in this model and proving our mix-net construction secure in this model would not be easy. Therefore, providing a formal security model for mix-nets and proving our mix-net construction secure in a formal security model remain to be our future work.

At present, we can only informally state that the mix-net achieves privacy, robustness and public verifiability. Robustness and public verifiability of the mix-net depends on verifiability of its shuffles (mix-centres). As any honest party can be the verifier, the mix-net achieves public verifiability. Privacy of the mix-net can be concluded from SP-CPA$_S$ of the shuffles.

6 Conclusion

In this paper, we proposed a formalization of verifiable shuffles and their security requirements under adaptive chosen permutation attacks and adaptive chosen transcript attacks. This provides a general framework for security assessment of shuffle systems. We proposed a verifiable shuffle system based on the Paillier public-key encryption system and proved its security in our proposed model. We also used the shuffle scheme to construct a robust mix-net.

References

1. Abe M (1998) Universally verifiable mix-net with verification work independent of the number of mix-servers. EUROCRYPT 1998, LNCS 1403, Springer-Verlag: 437-447.
2. Abe M (1999) Mix-networks on permutation networks. ASIACRYPT 1999, LNCS 1716, Springer-Verlag: 258-273.

3. Abe M, Hoshino F (2001) Remarks on Mix-Network Based on Permutation Networks. PKC 2001, LNCS, Springer-Verlag: 317-324.
4. Abe M, Imai H (2003) Flaws in Some Robust Optimistic Mix-nets. ACISP 2003, LNCS 2727, Springer-Verlag: 39-50.
5. Abe M (2004) Combining Encryption and Proof of Knowledge in the Random Oracle Model. Computer Journal 47/1.
6. Boneh D, Golle P (2002) Almost Entirely Correct Mixing with Application to Voting. ACM CCS 2002, ACM Press.
7. Brands S (1993) An efficient off-line electronic cash system based on the representation problem. CWI Technical Report CS-R9323.
8. Chaum D (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2): 84-88.
9. Choi S, Kim K (2003) Authentication and Payment Protocol Preserving Location Privacy in Mobile IP. GLOBECOM 2003.
10. Desmedt Y, Kurosawa K (2000) How to break a practical mix and design a new one. EUROCRYPT 2000, LNCS 1807, Springer-Verlag: 557-572.
11. Fouque P, Pointcheval D (2001) Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. ASIACRYPT 2001, LNCS 2248, Springer-Verlag: 351-368.
12. Furukawa J, Sako K (2001) An Efficient Scheme for Proving a Shuffle. CRYPTO 2001, LNCS 2139, Springer-Verlag: 368-389.
13. Furukawa J, Miyauchi H, Mori K, Obana S, Sako K (2002) An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling. Financial Cryptography 2002, LNCS 2357, Springer-Verlag.
14. Furukawa J (2004) Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. PKC 2004, LNCS 2947, Springer-Verlag: 319-332.
15. Gabber E, Gibbons P, Matias Y, Mayer A (1997) How to make personalized Web browsing simple, secure, and anonymous. Financial Cryptography 1997, LNCS 1318, Springer-Verlag: 17-31.
16. Goldreich O (2001) Foundations of Cryptography Basic Tools. Cambridge University Press.
17. Goldreich O (2004) Foundations of Cryptography, Basic Applications. Cambridge University Press.
18. Golle P, Zhong S, Boneh D, Jakobsson M, Juels A (2002) Optimistic Mixing for Exit-Polls. ASIACRYPT 2002, LNCS 2501, Springer-Verlag: 451-465.
19. Groth J (2003) A Verifiable Secret Shuffle of Homomorphic Encryptions. PKC 2003, LNCS 2567, Springer-Verlag:145–160.
20. Jakobsson M (1998) A practical mix. EUROCRYPT 1998, LNCS 1403, Springer-Verlag: 448-461.
21. Jakobsson M, M'Raihi D (1998) Mix-based electronic payments. SAC 1993, LNCS 1505 Springer-Verlag: 457-473.
22. Jakobsson M (1999) Flash mixing. PODC 1999, ACM Press: 83-89.
23. Jakobsson M, Juels A (1999) Millimix: Mixing in small batches. DIMACS Technical Report: 99-33.
24. Jakobsson M, Juels A (2000) Mix and match: Secure function evaluation via ciphertexts. ASIACRYPT 2000, LNCS 1976, Springer-Verlag: 162-177.
25. Jakobsson M, Juels A (2001) An Optimally Robust Hybrid Mix Network. PODC 2001, ACM Press.
26. Jakobsson M, Juels A, Rivest R (2002) Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. USENIX Security 2002.

27. Juels A (2001) Targeted advertising and privacy too. CT-RSA 2001, LNCS 2020, Springer-Verlag: 408-425.
28. Kong J, Hong X (2003) ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003): 291-302.
29. Menezes A, van Oorschot P, Vanstone S (1996) Handbook of Applied Cryptography. CRC Press.
30. Mitomo M, Kurosawa K (2000) Attack for flash mix. ASIACRYPT 2000, LNCS 1976, Springer-Verlag: 192-204.
31. Naor M, Yung M (1990) Public-Key Cryptosystems Provably Secure against Chosen Ciphertexts Attacks. STOC 1990, ACM Press: 427-437.
32. Neff A (2001) A verifiable secret shuffle and its application to e-voting. ACM CCS 2001, ACM Press: 116-125.
33. Neff A (2003) Verifiable Mixing (Shuffling) of ElGamal Pairs. Available online: http://www.votehere.org/vhti/documentation/egshuf.pdf.
34. Nguyen L, Safavi-Naini R (2003) Breaking and Mending Resilient Mix-nets. PET 2003, LNCS 2760, Springer-Verlag: 66-80.
35. Nguyen L, Safavi-Naini R, Kurosawa K (2004) Verifiable Shuffles: A Formal Model and a Paillier-based Efficient Construction with Provable Security. ACNS 2004, LNCS 3089, Springer-Verlag: 61-75.
36. Nguyen L, Safavi-Naini R (2004) An Efficient Verifiable Shuffle with Perfect Zero-knowledge Proof System. Cryptographic Algorithms and their Uses 2004: 40-56.
37. Ogata W, Kurosawa K, Sako K, Takatani K (1997) Fault tolerant anonymous channel. ICICS '97, LNCS 1334, Springer-Verlag: 440-444.
38. Ohkubo M, Abe M (2000) A length-invariant hybrid mix. ASIACRYPT 2000, LNCS 1976, Springer-Verlag: 178-191.
39. Paillier P (1999) Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. EUROCRYPT 1999, LNCS 1592, Springer-Verlag: 223-239.
40. Park C, Itoh K, Kurosawa K (1993) Efficient anonymous channel and all/nothing election scheme. EUROCRYPT 1993, LNCS 765, Springer-Verlag: 248-259.
41. Pfitzmann B (1994) Breaking an Efficient Anonymous Channel. EUROCRYPT 1994, LNCS 950, Springer-Verlag: 332-340.
42. Schnorr P, Jakobsson M (2000) Security of signed El Gamal encryption. ASIACRYPT 2000, LNCS 1976, Springer-Verlag: 73-89.
43. Shamir A (1979) How to Share a Secret. Communications of the ACM, 22: 612-613.
44. Tsiounis Y, Yung M (1998) On the security of El Gamal based encryption. PKC 1998, LNCS 1431, Springer-Verlag: 117-134.
45. Wikstrom D (2002) The security of a mix-center based on a semantically secure cryptosystem. Indocrypt 2002, LNCS 2551, Springer-Verlag: 368-381.
46. Wikstrom D (2003) Five Practical Attacks for "Optimistic Mixing for Exit-Polls". SAC 2003, LNCS 3006.

A Proofs of Theorems on Privacy Notions

A.1 Proof of Theorem 3

$\mathsf{IND\text{-}CPA}_S$ implies $\mathsf{SP\text{-}CPA}_S$. Suppose that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides IND-CPA$_S$. We will show that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides SP-CPA$_S$ by constructing, for every pair of PPT algorithms A_1 and A_2, a pair of PPT algorithms A'_1 and A'_2, such that the two conditions stated in Definition 7 hold. Specifically, A'_1 and A'_2 are constructed as follows:

1. $A'_1(1^l, t)$ returns $((\Pi_n, h_n, f_n), \delta')$, where $((\Pi_n, h_n, f_n), \delta')$ is generated as follows. First, A'_1 generates an instance of the shuffle scheme by letting $(pk, sk) \leftarrow G(1^l)$. Next, A'_1 invokes A_1 and gets $((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1(pk, t)$. Finally, A'_1 sets $\delta' = (pk, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \delta)$.
2. On input $((pk, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \delta), 1^n, \gamma)$ where $\gamma = h_n(\pi)$ for some permutation $\pi \in S_n$, A'_2 generates output as follows. It selects some fixed permutation $\pi_0 \in S_n$, simulates the shuffle algorithm S by generating a list L'_{out} of re-encrypted ciphertexts in L_{in} permuted by π_0. It then simulates an execution of the proof system $(\mathcal{P}, \mathcal{V})$ on input (pk, L_{in}, L'_{out}) and gets $View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L'_{out})$. Let $o_0 \leftarrow (L'_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L'_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \gamma)$, A'_2 outputs the value returned by $A_2^{O_S, O_E}(\delta, o_0)$.

Since A'_1 merely simulates the generation of a key pair and the actions of A_1 with respect to such a pair, the equal distribution condition (i.e., Item 2 in Definition 7) holds. Using the (corresponding) IND-CPA$_S$ property, we show that the distributions (δ, o_0) and (δ, o) are computationally indistinguishable, where $(pk, sk) \leftarrow G(1^l)$; $((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1(pk, t)$; $\pi \leftarrow \Pi_n(U_{poly(l)})$; $L_{out} \leftarrow S(pk, L_{in}, \pi)$; $o \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi))$; $L'_{out} \leftarrow S(pk, L_{in}, \pi_0)$ and $o_0 \leftarrow (L'_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L'_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi))$.

– Details: Suppose that given (Π_n, h_n, f_n) generated by $A_1(pk, t)$, there exists a PPT algorithm to distinguish between (δ, o_0) and (δ, o), where o and o_0 are generated as above. Then we obtain a distinguisher as in Definition 8 as follows. The first part of the distinguisher invokes A_1 and obtains $((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1(pk, t)$. It sets $\pi_{(1)} \leftarrow \Pi_n(U_{poly(l)})$ and $\pi_{(2)} = \pi_0$, and outputs $((\pi_{(1)}, \pi_{(2)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), (\delta, h_n(\pi_{(1)})))$. That is, $(\pi_{(1)}, \pi_{(2)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}})$ is the challenge template, and a challenge $o_{(i)}$ is generated as in Definition 8, where i is either 1 or 2. The second part of the new distinguisher, gets as input the challenge $o_{(i)}$ and the state generated by the first part $(\delta, h_n(\pi_{(1)}))$, and invokes the distinguisher of the contradiction hypothesis with input

$(\delta, (o_{(i)}, h_n(\pi_{(1)})))$. Thus, the new distinguisher violates the condition in Definition 8, in contradiction to the hypothesis that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides IND-CPA$_S$.

It follows that IND-CPA$_S$ (as in Definition 8) implies SP-CPA$_S$ (as in Definition 7).

SP-CPA$_S$ implies IND-CPA$_S$. We now turn to the opposite direction. Suppose that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides SP-CPA$_S$ but does not provide IND-CPA$_S$. Consider the challenge template $(\pi_{(1)}, \pi_{(2)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}})$ produced by the distinguishing adversary and $o_{(1)}$ and $o_{(2)}$ are generated as in Definition 8. We construct a SP-CPA$_S$ adversary by generating a corresponding challenge template $(\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}})$ such that.

– Π_n uniformly outputs either $\pi_{(1)}$ or $\pi_{(2)}$.
– The function f_n satisfies $f_n(\pi_{(1)}) = 1$ and $f_n(\pi_{(2)}) = 0$.
– The function h_n is defined arbitrarily subject to $h_n(\pi_{(1)}) = h_n(\pi_{(2)})$.

We can see that the SP-CPA$_S$ adversary has a noticeable advantage in guessing $f_n(\Pi_n(U_{poly(l)}))$ (by using the distinguishing gap between $o_{(1)}$ and $o_{(2)}$), whereas no algorithm that only gets $h_n(\Pi_n(U_{poly(l)}))$ can have any advantage in such a guess. We derive a contradiction to the hypothesis that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ provides SP-CPA$_S$ as in Definition 7, and the theorem follows.

A.2 Proof of Theorem 4

This theorem can be proved similar to the proof of Theorem 3. In order to show that IND-CTA$_S$ implies SP-CTA$_S$, given an adversary (A_1, A_2) we construct the following matching algorithm (A_1', A_2'):

1. $A_1'(1^l, t)$ returns $((\Pi_n, h_n, f_n), \delta')$, where $((\Pi_n, h_n, f_n), \delta')$ is generated as follows. First, A_1' generates an instance of the shuffle scheme by letting $(pk, sk) \leftarrow G(1^l)$. Next, A_1' invokes A_1, while simulating the oracle O_T (as A_1' knows (pk, sk)), and gets $((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1^{O_T}(pk, t)$. Finally, A_1' sets $\delta' = (pk, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \delta)$.
2. On input $((pk, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \delta), 1^n, \gamma)$ where $\gamma = h_n(\pi)$ for some permutation $\pi \in S_n$, A_2' generates output as follows. It selects some fixed permutation $\pi_0 \in S_n$, simulates the shuffle algorithm S by generating a list L_{out}' of re-encrypted ciphertexts in L_{in} permuted by π_0. It then simulates an execution of the proof system $(\mathcal{P}, \mathcal{V})$ on input (pk, L_{in}, L_{out}') and gets $View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}')$. Let $o_0 \leftarrow (L_{out}', View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}'), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, \gamma)$, A_2' outputs the value returned by $A_2^{O_T}(\delta, o_0)$. The generated key pair (pk, sk) allows A_2' to simulate the oracle O_T.

Again, since A'_1 merely simulates the generation of a key pair and the actions of A_1 with respect to such a pair, the equal distribution condition (i.e. the second item in Definition 9) holds. Using the (corresponding) IND-CTA$_S$ property, we show that (even in the presence of the oracle O_T) the distributions (δ, o_0) and (δ, o) are indistinguishable, where $(pk, sk) \leftarrow G(1^l)$; $((\Pi_n, h_n, f_n, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}), \delta) \leftarrow A_1^{O_T}(pk, t)$; $\pi \leftarrow \Pi_n(U_{poly(l)})$; $L_{out} \leftarrow S(pk, L_{in}, \pi)$; $o \leftarrow (L_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi))$; $L'_{out} \leftarrow S(pk, L_{in}, \pi_0)$ and $o_0 \leftarrow (L'_{out}, View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L'_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, h_n(\pi))$. The main thing to notice is that the oracle queries made by a possible distinguisher of the above distributions can be handled by a distinguisher of transcripts (as in Definition 10), by passing these queries to its own oracle. It follows that IND-CTA$_S$ (as in Definition 10) implies SP-CTA$_S$ (as in Definition 9).

We now turn to the opposite direction. Here the construction of a challenge template (as in Definition 9) is exactly as the corresponding construction in the proof of Theorem 3. Again, the thing to notice is that the oracle queries made by a possible distinguisher of transcripts (as in Definition 10) can be handled by the SP-CTA$_S$ adversary, by passing these queries to its own oracle. We derive a contradiction to the hypothesis that $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ satisfies Definition 9, and the theorem follows.

B Security Proofs for the Paillier-based Verifiable Shuffle scheme

B.1 Proof of Theorem 5

Suppose a matrix (A_{ij}) satisfies equations (3) and (4) and for all i and j, $gcd(A_{ij}, N)$ is different from p and q. As (A_{ij}) satisfies equations (3) and (4), based on Theorem 2, (A_{ij}) is a permutation matrix mod p and also a permutation matrix mod q. Thus, there exists a permutation π such that for all i and j:

$$A_{ij} = \begin{cases} 1 \bmod q \text{ if } \pi(i) = j \\ 0 \bmod q \text{ otherwise} \end{cases}$$

We now show that for all i and j, if $A_{ij} = 1$ (or 0, respectively) mod q, then $A_{ij} = 1$ (or 0, respectively) mod N.

Suppose there exist i' and j' such that $A_{i'j'} = 1 \bmod q$ but $A_{i'j'} \neq 1 \bmod N$. As (A_{ij}) is a permutation matrix mod p, $A_{i'j'} = 0$ or $1 \bmod p$. But $A_{i'j'} = 1 \bmod q$ and $A_{i'j'} \neq 1 \bmod N$, so $A_{i'j'} = 0 \bmod p$. That means $gcd(A_{i'j'}, N) = p$, which contradicts the condition "$gcd(A_{ij}, N)$ is different from p and q".

Suppose there exist i' and j' such that $A_{i'j'} = 0 \bmod q$ but $A_{i'j'} \neq 0 \bmod N$. As (A_{ij}) is a permutation matrix mod p, $A_{i'j'} = 0$ or $1 \bmod p$. But $A_{i'j'} = 0 \bmod q$ and $A_{i'j'} \neq 0 \bmod N$, so $A_{i'j'} = 1 \bmod p$. That means $gcd(A_{i'j'}, N) = q$, which contradicts the condition "$gcd(A_{ij}, N)$ is different from p and q".

Therefore, for all i and j, if $A_{ij} = 1$ (or 0, respectively) mod q, then $A_{ij} = 1$ (or 0, respectively) mod N. That means $(A_{ij})_{n\times n}$ is a permutation matrix modulo N.

B.2 Proof of Theorem 6 for Verifiability

To prove Theorem 6, we need Theorem 5 and Theorem 8, which are not required in the proofs for the Furukawa-Sako scheme. We observe that knowledge of $A_{i'j'}$ satisfying $gcd(A_{i'j'}, N) = p$ or q allows factorization of N and reveals the secret key λ. So, from Theorem 5, the objective of the proof system can be re-stated as follows.

The common input to the proof system includes $N, \{g_i\}, \{g'_i\}, i = 1, ..., n$. The auxiliary input to the prover $\mathcal{P}$ includes permutation π and $r_1, ..., r_n \in \mathbb{Z}_N^*$ satisfying $g'_i = r_i^N g_{\pi^{-1}(i)}$ and does not include the secret key $sk = \lambda$. The proof system $(\mathcal{P}, \mathcal{V})$ proves $\mathcal{P}$ knows a matrix $(A_{ij})_{n\times n}$ and $r_1, ..., r_n \in \mathbb{Z}_N^*$ such that equations (3) and (4) hold and

$$g'_i = r_i^N \prod_{j=1}^{n} g_j^{A_{ji}},\ i = 1, ..., n \tag{10}$$

Based on Definition 5 of Verifiability, Theorem 6 can be concluded from Theorem 9 and Theorem 10, which state the Completeness and Soundness properties of the proof system. The role of Theorem 8 will be explained in the proof of Theorem 10. Theorems 8, 9 and 10 are presented and proved as follows.

Theorem 8 *For a set of vectors* $\mathbf{S}$*, let* $\langle\mathbf{S}\rangle_k$ *denote the vector space spanned by* $\mathbf{S}$ *over* $\mathbb{Z}_k$ *(so the coordinates of a vector in* $\langle\mathbf{S}\rangle_k$ *are in* $\mathbb{Z}_k$*). And let* $|\mathbf{S}|$ *denote the number of elements in* $\mathbf{S}$*. Consider a set of vectors* $T_n = \{(1, c_1, ..., c_n) \mid (c_1, ..., c_n \in \mathbb{Z}_N) \wedge (\nexists Q_n \subseteq T_n : |Q_n| = n + 1 \wedge \langle Q_n\rangle_p = \mathbb{Z}_p^{n+1} \wedge \langle Q_n\rangle_q = \mathbb{Z}_q^{n+1})\}$ *(that means* T_n *is the set of vectors* $(1, c_1, ..., c_n)$*, where* $c_1, ..., c_n \in \mathbb{Z}_N$ *and there does not exist any subset* $Q_n \subseteq T_n$ *of size* $n + 1$ *such that* Q_n *spans* $\mathbb{Z}_p^{n+1}$ *and* $\mathbb{Z}_q^{n+1}$*). Then* $|T_n| \leq (p + q)N^{n-1}$*.*

Proof This is proved by induction.

- $n = 1$: Consider a set of vectors $T_1 \subseteq \{(1, c)|c \in \mathbb{Z}_N\}$ satisfying $|T_1| > (p+q)$; and a vector $(1, c_1) \in T_1$. Consider a set $R_1 = \{(1, c_1+kp \bmod N) |k \in \mathbb{Z}_q\} \cup \{(1, c_1 + kq \bmod N)|k \in \mathbb{Z}_p\}$. As $|R_1| = p + q - 1$, there exists $c'_1 \in \mathbb{Z}_N$ such that $(1, c'_1) \in T_1$ but $(1, c'_1) \notin R_1$. Then $Q_1 = \{(1, c_1), (1, c'_1)\}$ satisfies $|Q_1| = 2 \wedge \langle Q_1\rangle_p = \mathbb{Z}_p^2 \wedge \langle Q_1\rangle_q = \mathbb{Z}_q^2$.
- Suppose the theorem holds for n. We prove it is also true for $n + 1$. Let a set $T_{n+1} = \{(1, c_1, ..., c_{n+1})|(c_1, ..., c_{n+1} \in \mathbb{Z}_N) \wedge (\nexists Q_{n+1} \subseteq T_{n+1} : |Q_{n+1}| = n + 2 \wedge \langle Q_{n+1}\rangle_p = \mathbb{Z}_p^{n+2} \wedge \langle Q_{n+1}\rangle_q = \mathbb{Z}_q^{n+2})\}$. Consider $T'_n = \{(1, c_1, ..., c_n)|\exists c_{n+1} \in \mathbb{Z}_N : (1, c_1, ..., c_n,\ c_{n+1}) \in T_{n+1}\}$, there are two possibilities:

1. If $\nexists Q'_n \subseteq T'_n : |Q'_n| = n+1 \wedge \langle Q'_n \rangle_p = \mathbb{Z}_p^{n+1} \wedge \langle Q'_n \rangle_q = \mathbb{Z}_q^{n+1}$, then $|T'_n| \leq (p+q)N^{n-1}$, as the theorem holds for n. So $|T_{n+1}| \leq |T'_n|N \leq (p+q)N^n$.
2. If $\exists Q'_n \subseteq T'_n : |Q'_n| = n+1 \wedge \langle Q'_n \rangle_p = \mathbb{Z}_p^{n+1} \wedge \langle Q'_n \rangle_q = \mathbb{Z}_q^{n+1}$, select a set T of $n+1$ vectors $(1, c_{i1}, ..., c_{i(n+1)}) \in T_{n+1}$, $i = 1, ..., n+1$ such that $Q'_n = \{(1, c_{i1}, ..., c_{in})\}$
   Let $d = det \begin{pmatrix} 1 & c_{11} & ... & c_{1n} \\ .. & .. & .. & .. \\ 1 & c_{(n+1)1} & ... & c_{(n+1)n} \end{pmatrix}$ mod N, then $gcd(d, N) = 1$, so d^{-1} mod N exists.
   For each vector $x = (1, x_1, ..., x_{n+1}) \in T_{n+1}$ (including those in T), let
   $$d_x = det \begin{pmatrix} 1 & c_{11} & ... & c_{1(n+1)} \\ .. & .. & .. & .. \\ 1 & c_{(n+1)1} & ... & c_{(n+1)(n+1)} \\ 1 & x_1 & ... & x_{n+1} \end{pmatrix} = dx_{n+1} - F(x_1, ..., x_n) \bmod N$$
   for some function F. The conditions of T_{n+1} lead to either $d_x = 0$ mod p or $d_x = 0$ mod q.
   Suppose $d_x = 0$ mod p, then $x_{n+1} = d^{-1}F(x_1, ..., x_n)$ mod p, so the number of possible vectors $x = (1, x_1, ..., x_{n+1})$ is no more than qN^n. Similar for the case $d_x = 0$ mod q, the number of possible vectors $x = (1, x_1, ..., x_{n+1})$ is no more than pN^n. So $|T_{n+1}| \leq (p+q)N^n$.

Before presenting Theorems 9 and 10, it is necessary to recall properties of the Paillier public-key system.

Properties of the Paillier encryption scheme: The security proofs need the following properties of the Paillier cryptosystem. For $w \in \mathbb{Z}^*_{N^2}$, we call N^{th} *residuosity class* of w the unique integer $x \in \mathbb{Z}_N$ for which there exists $y \in \mathbb{Z}^*_N$ such that $w = y^N(1 + xN)$ [39]. The class of w is denoted $c(w)$. Note that the plaintext of a Paillier ciphertext is the class of that ciphertext. We have the following properties.

- $c(w) = 0 \Leftrightarrow w$ is a N^{th} residue mod N^2 (N^{th} residues mod N^2 are defined in the definition of the DCR assumption).
- $c(w_1w_2) = c(w_1) + c(w_2)$ mod N
- $c(w_1) = c(w_2) \Leftrightarrow \exists r \in \mathbb{Z}^*_N : w_1 = w_2r^N$

Theorem 9 *(Completeness) If $\mathcal{P}$ knows a matrix (A_{ij}) and $r_1, ..., r_n \in \mathbb{Z}^*_N$ satisfying (3), (4) and (10), and performs correctly in the protocol, then $\mathcal{V}$ always accepts.*

Proof Suppose $\mathcal{P}$ knows a matrix (A_{ij}) and $r_1, ..., r_n \in \mathbb{Z}^*_N$ satisfying equations (3), (4) and (10); and $\{\tilde{g_i}'\}, \tilde{g}', g', \{\dot{t_i}\}, \{\dot{v_i}\}, \dot{v}, \{\dot{w_i}\}, \dot{w}, \{c_i\}, \{s_i\}, \tilde{s}, s, u, v$ for $i = 1, ..., n$ are generated as specified in the protocol. Then the verifier outputs accept, as the following equations hold.

- $\tilde{s}^N \prod_{j=1}^n \tilde{g_j}^{s_j} = (\tilde{\alpha} \prod_{i=1}^n \tilde{r_i}^{c_i} \tilde{g_i}^{d_i})^N \prod_{j=1}^n \tilde{g_j}^{\sum_{i=1}^n A_{ji}c_i + \alpha_j}$
  $= (\tilde{\alpha}^N \prod_{j=1}^n \tilde{g_j}^{\alpha_j}) \prod_{j=1}^n (\tilde{r_j}^N \prod_{i=1}^n \tilde{g_i}^{A_{ij}})^{c_j} = \tilde{g}' \prod_{j=1}^n \tilde{g_j}'^{c_j}$

- $s^N \prod_{j=1}^n g_j^{s_j} = (\alpha \prod_{i=1}^n r_i^{c_i} g_i^{d_i})^N \prod_{j=1}^n g_j^{\sum_{i=1}^n A_{ji}c_i + \alpha_j}$
  $= (\alpha^N \prod_{j=1}^n g_j^{\alpha_j}) \prod_{j=1}^n (r_j^N \prod_{i=1}^n g_i^{A_{ij}})^{c_j} = g' \prod_{j=1}^n g_j'^{c_j}$.
- $u^N(1+N\sum_{j=1}^n(s_j^3 - c_j^3)) = (\rho \prod_{i=1}^n \rho_i^{c_i} \delta_i^{c_i^2})^N (1+N\sum_{j=1}^n((\sum_{i=1}^n A_{ji}c_i + \alpha_j)^3 - c_j^3)) = \rho^N(1+N\sum_{j=1}^n \alpha_j^3) \prod_{j=1}^n(\rho_j^N(1+N\sum_{i=1}^n 3\alpha_i^2 A_{ij}))^{c_j}$
  $\prod_{j=1}^n(\delta_j^N(1+N\sum_{i=1}^n 3\alpha_i A_{ij}))^{c_j^2} = \dot{v} \prod_{j=1}^n \dot{v_j}^{c_j} \dot{t_j}^{c_j^2}$.
- $v^N(1+N\sum_{j=1}^n(s_j^2 - c_j^2)) = (\tau \prod_{i=1}^n \tau_i^{c_i})^N (1+N\sum_{j=1}^n((\sum_{i=1}^n A_{ji}c_i + \alpha_j)^2 - c_j^2)) = \tau^N(1+N\sum_{j=1}^n \alpha_j^2) \prod_{j=1}^n(\tau_j^N(1+N\sum_{i=1}^n 2\alpha_i A_{ij}))^{c_j} = \dot{w} \prod_{j=1}^n \dot{w_j}^{c_j}$.

Theorem 10 *(Soundness) Under the CCR assumption, if $\mathcal{V}$ accepts with nonnegligible probability, then $\mathcal{P}$ knows a matrix (A_{ij}) satisfying equations (3), (4) and the following equation with overwhelming probability.*

$$c(g_i') = c(\prod_{j=1}^n g_j^{A_{ji}}),\ i = 1, ..., n \tag{11}$$

Proof This is proved using Theorem 8 and the following lemmas.

- Lemma 1 shows that if the CCR assumption holds, then it is computationally difficult for $\mathcal{P}$ to obtain $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.
- Lemma 2 and Theorem 8 show that if $\mathcal{V}$ accepts with non-negligible probability, then $\mathcal{P}$ knows a matrix $\{A_{ij}\}$ satisfying
  $$c(\tilde{g_i}') = c(\prod_{j=1}^n \tilde{g_j}^{A_{ji}}),\ i = 1, ..., n$$
- Lemmas 3 and 4 show that either this matrix $\{A_{ij}\}$ satisfies equation (3) or $\mathcal{P}$ can compute $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.
- Lemmas 3 and 5 show that either this matrix $\{A_{ij}\}$ satisfies equation (4) or $\mathcal{P}$ can compute $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.
- Lemma 6 shows that this matrix $\{A_{ij}\}$ satisfies equation 11.

So these lemmas and Theorem 8 show that if the CCR assumption holds and $\mathcal{V}$ accepts with non-negligible probability, then $\mathcal{P}$ knows a matrix (A_{ij}) satisfying equations (3), (4) and (11) with overwhelming probability.

Lemma 1 *If the CCR assumption holds, then it is computationally difficult for $\mathcal{P}$ to obtain $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not* 0.

Proof Suppose, with non-negligible probability, $\mathcal{P}$ can compute $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0. We construct a PPT algorithm $\mathcal{A}$ that can break the CCR assumption as follows. Suppose $\mathcal{A}$ is given a product N of two $l/2$-bit primes and $z \leftarrow \mathbb{Z}_{N^2}^*$, we show that $\mathcal{A}$ can compute $x \in \mathbb{Z}_N$ with non-negligible probability such that there exists $r \in \mathbb{Z}_N^*$ satisfying $r^N(1+xN) = z \bmod N^2$, or in other words, $c(z) = x$.

$\mathcal{A}$ simulates an instance of the verifiable shuffle scheme, where the public key is $pk = N$. $\mathcal{A}$ then generates $\{\tilde{g_i}\}_{i=1}^n$ as follows. $\mathcal{A}$ chooses $i_0 \leftarrow \{1,...,n\}$ and generates $r'_i \leftarrow \mathbb{Z}_N^*$, $x'_i \leftarrow \mathbb{Z}_N$ for $i = 1,...,n$. Then for $i = 1,...,n$ and $i \neq i_0$, $\mathcal{A}$ computes $\tilde{g_i} = r_i'^N(1+x'_iN)$. $\mathcal{A}$ computes $\tilde{g_{i_0}} = r_{i_0}'^N(1+x'_{i_0}N)z$. The set $\{\tilde{g_i}\}_{i=1}^n$ is then given to $\mathcal{P}$. As the distribution of $\{\tilde{g_i}\}_{i=1}^n$ is the same as the distribution of a set of n elements uniformly generated from $\mathbb{Z}_{N^2}^*$, $\mathcal{P}$ can compute $\{a_i\} \subset \mathbb{Z}_N$ such that $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0, with non-negligible probability ϵ. As i_0 is chosen uniformly from $\{1,...,n\}$, the probability that $a_{i_0} \neq 0$ is ϵ/n.

As $c(\prod_{i=1}^n \tilde{g_i}^{a_i}) = 0$, we have $a_{i_0}x + \sum_{i=1}^n a_i x'_i = 0 \bmod N$. So $\mathcal{A}$ can compute $x = -a_{i_0}^{-1}\sum_{i=1}^n a_i x'_i \bmod N$ and breaks the CCR assumption.

Lemma 2 *If $\mathcal{V}$ accepts with non-negligible probability, then $\mathcal{P}$ knows a matrix $\{A_{ij}\}$ and $\{\alpha_i\}$ satisfying*

$$c(\tilde{g_i}') = c(\prod_{j=1}^n \tilde{g_j}^{A_{ji}}),\ i = 1,...,n \tag{12}$$

$$c(\tilde{g}') = c(\prod_{j=1}^n \tilde{g_j}^{\alpha_j}) \tag{13}$$

Proof Consider the set T_n of all vectors $(1, c_1, ..., c_n)$ constructed from all the challenges $\{c_i\}$ generated by $\mathcal{V}$, for which $\mathcal{P}$ can compute responses $\{s_i\}$ such that $\mathcal{V}$ accepts. As $\mathcal{V}$ accepts with non-negligible probability, we have $|T_n|/|\mathbb{Z}_N^n|$ is non-negligible, that means $|T_n| > (p+q)N^{n-1}$. Based on Theorem 8, $\mathcal{P}$ can find a set Q_n of $n+1$ vectors so that $Q_n \subseteq T_n$, $< Q_n >_p = \mathbb{Z}_p^{n+1}$ and $< Q_n >_q = \mathbb{Z}_q^{n+1}$. Then $\mathcal{P}$ can obtain $\{A_{ij}\}_{i,j=1}^n \subset \mathbb{Z}_N$ and $\{\alpha_i\}_{i=1}^n \subset \mathbb{Z}_N$, such that for every $(1, c_1, ..., c_n) \in Q_n$ and corresponding response $\{s_i\}$, we have:

$$s_i = \sum_{j=1}^n A_{ij}c_j + \alpha_i \bmod N,\ i = 1,...,n$$

Replace these values of s_i into equation (6), we have:

$$c(\prod_{i=1}^n \tilde{g_i}^{\sum_{j=1}^n A_{ij}c_j + \alpha_i}) = c(\tilde{g}' \prod_{j=1}^n \tilde{g_j}'^{c_j}) \bmod N,\ \forall (1, c_1, ..., c_n) \in Q_n$$

$$\Rightarrow c(\frac{\prod_{j=1}^n \tilde{g_j}^{\alpha_j}}{\tilde{g}'}) + \sum_{i=1}^n c_i c(\frac{\prod_{j=1}^n \tilde{g_j}^{A_{ji}}}{\tilde{g_i}'}) = 0 \bmod N,\ \forall (1, c_1, ..., c_n) \in Q_n$$

So equations (12) and (13) hold.

Lemma 3 *Assume that $\mathcal{P}$ knows $\{A_{ij}\}$ and $\{\alpha_i\}$ satisfying equations (12) and (13). If $\mathcal{P}$ knows $\{s_i\}$ and $\tilde{s}$ which satisfy equation (6), then either equations (5) hold, or $\mathcal{P}$ can generate $\{a_i\}$ with overwhelming probability such that $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.*

Proof Consider the case when $\mathcal{P}$ knows $\{s_i\}$ and $\tilde{s}$ satisfying equation (6); and there exists $i_0 \in \{1, ..., n\}$ satisfying $s_{i_0} \neq \sum_{j=1}^{n} A_{i_0 j} c_j + \alpha_{i_0} \bmod N$. Then $\mathcal{P}$ can find $a_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i - s_i \bmod N$, $i = 1, ..., n$ such that $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ and $a_{i_0} \neq 0$. This can be shown by replacing the values of $c(\tilde{g_i}')$ and $c(\tilde{g}')$ from equations (12) and (13) into equation $c(\prod_{j=1}^{n} \tilde{g_j}^{s_j}) = c(\tilde{g}') + \sum_{j=1}^{n} c_j c(\tilde{g_j}') \bmod N$, which is a result of equation (6).

Consider the other case when $\mathcal{P}$ knows $\{s_i\}$ and $\tilde{s}$ satisfying equation (6); and every $i \in \{1, ..., n\}$ satisfies $s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod N$. In this case, equations (5) hold.

Lemma 4 *Assume that $\mathcal{P}$ knows $\{A_{ij}\}$ and $\{\alpha_i\}$ satisfying equations (12) and (13). If equations (6) and (9) hold with non-negligible probability, then either equation (3) holds, or $\mathcal{P}$ can generate $\{a_i\}$ with overwhelming probability such that $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.*

Proof As in Lemma 3, if equation (6) hold, then either equations (5) hold, or $\mathcal{P}$ can generate non-trivial $\{a_i\}$ satisfying $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ with overwhelming probability. For the former case, replace the values of s_i from equations (5) into (9) and we have:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (\sum_{h=1}^{n} A_{hi} A_{hj} - \delta_{ij}) c_i c_j + \sum_{i=1}^{n} (\sum_{j=1}^{n} 2\alpha_j A_{ji} - c(\dot{w}_i)) c_i$$
$$+ (\sum_{j=1}^{n} \alpha_j^2 - c(\dot{w})) = 0 \bmod N$$

where

$$\delta_{ij} = \begin{cases} 1 \bmod N \text{ if } i = j \\ 0 \bmod N \text{ otherwise} \end{cases}$$

So if equation (3) does not hold for some i and j, then the probability that equation (9) holds is negligible.

Lemma 5 *Assume that $\mathcal{P}$ knows $\{A_{ij}\}$ and $\{\alpha_i\}$ satisfying equations (12) and (13). If equations (6) and (8) hold with non-negligible probability, then either equation (4) holds, or $\mathcal{P}$ can generate $\{a_i\}$ with overwhelming probability such that $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.*

Proof As in Lemma 3, if equation (6) hold, then either equations (5) hold, or $\mathcal{P}$ can generate non-trivial $\{a_i\}$ satisfying $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ with overwhelming probability. For the former case, replace the values of s_i from equations (5) into (8) and we have:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(\sum_{h=1}^{n} A_{hi}A_{hj}A_{hk} - \delta_{ijk})c_i c_j c_k + \sum_{i=1}^{n}(\sum_{j=1}^{n} 3\alpha_j A_{ji}^2 - c(\dot{t_i}))c_i^2 +$$
$$\sum_{i=1}^{n}(\sum_{j=1}^{n} 3\alpha_j^2 A_{ji} - c(\dot{v_i}))c_i + (\sum_{j=1}^{n} \alpha_j^3 - c(\dot{v})) = 0 \bmod N$$

where

$$\delta_{ijk} = \begin{cases} 1 \bmod N \text{ if } i = j = k \\ 0 \bmod N \text{ otherwise} \end{cases}$$

So if equation (4) does not hold for some i and j, then the probability that equation (8) holds is negligible.

Lemma 6 *Assume that $\mathcal{P}$ knows $\{A_{ij}\}$ and $\{\alpha_i\}$ satisfying equations (12) and (13), and $\{s_i\}$ and $\tilde{s}$ satisfying equation (6). If equation (7) holds with non-negligible probability, then either the equations*

$$c(g'_i) = c(\prod_{j=1}^{n} g_j^{A_{ji}}),\ i = 1, ..., n \tag{14}$$

$$c(g') = c(\prod_{j=1}^{n} g_j^{\alpha_j}) \tag{15}$$

hold or $\mathcal{P}$ can generate $\{a_i\}$ with overwhelming probability such that $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ and at least one element of $\{a_i\}$ is not 0.

Proof As in Lemma 3, if equation (6) hold, then either equations (5) hold, or $\mathcal{P}$ can generate non-trivial $\{a_i\}$ satisfying $c(\prod_{i=1}^{n} \tilde{g_i}^{a_i}) = 0$ with overwhelming probability. For the former case, replace the values of s_i from equations (5) into (7), we have:

$$c(\prod_{i=1}^{n} g_i^{\sum_{j=1}^{n} A_{ij}c_j + \alpha_i}) = c(g' \prod_{j=1}^{n} {g'_j}^{c_j}) \bmod N$$

$$\Rightarrow c(\frac{\prod_{j=1}^{n} g_j^{\alpha_j}}{g'}) + \sum_{i=1}^{n} c_i c(\frac{\prod_{j=1}^{n} g_j^{A_{ji}}}{g'_i}) = 0 \bmod N$$

So equations (14) and (15) hold.

B.3 Proof of Theorem 7 for Privacy

Based on Theorem 3, proving Theorem 7 is equivalent to proving that the shuffle provides IND-CPA$_S$ if the DCR assumption holds. We need Definition 11 and Lemma 7.

Definition 11 *Let R_m be the set of m-element tuples where all elements are in $\mathbb{Z}^*_{N^2}$ and let $D_m \subset R_m$ be the set of m-element tuples where all elements are N-th residues modulo N^2. The DCR$_m$ problem is defined as the problem of distinguishing instances uniformly chosen from R_m and those uniformly chosen from D_m. The DCR$_m$ assumption states that the DCR$_m$ problem is computationally difficult.*

Lemma 7 *For any $m \geq 1$, the DCR$_m$ assumption holds if the DCR assumption holds.*

Proof We prove the lemma by induction. We prove that if either the DCR assumption holds or the DCR$_{m-1}$ assumption holds, then the DCR$_m$ assumption holds.

We define the subset M_m of R_m to be the set of tuples $I = (x_1, ..., x_m)$ such that $x_1, ..., x_{m-1}$ are N-th residues modulo N^2 and $x_m \in \mathbb{Z}^*_{N^2}$. Hence, D_m is a subset of M_m.

If the DCR$_m$ problem is easy, then we can either distinguish between instances chosen uniformly from R_m and M_m or distinguish between instances chosen uniformly from M_m and D_m. In the former case, it means that the DCR$_{m-1}$ problem is easy. In the following, we show that in the latter case, the DCR problem is easy.

For any $I_1 = (x) \in R_1$, we generate a tuple $I_m \in R_m$ as $I_m = (r_1^N, r_2^N, ..., r_{m-1}^N, x)$ where $r_i \leftarrow \mathbb{Z}^*_N$. If I_1 is chosen uniformly from D_1, then I_m is distributed uniformly in D_m. And if I_1 is chosen uniformly from R_1, then I_m is distributed uniformly in M_m. Therefore, if D_m and M_m are distinguishable, then the DCR problem is easy.

□

We now prove that the shuffle provides IND-CPA$_S$ if the DCR assumption holds. Suppose there is a publicly known set, $\{\tilde{g}_i\}_{i=1}^n$, of elements uniformly generated from $\mathbb{Z}^*_{N^2}$. And suppose the challenge template includes two permutations $\pi_{(1)}, \pi_{(2)} \in S_n$, a list of ciphertexts $L_{in} = (g_1, ..., g_n)$, the list of corresponding plaintexts $L_{in}^{(p)}$ and the corresponding probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$. The actual challenge $o^{\pi_{(k)}}$, which is randomly generated by using $\pi_{(k)}$ ($k = 1$ or 2) and is given to the adversary, includes L_{in}, $L_{in}^{(p)}$, $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$, a list of re-encrypted ciphertexts $L_{out} = (g'_1, ..., g'_n)$ and
$View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}) = (\{\tilde{g}_i\}, \{\tilde{g}_i{}'\}, \tilde{g}', g', \{\dot{t}_i\}, \{\dot{v}_i\}, \{\dot{w}_i\}, \dot{v}, \dot{w}, \{c_i\}, \{s_i\}, \tilde{s}, s, u, v)$
Let $\mathcal{O}^{\pi_{(k)}}$ be the set of all possible $o^{\pi_{(k)}}$.

Let $\mathcal{O}_g$ be the set of all tuples o_g, each of which includes L_{in}, $L_{in}^{(p)}$, $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$, a list of random ciphertexts $L_{out} = (g'_1, ..., g'_n)$, the set $\{\tilde{g}_i\}$, a tuple $(\{\tilde{g}_i{}'\}_{i=1}^n, \{\dot{t}_i\}_{i=1}^n, \{\dot{v}_i\}_{i=1}^n, \{\dot{w}_i\}_{i=1}^n)$ of randomly generated elements of $\mathbb{Z}^*_{N^2}$, a tuple $(\{c_i\}_{i=1}^n, \{s_i\}_{i=1}^n)$ of randomly generated elements of $\mathbb{Z}_N$, a tuple $(\tilde{s}, s, u, v)$ of randomly generated elements of $\mathbb{Z}_N^*$ and $\tilde{g}', g', \dot{v}, \dot{w}$ satisfying:

$$\tilde{g}' = \tilde{s}^N \prod_{j=1}^{n} \tilde{g_j}^{s_j} \tilde{g_j}'^{-c_j} \tag{16}$$

$$g' = s^N \prod_{j=1}^{n} g_j^{s_j} g_j'^{-c_j} \tag{17}$$

$$\dot{v} = u^N (1 + N \sum_{j=1}^{n} (s_j^3 - c_j^3)) \prod_{j=1}^{n} \dot{v_j}^{-c_j} \dot{t_j}^{-c_j^2} \tag{18}$$

$$\dot{w} = v^N (1 + N \sum_{j=1}^{n} (s_j^2 - c_j^2)) \prod_{j=1}^{n} \dot{w_j}^{-c_j} \tag{19}$$

We first prove that if the DCR$_{5n}$ assumption holds, then the actual challenge $o^{\pi_{(k)}}$ uniformly chosen from $\mathcal{O}^{\pi_{(k)}}$ is computationally indistinguishable from a tuple o_g uniformly chosen from $\mathcal{O}_g$.

We show that from an element $I = (h_1, .., h_n, \tilde{h_1}, .., \tilde{h_n}, \overline{t_1}, .., \overline{t_n}, \overline{v_1}, .., \overline{v_n}, \overline{w_1}, .., \overline{w_n})$ of D_{5n} or R_{5n} (see Definition 11), we can generate a random element o_r of $\mathcal{O}^{\pi_{(1)}}$ or $\mathcal{O}_g$ as follows. Choose $\{c_i\}_{i=1}^n$ and $\{s_i\}_{i=1}^n$ uniformly from $\mathbb{Z}_N$ and $\tilde{s}, s, u, v$ uniformly from $\mathbb{Z}_N^*$. Compute

$$\begin{aligned}
\alpha_i &= s_i - c_{\pi_{(1)}(i)} \bmod N,\ i = 1, ..., n \\
g_i' &= h_i g_{\pi_{(1)}^{-1}(i)},\ i = 1, ..., n \\
\tilde{g_i}' &= \tilde{h_i} \tilde{g}_{\pi_{(1)}^{-1}(i)},\ i = 1, ..., n \\
\dot{t_i} &= \overline{t_i}(1 + N3\alpha_{\pi_{(1)}^{-1}(i)}),\ i = 1, ..., n \\
\dot{v_i} &= \overline{v_i}(1 + N3\alpha^2_{\pi_{(1)}^{-1}(i)}),\ i = 1, ..., n \\
\dot{w_i} &= \overline{w_i}(1 + N2\alpha_{\pi_{(1)}^{-1}(i)}),\ i = 1, ..., n
\end{aligned}$$

And compute $\tilde{g}', g', \dot{v}, \dot{w}$ as in Equations (16), (17), (18) and (19). We have o_r=$(L_{in}$, $L_{in}^{(p)}$, $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$, $(g_1', ..., g_n')$, $(\{\tilde{g_i}\}, \{\tilde{g_i}'\}, \tilde{g}', g', \{\dot{t_i}\}, \{\dot{v_i}\}, \{\dot{w_i}\}, \dot{v}, \dot{w}$, $\{c_i\}, \{s_i\}, \tilde{s}, s, u, v))$.

Then $o_r \in \mathcal{O}^{\pi_{(1)}}$ if and only if $I \in D_{5n}$, and $o_r \in \mathcal{O}_g$ if and only if $I \in R_{5n}$. So, if the DCR$_{5n}$ assumption holds, then a random element of $\mathcal{O}^{\pi_{(1)}}$ is computationally indistinguishable from a random element of $\mathcal{O}_g$, and so is from a random element of $\mathcal{O}^{\pi_{(2)}}$.

Therefore, if the DCR$_{5n}$ assumption holds, then a challenge generated from $\pi_{(1)}$, which is a random element of $\mathcal{O}^{\pi_{(1)}}$, is computationally indistinguishable from a challenge generated from $\pi_{(2)}$, which is a random element of $\mathcal{O}^{\pi_{(2)}}$ (as both are computationally indistinguishable from a random element of $\mathcal{O}_g$). Based on Lemma 7, if the DCR assumption holds, then the shuffle achieves IND-CPA$_S$.