Verifiable Shuffles: A Formal Model and a Paillier-Based Efficient Construction with Provable Security

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

[1] School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
{ldn01,rei}@uow.edu.au

[2] Department of Computer and Information Sciences
Ibaraki University 4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan
kurosawa@cis.ibaraki.ac.jp

Abstract. We propose a formal model for security of verifiable shuffles and prove security of a number of recently proposed shuffle schemes in this model. The model is general and can be extended to mix-nets and verifiable shuffle decryption. We propose a new efficient verifiable shuffle system based on Paillier encryption scheme and prove its security in the proposed model.

Keywords: 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 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 [3] 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 system [3], Web browsing [9], electronic voting [18], anonymous payment systems [4], location privacy for mobile networks [16] and mobile IP [4], secure multiparty computation [14] and privacy in advertisements [15].

A mix-net consists of a number of mix-centres that collectively permute and decrypt the mix-net 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, hence called *shuffle decryption*. Mix-nets that use shuffle decryption could be more efficient but in case of failure of one of the mix-centres, they need more computation to recover [8].

The main security property of shuffle systems is providing *unlinkability* of elements of its input to the elements of the output list for outsiders, and so effectively keeping the permutation secret. We refer to this property as *shuffle privacy*. A second 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 the amount of computation and communication that is required for providing 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. In Crypto'01, Furukawa and Sako [6] 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. However in a subsequent paper [7], they noted that the proof system was not zero-knowledge. They however gave a definition of privacy for shuffles and showed that the protocol satisfied that definition. The definition requires that the verifier cannot learn anything about the 'relation' between the output of the shuffle and its input, using the transcript of the protocol. Neff [18,19] and later Groth [13] proposed shuffles that provide zero-knowledge property for their proofs.

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 will be also important for formalising security of mix-nets. Recently proposed attacks [1,20,25] 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 *chosen transcript attack (CTA$_S$)* (similar to chosen ciphertext). A subtle difference between this model and the model of a traditional encryption system is that in this case the adversary does not only specify the distribution of 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. We note that the prover, the shuffle, does not have access to the private key of encryption. This is the first complete model for shuffle security with active adversary and under CPA$_S$ and CTA$_S$. The model can be extended to verifiable shuffle decryption and mix-nets, and so providing a unified framework for security evaluation of these systems. We prove security of Furukawa-Sako, Neff and Groth schemes in this model.

A *second contribution* of this paper is proposing a new efficient verifiable shuffle based on Paillier encryption system [22]. Paillier encryption system provides semantic security against adaptive chosen plaintext attack (CPA) in standard model and similar to El Gamal cryptosystem, it is possible to define a re-encryption operation for it. The shuffle uses Furukawa-Sako approach for characterisation of permutation matrices but has computations over a composite modulus which complicates security proofs (We have to prove Theorem 6 and Theorem 7). We prove privacy and verifiability of the shuffle in our proposed model. The proof technique can also be used to prove privacy of Furukawa-Sako, Neff and Groth schemes in our model. Compared to Furukawa-Sako and Groth, our proof system has a more efficient initialisation phase and similar to Groth's shuffle, does not require the message space to be prime (a product of two primes instead). By using the NM-CCA robust threshold version of Paillier encryption scheme [5], a robust mix-net can be constructed from our verifiable shuffle, as will be shown in the full version of our paper [21].

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 based on Paillier public-key system, its security proofs and efficiency analysis.

2 Background

2.1 Public-Key Encryption Schemes

A public-key encryption scheme consists of three probabilistic polynomial time (PPT) algorithms (G, E, D). The key generation algorithm G on input 1^l outputs (pk, sk) where pk is a public key, sk is the secret key and l is a security parameter. The encryption algorithm E takes as input the public key pk and a plaintext and outputs a ciphertext. The decryption algorithm D takes as input the secret key sk and a ciphertext and outputs a plaintext. A public-key encryption scheme may have a *re-encryption* function. Following the definition in [24], 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': $Pr[c' = R_{pk}(c)] = Pr[c' = E_{pk}(m)]$ (2.1). A public-key scheme with a re-encryption function is denoted by (G, E, D, R). Note that we write $E_{pk}(m)$, $D_{sk}(c)$ and $R_{pk}(c)$ instead of $E(pk, m)$, $D(sk, c)$ and $R(pk, c)$ respectively.

Due to space limitation, for a discussion about encryption security requirements, including semantic security (SS), indistinguishability (IND) and non-malleability (NM) against chosen plaintext attacks (CPA) and chosen ciphertext attacks (CCA), we refer to the full version of this paper [21].

2.2 Paillier Public-Key System

Key generation: Let $N = pq$, where p and q be large primes. Denote λ as Carmichael value of N, so $\lambda = lcm(p-1, q-1)$. 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 $m \in Z_N$ can be encrypted by choosing an $r \in_R Z_N^*$ (i.e. chosen randomly and with uniform distribution from Z_N^*) and computing the ciphertext $g = r^N(1 + mN)$. [1]

Re-encryption: A Paillier ciphertext g for a plaintext m can be re-encrypted as $g' = r'^N \times g$ for the same plaintext m, where $r' \in_R Z_N^*$. The re-encryption satisfies the condition (2.1) above.

Decryption: Ciphertext $g \in Z_{N^2}^*$ can be decrypted as $m = L(g^\lambda \bmod N^2)/\lambda \bmod N$, where the function L takes its input from the set $\{u < N^2 | u = 1 \bmod N\}$ and is defined as $L(u) = (u-1)/N$.

Decisional Composite Residuosity Assumption (DCRA): A number $z \in Z_{N^2}^*$ is said to be an *e-th residue mod* N^2 if there exists a number $y \in Z_{N^2}^*$ such that $z = y^e$. DCRA states that there is no polynomial time distinguisher for the N-th residues modulo N^2.

Security: Paillier encryption scheme has SS-CPA if and only if DCRA holds.

NM-CCA robust threshold encryption scheme: Using the twin-encryption paradigm of [17], Shamir sharing scheme [23], the proof of equality of discrete logs and a simulation-sound proof of equality of plaintexts, Fouque and Pointcheval [5] proposed a NM-CCA robust threshold encryption scheme based on Paillier public-key system that is proved secure in the random oracle model. This encryption system can be used to construct a robust mix-net.

2.3 Furukawa-Sako Shuffle

Furukawa and Sako [6] proposed an efficient verifiable shuffle based on El Gamal public-key system. In their scheme, a permutation is represented as a matrix (Definition 1) and their proof system is based on proving two equations based on the matrix (Theorem 1). However, Furukawa-Sako's proof of zero-knowledgeness is not correct [7].

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

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

[1] Paillier encryption is originally defined as $g = r^N e^m$, where $e \in Z_{N^2}^*$ and its order in modulo N^2 is a non-zero multiple of N. For efficiency we use $e = 1 + N$. Our results do not depend on this choice and are true for all values of e.

Theorem 1. *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 Notation and Terminology

For a list L of elements, $|L|$ denotes the size of the list, $L[i]$ denotes the i^{th} element of the list and $\pi(L)$ the list of elements in L permuted by a permutation π. Let T_n denote the set of all permutations on $\{1, ..., n\}$. A *positive polynomial* is a polynomial for which the leading coefficient is positive. Let $poly(n)$ refer to some fixed but unspecified polynomial and U_n denote a random variable uniformly distributed over $\{0,1\}^n$. When a PPT algorithm M takes an input x and produces an output y, we write $y \overset{R}{\leftarrow} M(x)$ and denote $C_M^{x,y}$ the probabilistic input (sequence of internal random coin tosses) of M. For example, if Paillier ciphertext $g = r^N(1+mN)$, then $C_E^{(N,m),g} = r$. We can abuse this notation by writing $C_{E_{pk}}^{m,c}$ instead of $C_E^{(pk,m),c}$ and similar for D_{sk} and R_{pk}. We use $C_M^{L_x,L_y}$ to denote the list of probabilistic inputs of M where the *ith* element of the list is the probabilistic input that takes the *ith* element of the list L_x to the *ith* element of the list L_y. The set of possible outputs of M on input x is denoted by $[M(x)]$.

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 the defined types. An interactive proof system $(\mathcal{P}, \mathcal{V})$ consists of two party: 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 [10]. For a proof system $(\mathcal{P}, \mathcal{V})$, $View_{\mathcal{V}}^{\mathcal{P}}(x)$ denotes all that $\mathcal{V}$ can see from the execution of the proof system on input x (in other words, the transcript of the proof system on input x).

3.2 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 scheme with a re-encryption function. 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}.*

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

A shuffle takes a list of ciphertexts and outputs a permuted list of their re-encryptions. If verifiable, it then runs a proof system to prove that the output is really a permutation of the re-encryptions of 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 scheme with a re-encryption function (G, E, D, R). Suppose the 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 $\pi \in T_n$, 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.*

3.3 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 $(\mathcal{P}, \mathcal{V})$.

Verifiability. The proof system proves that the output of the shuffle is a permutation of the 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 (non-uniform) auxiliary input t.

The private input y of the prover does not include the private key sk but may include information about the lists of plaintexts $L_{in}^{(p)}, L_{out}^{(p)}$ and the corresponding probabilistic inputs $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, C_{E_{pk}}^{L_{out}^{(p)}, L_{out}}$. The following definition is for interactive proof systems but can be trivially modified for non-interactive proof systems.

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 polynomial $p()$, there exists an l_0 such that for all $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

$$Pr\begin{bmatrix} \langle \mathcal{P}(y), \mathcal{V}\rangle(pk, L_{in}, L_{out}) = 1 \text{ given } (pk, L_{in}, L_{out}) \in \mathcal{L}_{\mathcal{RP}} \\ \text{where } \ (pk, sk) \overset{R}{\leftarrow} G(1^l), \\ (L_{in}, L_{out}) \overset{R}{\leftarrow} A(pk, t), \\ y \leftarrow w(pk, L_{in}, L_{out}) \end{bmatrix} > 1 - \frac{1}{p(l)}$$

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

$$Pr\begin{bmatrix} \langle \mathcal{B}(y), \mathcal{V}\rangle(pk, L_{in}, L_{out}) = 1 \text{ given } (pk, L_{in}, L_{out}) \notin \mathcal{L}_{\mathcal{RP}} \\ \text{where } \ (pk, sk) \overset{R}{\leftarrow} G(1^l), \\ (\pi, L_{in}, L_{out}) \overset{R}{\leftarrow} A(pk, t), \\ y \leftarrow (\pi, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}, L_{out}^{(p)}, C_{E_{pk}}^{L_{out}^{(p)}, L_{out}}) \end{bmatrix} < \frac{1}{p(l)}$$

Privacy. First assume the algorithm S performs correctly and the aim is to model concealment of the permutation. The shuffle is 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 using notions of 'concealment' of plaintexts in encryption systems 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 obtains permutations that correspond to valid shuffle transcripts that it adaptively chooses. The transcript of a verifiable shuffle's execution consists of the lists of input ciphertexts and output ciphertexts and the transcript of the proof system. An *adaptive attack* has 4 steps.

• *Key generation:* A trusted party generates the keys $(pk, sk) \overset{R}{\leftarrow} G(1^l)$. The adversary is given $(1^l, pk)$. (sk is used for decryption and is also not given to the shuffle.)

• *Oracle queries:* The adversary (adaptively) uses the information obtained so far to make queries to some oracles. The types of oracles determine the type of the attack (CPA$_S$ and CTA$_S$). After making a number of such queries, the adversary moves to the next stage.

• *Challenge generation:* Using the information obtained so far, the adversary specifies a *challenge template*, according to which an actual challenge will be generated.

• *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. 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 only to A_1. We write $(\tau, \delta) \overset{R}{\leftarrow} A_1^{Oracles}(pk, t)$, and $v \overset{R}{\leftarrow} A_2^{Oracles}(\delta, o)$. where *Oracles* specifies 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 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 triplet of polynomial-size circuits Π_n, h_n, f_n and a list of n ciphertexts L_{in}. Π_n specifies a distribution on the set T_n (of all permutations on $\{1, ..., n\}$): it takes $poly(l)$-bit (l is the security parameter) input and outputs a permutation $\pi \in T_n$. 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 transcript of the proof system, $View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out})$, the partial information $h_n(\pi)$, the list of n input 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 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 $\pi_{(1)}, \pi_{(2)} \in T_n$ and a list of n ciphertexts 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 transcript 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 has access to two oracles. The first oracle takes a permutation and a list of input ciphertexts and produces a ciphertext list output by the algorithm S and corresponding to the input list, and the transcript of the proof system $(\mathcal{P}, \mathcal{V})$ when the shuffle interacts with an honest verifier. The second oracle takes a plaintext and returns the ciphertext encrypted by algorithm E_{pk} corresponding to plaintext. The adversary is adaptive and queries are chosen by taking the results of all previous queries into account. We note that in CPA$_S$ the adversary can compute all answers to the queries using public information however using oracles provides consistency in our presentation.

Definition 6. *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to have semantic privacy under chosen permutation attack (SP-CPA$_S$) 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 polynomial $p()$, there exists an l_0 such that for all $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

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

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

2. *For every l and t above, the parts (Π_n, h_n, f_n) in the random variables $A_1^{(S,(\mathcal{P},\mathcal{V})),E_{pk}}(pk, t)$ and $A_1'(1^l, t)$ are identically distributed.*

Definition 7. *A verifiable shuffle $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ is said to provide indistinguishability under chosen permutation attack (IND-CPA$_S$) if for every pair of PPT* *oracle machines, A_1 and A_2, for every positive polynomial $p()$, there exists an l_0 such that for all $l > l_0$ and $t \in \{0,1\}^{poly(l)}$, it holds that*

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

where

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

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

The following theorem shows the equivalence of SP-CPA$_S$ and IND-CPA$_S$. The proof is similar to the proof of the equivalence of SS-CPA and IND-CPA [11].

Theorem 2. *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 two oracles described before, the adversary has access to another oracle T, that takes a transcript of a shuffle execution and returns the corresponding permutation if the transcript is valid, and an error symbol, otherwise. We assume that in step 4, the adversary can not use the transcript in the actual challenge as the query to T.

We note that if the shuffle does not provide verifiability, then the adversary can always learn the permutation. This is because the shuffle transcript consists of an input and an output ciphertext list and the adversary can use re-encryption to generate another input and output ciphertext list that he can present to T and obtain the permutation. For verifiable shuffles, the attack can be prevented by using proof systems. For example, informally, by adding proofs of knowledge in the verifiability proof, construction of new valid transcripts from old ones can be prevented.

Definitions of SP-CTA$_S$ and IND-CTA$_S$ and the theorem stating their equivalence are quite similar to Definition 6, 7 and Theorem 2 and can be found in the full version of this paper [21].

3.4 Applications to Some Verifiable Shuffles

The following theorems shows security of the Furukawa-Sako [6], Neff [19] and Groth [13] verifiable shuffles. The proof of Verifiability (Theorem 3) can be constructed from proofs of Completeness and Soundness in the corresponding papers. The proof of SP-CPA$_S$ (Theorem 4) is similar to the verifiable shuffle in the next section.

Theorem 3. *Furukawa-Sako shuffle provides Verifiability if Discrete Log Assumption holds. Neff shuffle achieves Verifiability with overwhelming probability. Groth shuffle provides Verifiability if the encryption scheme provides SS-CPA and the commitment scheme is secure.*

Theorem 4. *Furukawa-Sako and Neff shuffles provide SP-CPA$_S$ if Decisional Diffie-Hellman Assumption holds. Groth shuffle provides SP-CPA$_S$ under conditions specified in Theorem 3.*

4 A Verifiable Shuffle Based on Paillier Public-Key System

4.1 Description

In our verifiable shuffle, the public-key re-encryption scheme $\mathcal{RP}$ is the Paillier scheme. The public key is $pk = N$ and the secret key is $sk = \lambda$. The algorithm S takes pk, a list of Paillier ciphertexts $g_1, ..., g_n \in Z^*_{N^2}$ and a permutation π and outputs another list of Paillier ciphertexts $g'_1, ..., g'_n \in Z^*_{N^2}$. The proof system $(\mathcal{P}, \mathcal{V})$ is described in the next subsection.

4.2 Proof System

The proof system $(\mathcal{P}, \mathcal{V})$ proves that the prover $\mathcal{P}$ knows permutation π and $r_1, ..., r_n \in Z^*_N$ so that $g'_i = r_i^N g_{\pi^{-1}(i)}$. 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 in $Z^*_{N^2}$, which is generated randomly and independently from the ciphertexts. Therefore if $DCRA$ holds, then it is easy to show that without knowing the secret key sk, it is infeasible to obtain non-trivial $\{a_i\}$ so that there exists $z \in Z^*_N$ satisfying $\prod_{i=1}^n \tilde{g}_i{}^{a_i} = z^N$ in polynomial time. Represent the permutation π by a permutation matrix $(A_{ij})_{n\times n}$, the protocol is as follows:

1. $\mathcal{P}$ generates: $\alpha_i \in_R Z_N, \alpha, \tilde{r_i}, \tilde{\alpha}, \delta_i, \rho, \rho_i, \tau, \tau_i \in_R Z^*_N,\ i = 1, ..., n$
2. $\mathcal{P}$ computes:

$$\tilde{g_i}' = \tilde{r_i}^N \prod_{j=1}^n \tilde{g_j}^{A_{ji}};\ \dot{w_i} = \tau_i^N(1 + N\sum_{j=1}^n 2\alpha_j A_{ji}),\ i = 1, ..., n;\ g' = \alpha^N \prod_{j=1}^n g_j^{\alpha_j}$$

$$\tilde{g}' = \tilde{\alpha}^N \prod_{j=1}^n \tilde{g_j}^{\alpha_j};\ \dot{v} = \rho^N(1 + N\sum_{j=1}^n \alpha_j^3);\ \dot{w} = \tau^N(1 + N\sum_{j=1}^n \alpha_j^2)$$

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

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 $\{c_i\}$, $c_i \in_R Z_N$, $i = 1, ..., n$

5. $\mathcal{P} \longrightarrow \mathcal{V}$: the following responses

$$s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod N, i = 1, ..., n;\ \tilde{s} = \tilde{\alpha} \prod_{i=1}^{n} \tilde{r_i}^{c_i} \tilde{g_i}^{d_i} \bmod N$$

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

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:

$$\tilde{s}^N \prod_{j=1}^{n} \tilde{g_j}^{s_j} = \tilde{g}' \prod_{j=1}^{n} \tilde{g_j}'^{c_j};\ u^N (1 + N \sum_{j=1}^{n} (s_j^3 - c_j^3)) = \dot{v} \prod_{j=1}^{n} \dot{v_j}^{c_j} \dot{t_j}^{c_j^2}$$

$$s^N \prod_{j=1}^{n} g_j^{s_j} = g' \prod_{j=1}^{n} g_j'^{c_j};\ v^N (1 + N \sum_{j=1}^{n} (s_j^2 - c_j^2)) = \dot{w} \prod_{j=1}^{n} \dot{w_j}^{c_j}$$

4.3 Security

The proposed shuffle provides Verifiability and SP-CPA$_S$ under DCRA, as stated in Theorem 5 and Theorem 8.

Theorem 5. *The shuffle achieves Verifiability if DCRA holds.*

To prove Theorem 5, we need Theorem 6 and Theorem 7. The rest of the proof of Theorem 5 is quite similar to the Completeness and Soundness proofs of Furukawa-Sako scheme [6] and can be found in the full version of this paper [21].

Theorem 6. *A matrix $(A_{ij})_{n \times n}$ is a permutation matrix modulo N or there exists i', j' such that $gcd(A_{i'j'}, N) = p$, if for all i, j, k, both*

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

$$\sum_{l=1}^{n} A_{li} A_{lj} A_{lk} = \begin{cases} 1 \ mod\ N\ if\ i = j = k \\ 0 \ mod\ N\ otherwise \end{cases} \tag{2}$$

hold.

Proof. Suppose a matrix (A_{ij}) satisfying (1) and (2), then (A_{ij}) is a permutation matrix mod p and also a permutation matrix mod q, based on Theorem 1. Therefore, if (A_{ij}) is not a permutation matrix mod N, then there exists i', j' such that $A_{i'j'} = 0$ mod p and $A_{i'j'} = 1$ mod q. It leads to $gcd(A_{i'j'}, N) = p$.

Theorem 7. *Denote $\langle S \rangle_k$ the vector space spanned by a set of vectors S in modular k and $|S|$ the number of elements in S. Suppose a set of vectors $S_n = \{(1, c_1, ..., c_n) | (c_1, ..., c_n \in Z_N) \wedge (\nexists Q_n \subseteq S_n : |Q_n| = n + 1 \wedge \langle Q_n \rangle_p = Z_p^{n+1} \wedge \langle Q_n \rangle_q = Z_q^{n+1})\}$. Then $|S_n| \leq (p + q) N^{n-1}$.*

Proof. It is proved by induction as follows

- $n = 1$: Suppose a set of vectors $S_1 \subseteq \{(1,c)|c \in Z_N\}$ satisfying $|S_1| > (p+q)$; and a vector $(1,c_1) \in S_1$. Consider a set $R_1 = \{(1, c_1 + kp \bmod N)|k \in Z_q\} \cup \{(1, c_1 + kq \bmod N)|k \in Z_p\}$. As $|R_1| = p+q-1$, there exists $c_1' \in Z_N$ so that $(1,c_1') \in S_1$ but $(1,c_1') \notin R_1$. Then $Q_1 = \{(1,c_1),(1,c_1')\}$ satisfying $|Q_1| = 2 \wedge \langle Q_1\rangle_p = Z_p^2 \wedge \langle Q_1\rangle_q = Z_q^2)\}$.
- Assume it is right for n. We prove it is also right for $n+1$. Let a set $S_{n+1} = \{(1,c_1,...,c_{n+1})|(c_1,...,c_{n+1} \in Z_N) \wedge (\nexists Q_{n+1} \subseteq S_{n+1} : |Q_{n+1}| = n+2 \wedge \langle Q_{n+1}\rangle_p = Z_p^{n+2} \wedge \langle Q_{n+1}\rangle_q = Z_q^{n+2})\}$. Consider $S_n' = \{(1,c_1,...,c_n)|\exists c_{n+1} \in Z_N : (1,c_1,...,c_n,c_{n+1}) \in S_{n+1}\}$, there are two possibilities:
  1. If $\nexists Q_n' \subseteq S_n' : |Q_n'| = n+1 \wedge \langle Q_n'\rangle_p = Z_p^{n+1} \wedge \langle Q_n'\rangle_q = Z_q^{n+1}$, then $|S_n'| \le (p+q)N^{n-1}$, as the theorem is right for n. So $|S_{n+1}| \le |S_n'|N \le (p+q)N^n$.
  2. If $\exists Q_n' \subseteq S_n' : |Q_n'| = n+1 \wedge \langle Q_n'\rangle_p = Z_p^{n+1} \wedge \langle Q_n'\rangle_q = Z_q^{n+1}$, select a set T of $n+1$ vectors $(1,c_{i1},...,c_{i(n+1)}) \in S_{n+1}$, $i = 1,...,n+1$ so that $Q_n' = \{(1,c_{i1},...,c_{in})\}$
     Let $d = det\begin{pmatrix}1 & c_{11} & ... & c_{1n}\\ .. & .. & .. & ..\\ 1 & c_{(n+1)1} & ... & c_{(n+1)n}\end{pmatrix} \bmod N$, then $gcd(d,N) = 1$, so $d^{-1} \bmod N$ exists.
     For each vector $x = (1,x_1,...,x_{n+1}) \in S_{n+1}$ (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 S_{n+1} leads to either $d_x = 0 \bmod p$ or $d_x = 0 \bmod q$.
     Suppose $d_x = 0 \bmod p$, then $x_{n+1} = d^{-1}F(x_1,...,x_n) \bmod p$, so the number of possible vectors $x = (1,x_1,...,x_{n+1})$ is no more than qN^n. Similar for the case $d_x = 0 \bmod q$, the number of possible vectors $x = (1,x_1,...,x_{n+1})$ is no more than pN^n. So $|S_{n+1}| \le (p+q)N^n$.

Theorem 8. *The shuffle achieves SP-CPA$_S$ if and only if DCRA holds.*

Based on Theorem 2, proving Theorem 8 is equivalent to proving Theorem 9 below. We need Definition 8 and Lemma 1 to prove Theorem 9. Proof of Lemma 1 can be found in the full version of this paper [21].

Definition 8. *Define R_m to be the set of tuples of m elements in $Z_{N^2}^*$ and subset D_m of R_m to be the set of tuples of m N-th residues modulo N^2. We then define the problem of distinguishing instances uniformly chosen from R_m and those from D_m by $DCRA_m$.*

Lemma 1. *For any $m \ge 1$, $DCRA_m$ is easy if and only if DCRA is easy.*

Theorem 9. *The shuffle achieves IND-CPA$_S$ if and only if DCRA holds.*

Proof. Suppose the challenge template includes two permutations $\pi_{(1)}, \pi_{(2)} \in T_n$ and a list of ciphertexts $L_{in} = (g_1, ..., g_n)$. The actual challenge o to the adversary includes L_{in}, the list of corresponding plaintexts $L_{in}^{(p)}$, $C_{E_{pk}}^{L_{in}^{(p)}, L_{in}}$, a list of re-encryption ciphertexts $L_{out} = (g'_1, ..., g'_n)$ and

$$View_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}) = (\{\tilde{g_i}\}, \{\tilde{g_i}'\}, \tilde{g}', g', \{\dot{t_i}\}, \{\dot{v_i}\}, \{\dot{w_i}\}, \dot{v}, \dot{w}, \{c_i\}, \{s_i\}, \tilde{s}, s, u, v)$$

satisfying: $g' = s^N \prod_{j=1}^n g_j^{s_j} {g_j}'^{-c_j}$, $\dot{v} = u^N (1+N\sum_{j=1}^n (s_j^3 - c_j^3)) \prod_{j=1}^n \dot{v_j}^{-c_j} \dot{t_j}^{-c_j^2}$, $\tilde{g}' = \tilde{s}^N \prod_{j=1}^n \tilde{g_j}^{s_j} \tilde{g_j}'^{-c_j}$, $\dot{w} = v^N (1 + N\sum_{j=1}^n (s_j^2 - c_j^2)) \prod_{j=1}^n \dot{w_j}^{-c_j}$.

Compute $I_{\pi_{(1)}} = (h_1, .., h_n, \tilde{h_1}, .., \tilde{h_n}, \overline{t_1}, .., \overline{t_n}, \overline{v_1}, .., \overline{v_n}, \overline{w_1}, .., \overline{w_n})$, where

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

Then $\pi_{(1)}$ is the permutation used for the actual challenge o if and only if $I_{\pi_{(1)}} \in D_{5n}$. Therefore, based on Lemma 1, if the actual challenge o is computationally distinguishable under chosen shuffle attacks, then DCRA is easy, and vice-versa.

4.4 Efficiency

The proposed shuffle has the round efficiency (3 rounds) and the number of exponentiations (about $18n$) of Furukawa-Sako protocol, compared to Groth's protocol with a 7 round proof. The shuffle has less rounds and requires smaller number of exponentiations compared to Neff's protocol with 7 rounds and $23n$ exponentiations. (Note that exponentiations in our case is modulo N^2 which is more expensive than modulo p and so the number of bit operations in Furukawa-Sako's shuffle is smaller.) Compared with Furukawa-Sako and Groth's proof system, our proposed proof system has a more efficient initialization phase. In both those systems for El Gamal ciphertexts, a set of subgroup elements is used. Construction of these elements in general is computationally expensive [19]. Our proof system also relies on a set $(\{\tilde{g_1}, ..., \tilde{g_n}\})$ of elements of $Z_{N^2}^*$ that are just randomly generated.

References

1. M. Abe and H. Imai. Flaws in Some Robust Optimistic Mix-nets. ACISP 2003, LNCS 2727, pp. 39-50.
2. S. Brands. An efficient off-line electronic cash system based on the representation problem. CWI Technical Report CS-R9323, 1993.
3. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2):84-88, 1981.

4. S. Choi and K. Kim. Authentication and Payment Protocol Preserving Location Privacy in Mobile IP. GLOBECOM 2003.
5. P. Fouque and D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. Asiacrypt 2001, LNCS 2248, pp. 351-369.
6. J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle. Crypto 2001, LNCS 2139, pp. 368-387.
7. J. Furukawa, H. Miyauchi, K. Mori, S. Obana and K. Sako. An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling. Financial Cryptography 2002, LNCS 2357.
8. J. Furukawa. Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. PKC 2004, LNCS 2947, pp. 319-332.
9. E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized Web browsing simple, secure, and anonymous. Financial Cryptography 1997, LNCS 1318, pp. 17-31.
10. O. Goldreich. Foundations of Cryptography, Basic Tools. Cambridge University Press 2001.
11. O. Goldreich. Foundations of Cryptography, Basic Applications. Cambridge University Press 2004.
12. P. Golle, S. Zhong, D. Boneh, M. Jakobsson and A. Juels. Optimistic Mixing for Exit-Polls. Asiacrypt 2002, LNCS 2501, pp. 451-465.
13. J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. PKC 2003, LNCS 2567, pp. 145-160.
14. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. Asiacrypt 2000, LNCS 1976, pp. 162-177.
15. A. Juels. Targeted advertising and privacy too. RSA Conference Cryptographers' Track 2001, LNCS 2020, pp. 408-425.
16. J. Kong and X. Hong. 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, pp. 291-302.
17. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertexts Attacks. ACM STOC 1990, pp. 427-437.
18. A. Neff. A verifiable secret shuffle and its application to e-voting. ACM CCS 2001, pp. 116-125.
19. A. Neff. Verifiable Mixing (Shuffling) of ElGamal Pairs. http://www.votehere.org/vhti/documentation/egshuf.pdf.
20. L. Nguyen and R. Safavi-Naini. Breaking and Mending Resilient Mix-nets. Privacy Enhancing Technologies workshop (PET) 2003, LNCS 2760, pp. 66-80.
21. L. Nguyen, R. Safavi-Naini and K. Kurosawa. Verifiable Shuffles: A Formal Model and a Paillier-based Efficient Construction with Provable Security. Full version. Email: ldn01@uow.edu.au.
22. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt 1999, LNCS 1592, pp. 223-239.
23. A. Shamir. How to Share a Secret. Communications of the ACM, 22:612-613, 1979.
24. D. Wikstrom. The security of a mix-center based on a semantically secure cryptosystem. Indocrypt 2002, LNCS 2551, pp. 368-381.
25. D. Wikstrom. Five Practical Attacks for "Optimistic Mixing for Exit-Polls". SAC 2003, LNCS 3006.