Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability

Jun Furukawa

NEC Corporation, 4-1-1 Miyazaki, Miyamae, Kawasaki 216-8555, Japan
j-furukawa@ay.jp.nec.com

Abstract. In this paper, we propose a scheme to simultaneously prove the correctness of both shuffling and decryption. Our scheme is the most efficient of all previous schemes, as a total, in proving the correctness of both shuffling and decryption of ElGamal ciphertexts. We also propose a formal definition for the core requirement of unlinkability in verifiable shuffle-decryption, and then prove that our scheme satisfies this requirement. The proposed definition may be also useful for proving the security of verifiable shuffle-decryption, hybrid mix network, and other mix-nets.

Keywords: Voting, Shuffle, Decryption, Permutation Hiding

1 Introduction

A mix-net [3] scheme is useful in applications, such as voting, which require anonymity. Crucial to a mix-net scheme is the execution of multiple rounds of shuffling and decryption by multiple, independent mixers, so that none of the output decryptions can be linked to any of the input encryptions.

To ensure the correctness of output, it is desirable to achieve the property of universal verifiability. Early studies, such as those by Sako and Kilian [21] and Abe [1], required vast amounts of computation to prove and verify the correctness of a mix-net without sacrificing unlinkability, However, recently proposed schemes [9, 8, 17, 13] were sufficiently efficient and practical. The schemes of [9, 8] use the property of permutation matrixes, and the schemes of [17, 13] use the fact that polynomials remain invariant under the permutations of their roots. The schemes of [9], [17], and [13] require the respective computation of $18k$, $42k$, and $12k$ modular exponentiations to prove and verify the correctness of a shuffling of k data. The scheme of [8] requires $19k$ modular exponentiations to prove and verify both shuffling and decryption. Groth's scheme [13] is the most efficient.

A result of these recent works is that proving the correctness of decryption now costs as much as proving the correctness of shuffling. Hence, decreasing the cost of proving decryption has also become important in mix-net. The scheme of [8], which was based on the scheme of [9], made it possible to simultaneously prove the correctness of both a shuffling and a decryption; this is more efficient in terms of computation and communication complexity than proving each of these separately.

However, as is mentioned in [8], the scheme of [9, 8] is not a zero-knowledge, and this simultaneously proving technique never yields a zero-knowledge protocol [1]. A simple combination of two zero-knowledge protocols of a verifiable shuffle and of a verifiable decryption also does not yield a zero-knowledge protocol since the intermediate state cannot be simulated. Therefore, a formal definition for the core requirement of unlinkability in verifiable shuffle-decryption, which notion is *weaker* than that of zero-knowledge, is desired.

Such a formal definition will also be useful for considering the security of verifiable mix-net, hybrid mix network, flush mix, and other mix-net [12, 14, 15, 18]. For example, during the decryptions in a hybrid mix network, servers who decrypt ciphertexts generate many extra data that are not themselves encryptions of plain texts (e.g., encrypted secret keys, MAC code, intermediate states, etc.). Hence, even if each component protocol of a hybrid mix network is zero-knowledge, we must confirm that these extra data do not spoil the unlinkability of the total hybrid mix network.

In this paper, we first propose a formal definition for the core requirement of unlinkability in verifiable shuffle-decryption. Next, we propose the most efficient scheme to simultaneously prove the correctness of both shuffling and decryption, which is an improved version of the scheme of [8]. Finally, we prove that the proposed scheme satisfies the proposed requirement.

Our scheme requires roughly $14k$ exponentiations to prove and verify the correctness of both a shuffle and a decryption of k-data, $1344k$ bits of communication, and five rounds. To prove and verify the correctness of both a shuffle and a decryption of k-data with Groth's protocol [13] by using the standard technique of proving the correctness of decryption, we require $15k$ exponentiations, $2528k$ bits of communication, and seven rounds.

Although the security of the schemes of [9] and [8] have never been proven, We are now able to prove that these schemes satisfy the proposed requirement and are secure. Contrary, it is easy to prove that several hybrid mix-network which are vulnerable against resending message attack, such as [15], do not satisfy the proposed requirement.

Our paper is organized as follows. Section 2 introduces the model of shuffle-decryption. Section 3 proposes a definition for the requirement of unlinkability in verifiable shuffle decryption. Section 4 proposes a protocol by which we are able to simultaneously prove the correctness of a shuffle-decryption in an efficient way. Section 5 compares the efficiency of our protocol to prior work.

[1] Whereas, it is easy to make a perfect (and more efficient) zero-knowledge version of the protocol proposed in [9]. This version is presented in an appendix of [7].

2 Notation and Model

2.1 Notation

Let p, q be two primes s.t. $q|p-1$ and $3 \not| (q-1)$, $\mathbb{G}_q$ be an order q subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$, g_0 be an element of $\mathbb{G}_q$, and k be the number of ElGamal ciphertexts to be shuffled, and ℓ be the number of shufflers. Let $x'^{(\lambda)} \in_R (\mathbb{Z}/p\mathbb{Z})^*$ be a private key of λ-th shuffler used for the partial decryption and $y^{(\lambda)} = {g_0}^{x'^{(\lambda)}} \bmod p$ be the corresponding public key. Let ${m_0}^{(\lambda)} = \prod_{\kappa=1}^{\lambda} y^{(\kappa)} \bmod p$ be a public key public key used for the shuffle of λ-th shuffler.

Let $(g_i^{(\ell)}, m_i^{(\ell)}) = ({g_0}^{\bar{r}_i}, {m_0}^{\bar{r}_i} M_i)_{i=1,\ldots,k} \bmod p$ be a tuple of ElGamal ciphertexts to be input ℓ-th shuffler where $\{M_i \in \mathbb{G}_q\}_{i=1,\ldots,k}$ is a set of plain texts to be encrypted, $\{\bar{r}_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ be uniformly and randomly chosen elements of $\mathbb{Z}/q\mathbb{Z}$. Let $(g_i^{(\lambda)}, m_i^{(\lambda)})_{i=1,\ldots,k}$ be a tuple of ciphertexts to be input to λ-th shuffler, who shuffle them with public key $(g_0, m_0^{(\lambda)})$ and then partially decrypts them with the private key $x'^{(\lambda)}$. The resulting tuple of ciphertexts is $(g_i^{(\lambda-1)}, m_i^{(\lambda-1)})_{i=1,\ldots,k}$ which is passed to $(\lambda-1)$-th shuffler. In the rest of the paper, we only consider λ-th shuffler and omit the index (λ).

Treating the public key g_0, m_0 as if it were an element in a ciphertext vector may be awkward, but it gives a more compact and unified representation to variables. Here, the public key is a set, $\{p, q, g_0, m_0, y\}$. P is a prover who shuffles and decrypts and proves the validity of shuffle and decryption to a verifier V.

The only shuffling we have considered in this paper is that of ElGamal cryptosystem, which is the most elegant candidate cryptosystem used for mix-net. However, extensions of the requirements of unlinkability defined in this paper to other cryptosystems are easy.

2.2 ElGamal Shuffle decryption

ElGamal shuffling is a procedure that, given k ElGamal ciphertexts $(g_i, m_i)_{i=1,\ldots,k}$, outputs ElGamal ciphertexts

$$(g'_i, m'_i) = ({g_0}^{s_i} g_{\phi^{-1}(i)}, {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p \quad i = 1, \cdots, k,$$

where $s_i \in_R \mathbb{Z}/q\mathbb{Z}$ for $i = 1, \cdots, k$ and a permutation of indices $\phi : \{1, \ldots, k\} \rightarrow \{i = 1, \ldots, k\}$ are chosen uniformly and randomly.

Shuffling of ElGamal ciphertexts results in the following two important properties:

1. There exists a permutation ϕ s.t. equations $D_x((g'_i, m'_i)) = D_x((g_{\phi^{-1}(i)}, m_{\phi^{-1}(i)}))$ hold for all i. Here, $D_x(\cdot)$ is a decryption algorithm that uses the private key x.
2. As long as the decision Diffie-Hellman problem is difficult to solve, no polynomially bounded algorithm, given only $p, q, g, y, (g_i, m_i), (g'_i, m'_i); i = 1, \cdots, k,$ has an advantage over the random-guessing algorithm in guessing any part of permutation ϕ for uniformly and randomly chosen $g_0, m_0, s_i, \bar{r}_i, \phi$.

ElGamal shuffle decryption is a combination procedure of ElGamal shuffling and partial decryption that, given k ElGamal ciphertexts $(g_i, m_i); i = 1, \cdots, k$, outputs ElGamal ciphertexts

$$(g'_i, m'_i) = ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g'_i}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p \quad i = 1, \cdots, k, \tag{1}$$

where $s_i \in_R \mathbb{Z}/q\mathbb{Z} i = 1, \cdots, k$ and ϕ are chosen uniformly and randomly. Here, the multiplication by ${g'_i}^{-x'}$ in the second term has the effect of partial decryption.

A sequence of shuffles-decryptions composes a mix-net[21]. In this paper, we propose a formal definition for the core requirement of unlinkability in this verifiable ElGamal shuffle-decryption, and then we propose an efficient verifiable ElGamal shuffle-decryption.

3 Complete permutation hiding

We propose here the notion of *complete permutation hiding* (CPH) as a core requirement of unlinkability in verifiable shuffle-decryption. If a verifiable shuffle-decryption is CPH, honest verifiers will learn nothing new about its permutation from an interaction with a prover in an **overwhelming** number of cases of random tape that a prover has chosen uniformly and randomly, whereas, if the protocol is zero-knowledge, verifiers will learn nothing new in **every** case of the random tape. In other words, we define CPH so that verifiers learn nothing about the permutation in an overwhelming number of cases of common input X_n and witness W_n that the generator G_R (defined below) outputs.

Let I_n be a *set* of domain parameters $1^n, p, q$, where p and q are primes and are the lengths of the polynomial of n, private key $\bar{x}$, plain texts $\{M_i \in \mathbb{G}_q\}_{i=1,\ldots,k}$, and random tape Z_n. Let $enc(U)$ be an *encoding of a probabilistic polynomial time (PPT) Turing machine* U which generates cipher-texts $(g_i, m_i)_{i=1,\ldots,k}$ input to the shuffle-decryption procedure. We assume the existence of a knowledge extractor that can extract $\{\bar{r}_i\}_{i=1,\ldots,k}$ such that ${g_0}^{\bar{r}_i} = g_i$ from U. This assumption is satisfied if all generators of cipher-texts are imposed to prove the knowledge of $\bar{r}_i$, and such a compulsion prevents an adaptively chosen cipher-text attack.

Definition 1. *Given* $I_n (= \{1^n, p, q, \bar{x} \in \mathbb{Z}/q\mathbb{Z}, \{M_i \in \mathbb{G}_q\}_{i=1,\cdots,k}, Z_n\})$ *and* $enc(U)$*,* instance Generator G_R *chooses* $g_0 \in_R \mathbb{G}_q, x' \in_R \mathbb{Z}/q\mathbb{Z}$*,* $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$*, and a permutation* ϕ *uniformly and randomly and computes;*

$$m_0 = {g_0}^{x'+\bar{x}}, y = {g_0}^{x'} \bmod p$$
$$(g_i, m_i) = U(I_n, g_0, y) \in \mathbb{G}_q \times \mathbb{G}_q$$
$$(g'_i, m'_i) = ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g_i}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p.$$

G_R *then outputs common input* X_n *and witness* W_n*:*

$$X_n = \{p, q, y, \bar{x}, g_0, m_0, \{(g_i, m_i)\}_{i=1,\cdots,k}, \{(g'_i, m'_i)\}_{i=1,\cdots,k}\},$$
$$W_n = \{\phi, \{s_i\}_{i=1,\cdots,k}, x'\}.$$

In the above definition, U is a PPT Turing machine that plays the role of (malicious and colluding) players who generate cipher-texts $\{(g_i, m_i)\}$. Although U is determined before the public parameter is generated, it does not lose generality because it has this public parameter as an input. In a case where U realizes honest players, it outputs

$$(g_i, m_i) = ({g_0}^{\bar{r}_i}, M_i {m_0}^{\bar{r}_i}) \bmod p$$

using random numbers $\{\bar{r}_i\}_{i=1,\ldots,k}$ generated from the random tape Z_n.

We say X_n and W_n *satisfy relation* R if the following equations are satisfied:

$$m_0 = {g_0}^{x'+\bar{x}}, y = {g_0}^{x'} \pmod p$$
$$(g'_i, m'_i) = ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g_i}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \pmod p.$$

We denote this fact as $(X_n, W_n) \in R$. If there exists a witness W_n for a common input X_n that satisfies $(X_n, W_n) \in R$, common input X_n is a *correct shuffle-decryption*. Generator G_R outputs such a X_n.

Definition 2. *Let $View_V^P(X_n, W_n)$ be V's* view of an interaction *with P, which is composed of the common input X_n, messages V receives from P, random tape input to V, and messages V sends to P during joint computation employing X_n, where P has auxiliary input W_n s.t., $(X_n, W_n) \in R$. $View_V^P$ is an abbreviation of $View_V^P(X_n, W_n)$.*

We consider the case when a semi-honest verifier may collude with malicious players who encrypt the ciphertexts and other provers who shuffle and decrypt in the same mix-net. Such a verifier and players may obtain partial information regarding the plain texts $\{M_i\}$, private key $\bar{x}$ (the sum of other prover's private keys in the mix-net), random tapes of players, and even a part of the permutation ϕ in addition to $View_V^P$. Moreover, they may obtain the results of other shuffle-decryptions executed by the same prover.

Then it is reasonable to describe this extra information as $H(I_n, enc(U), X_n, \phi)$ and input cipher-texts generated by the malicious player as $U(I_n, g_0, y)$ using PPT Turing machines $H(\cdot)$ and $U(\cdot)$. Note that $\{s_i\}$ are not included in the arguments of H, because we consider only the case where the prover never reveals these values to any one and the case where the prover never uses the same $\{s_i\}$ for other shuffle-decryptions.

Even though the verifier and the players may obtain the results of other shuffle-decryptions executed by the same prover who uses x', we do not include x' into the input of U and H. Instead, we assume that there exists a PPT Turing machine K such that the distribution of $View_V^P$ for such H and U and that of $K(I_n, g_0, y, enc(U), \phi)$ are the same. We denote this as $View_V^P \approx K(I_n, g_0, y, enc(U), \phi)$. The exclusion of x' is crucial because it enables us to consider the security of shuffle-decryption over the distribution of X_n i.e., of x'.

We describe information about the permutation ϕ that verifiers try to learn as $f(\phi)$ using PPT Turing machine f. This description can be justified because the expression $f(\phi)$ is sufficient to express any bit of ϕ and any kind of check sum for ϕ.

Now we can say that a verifiable shuffle-decryption protocol hides its permutations completely with respect to G_R - i.e., CPH occurs - if there exists a probabilistic polynomial time algorithm E'^E (which has black box access to E) with inputs X_n and $H(I_n, enc(U), X_n, \phi)$ that suffers no disadvantage with respect to learning anything about the permutations compared to any probabilistic polynomial time verifier E having input $View_V^P$ and $H(I_n, enc(U), X_n, \phi)$. This leads to,

Definition 3. (complete permutation hiding) *A verifiable shuffle decryption protocol* (P, V, G_R) *achieves* complete permutation hiding *if*

$$\begin{aligned}
&{}^{\exists}E'^E\ {}^{\forall}E\ {}^{\forall}H\ {}^{\forall}f\ {}^{\forall}U\ {}^{\forall}c > 0\ {}^{\exists}N\ {}^{\forall}n > N\ {}^{\forall}I_n\\
&\Pr[E(View_V^P, H(I_n, enc(U), X_n, \phi)) = f(\phi)]\\
&< \Pr[E'^E(X_n, H(I_n, enc(U), X_n, \phi)) = f(\phi)] + \frac{1}{n^c}, \qquad (2)\\
&\mathit{and}\\
&{}^{\exists}K\ \ View_V^P \approx K(I_n, g_0, y, enc(U), \phi)
\end{aligned}$$

where E', E, H, f, U, K *are PPT Turing machine. The left probability in Eq.(2) is taken over the distribution of the random tapes input to* G_R, [2] P, V, H, *and* E. *The right probability in Eq.(2) is taken over the distribution of the random tapes input to* G_R, H, E', *and* E. E' *may use* E *as a black box.*

If the verifiable shuffle-decryption protocol is CPH, we can say that for **every** input ciphertexts set $\{(g_i, m_i)\}$ and its corresponding output ciphertexts set $\{(g'_i, m'_i)\}$, whatever an honest verifier who has partial information $(H(I_n, enc(U), X_n, \phi))$ about the common input (X_n), can learn about the permutation (ϕ) after interacting with a prover, can also - in an **overwhelming** number of cases of common input (X_n)- be efficiently computed from that common input (X_n) and that partial information $(H(I_n, enc(U), X_n, \phi))$ alone using a PPT Turing machine E' without interaction with the prover as long as the prover has chosen the private key x', permutation ϕ, and random numbers $\{s_i\}$ uniformly and randomly.

Note that we are considering the case even where malicious and colluding players, who have the results of other shuffle-decryptions with the same x', are engaged in generating $\{(g_i, m_i)\}$ of common input. Hence, CPH guarantees security when shuffle-decryptions with the same private key are repeatedly executed [3].

[2] Since the probability is taken over a distribution containing x', we have excluded any adversary who knows x'.

[3] The definition of shuffle-decryption stated in [8] is "No polynomially bounded adversary can compute any partial information of the permutation from the protocol". Unlike our new definition, this definition does not mention the case where the verifier

Extensions of the proposed definition for requirements regarding unlinkability to other mix-net systems (in the sense that verifiers can learn nothing new about the permutation in an overwhelming number of cases of common input) are easy. Hence, extended-CPHs may be suitable measures of the security of verifiable shuffle-decryptions, verifiable mix-nets, verifiable hybrid mix networks, and other verifiable mix-nets.

4 Proposed Verifiable Shuffle Decryption

In this section, we propose a CPH verifiable shuffle decryption scheme, which is *special* in the sense that the verifier's random tape is identical to its challenge. The proposed protocol is the most efficient of all previous schemes, as a total, to prove the correctness of both shuffling and decryption of ElGamal ciphertexts. The scheme requires five rounds.

4.1 Permutation Matrix

Our scheme uses the property of permutation matrix defined below.

Definition 4. *Let q be a prime. A matrix $(A_{ij})_{i,j=1,\cdots,k}$ is a* permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ *if it satisfies*

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

for a permutation function $\phi : \{1, \ldots, k\} \to \{1, \ldots, k\}$.

Using a permutation matrix (A_{ji}), which corresponds to a permutation ϕ, we find that Eq. (1) can be expressed as

$$(g'_i, m'_i) = ({g_0}^{s_i} \prod_{j=1}^{k} g_j^{A_{ji}}, g_i'^{-x'} m_0^{s_i} \prod_{j=1}^{k} m_j^{A_{ji}}) \bmod p. \tag{3}$$

Therefore, proving the correctness of the shuffle is equivalent to proving the existence of a $x' \in \mathbb{Z}/q\mathbb{Z}$, an $s_i \in \mathbb{Z}/q\mathbb{Z}$ for $i = 1, \ldots, k$ and a permutation matrix $(A_{ji})_{i,j=1,\ldots,k}$ which satisfy Eq. (3).

The following theorem is the key to constructing the proposed protocol.

Theorem 1. ([9] Theorem 1) *Let q be a prime. A matrix $(A_{ij})_{i,j=1,\ldots,n}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ $\Leftrightarrow$*

$$\sum_{h=1}^{n} A_{hi} A_{hj} A_{hk} = \delta_{ijk} \triangleq \begin{cases} 1 \pmod q & \textit{if} \quad i = j = k \\ 0 \pmod q & \textit{if otherwise} \end{cases} \quad \textit{and} \tag{4}$$

$$\sum_{h=1}^{n} A_{hi} A_{hj} = \delta_{ij} \triangleq \begin{cases} 1 \pmod q, & \textit{if } i = j \\ 0 \pmod q, & \textit{if } i \neq j \end{cases} \tag{5}$$

has already obtained partial information before the protocol begins and where the shuffle-decryptions with the same private key are repeatedly executed. These cases seem to occur quite often.

for all i, j, and k.

Proof. See the proof of Theorem 1 in [9] or appendix of [7].

Theorem 2. *For $3 \not| (q-1)$, a matrix $(A_{ij})_{i,j=1,\ldots,n}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ $\Leftrightarrow$ Eq. (4) holds.*

Proof. ($\Rightarrow$) is trivial. ($\Leftarrow$); From the proof of Theorem 1 in [9], if Eq.(4) holds, then there is only one non-zero element e_i in the $i-th$ row and it must satisfies $e_i^3 = 1 \bmod q$. Because $3 \not| (q-1)$ implies that 1 is the only cubic root of 1 in $\mathbb{Z}/q\mathbb{Z}$, e_i must be 1. Therefore, matrix $(A_{ij})_{i,j=1,\ldots,n}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$.

The soundness of our scheme depends directly on Theorem 2.

4.2 Protocol Structure and Tricks for Efficiency

The verifiable shuffle decryption protocol we will propose in this section is almost the same as the scheme proposed in [8]. The proposed scheme and the scheme of [8] are roughly composed of four proofs. These are, (i) generation of $\{f'_i\}_{i=1,\cdots,k}$ and a proof of knowledge of s_i and (A_{ji}) that satisfy

$$f'_i = f_0^{s_i} \prod_{j=1}^{k} f_j^{A_{ji}} \bmod p \quad i = 1, \cdots, k, \tag{6}$$

for uniformly and randomly chosen $f_\mu \in_R \mathbb{G}_q; (\mu = 0, \ldots, k)$, (ii) proof that (A_{ji}) whose knowledge proved in (i) is a permutation matrix (using Theorem 1 or 2), (iii) proof that s_i and (A_{ji}) whose knowledge proved in (i) also satisfies Eq. (3), and (iv) proof of knowledge of the decryption key.

In Proof (ii), there are commitment, challenge, and response phase. The main difference between our scheme and the scheme of [8] is that we have introduced the values f_{-2}, f_{-1} in the proposed scheme. Because of these values f'_is in the commitment are modified from $f'_i = f_0^{A_{0i}} f_{\phi^{-1}(i)}$ to $f'_i = f_{-2}^{A_{-2i}} f_{-1}^{A_{-1i}} f_0^{A_{0i}} f_{\phi^{-1}(i)}$. As a results, we have more redundancy (A_{-2i}, A_{-1i}) to generate the f'_i. Then we adjusted A_{-2i}, A_{-1i} so that some values in the commitment to be zero, which decreased the number of terms in checking equations in the response phase [4]. Another difference between them is that the proposed scheme adopts the prime q such that $3 \not| q-1$. Because of this, verifiers do not need to confirm that Equation (5) holds [5] any more. The other difference between them is with respect to the verification of Eq. (9). A verifying that Eq. (9) holds, is equivalent to verifying equations

$$\prod_{\nu=-2}^{k} f_\nu{}^{r_\nu} = f'_0 \prod_{i=1}^{k} f'^{c_i}_i \ , \quad \prod_{\nu=-2}^{k} f_\nu{}^{r'_\nu} = \tilde{f}'_0 \prod_{i=1}^{k} f'^{c_i{}^2}_i \pmod p$$

[4] The equation related to Equation 12 is the 7-th equation in the verification phase of the scheme of [8]. We can see that terms quadratic and linear to the challenge are disappeared in the proposed protocol.

[5] 8-th equation in the verification phase of the scheme of [8].

hold, where the former is more efficient [6].

4.3 Proposed Protocol

We now describe our verifiable ElGamal shuffle decryption and our scheme. Let public parameters p, q, g_0, y, m_0 and private key x' be as described before. We assume another public key $F_n \triangleq \{f_\nu \in_R \mathbb{G}_q\}_{\nu=-2,\ldots,k}$ are $k+3$ $\mathbb{G}_q$ elements that are uniformly and randomly generated so that neither P nor V can generate non-trivial integers $a, \{a_\nu\}_{\nu=-2,\ldots,k}$ satisfying $g_0^a \prod_{\nu=-2}^{k} f_\nu{}^{a_\nu} = 1 \pmod p$ with non-negligible probability.

ElGamal shuffle decryption P uniformly and randomly chooses $A_{0i} \in_R \mathbb{Z}/q\mathbb{Z}$ for $i = 1, \cdots, k$ and a permutation matrix $(A_{ji})_{i,j=1,\cdots,k}$ and then shuffles and decrypts k ElGamal ciphertexts $\{(g_i, m_i)\}_{i=1,\cdots,k}$ to $\{(g'_i, m'_i)\}_{i=1,\cdots,k}$ as

$$
\begin{aligned}
(g'_i, m'_i) &= (g_0{}^{A_{0i}} g_{\phi^{-1}(i)}, g_i'^{-x'} m_0{}^{A_{0i}} m_{\phi^{-1}(i)}) \bmod p \\
&= (\prod_{\nu=0}^{k} g_\nu{}^{A_{\nu i}}, g_i'^{-x'} \prod_{\nu=0}^{k} m_\nu{}^{A_{\nu i}}) \bmod p. \qquad (7)
\end{aligned}
$$

In our protocol, the witness W_n is a set $\{x', (A_{ji})_{i,j=1,\ldots,k}, \{A_{0i}\}_{i=1,\ldots,k}\}$, and the common input X_n is a set $\{p, q, g_0, y, m_0, F_n, (g_i, m_i)_{i=1,\ldots,k}, (g'_i, m'_i)_{i=1,\ldots,k}\}$. P is given X_n and W_n, and V is given X_n.

Proving a shuffle decryption **Commitment-1:** P uniformly and randomly chooses $\{A_{\nu 0}, A'_\nu \in_R \mathbb{Z}/q\mathbb{Z}\}_{\nu=-2,\ldots,k}$ and then computes:

$$
A_{-1i} = \sum_{j=1}^{k} 3A_{j0}A_{ji} \bmod q \ , \quad A_{-2i} = \sum_{j=1}^{k} 3{A_{j0}}^2 A_{ji} \bmod q \quad i = 1, \cdots, k
$$

$$
\begin{aligned}
f'_\mu &= \prod_{\nu=-2}^{k} f_\nu{}^{A_{\nu\mu}} \bmod p \quad \mu = 0, \cdots, k \\
\tilde{f}'_0 &= \prod_{\nu=-2}^{k} f_\nu{}^{A'_\nu} \bmod p \ , \quad g'_0 = \prod_{\nu=0}^{k} g_\nu{}^{A_{\nu 0}} \bmod p \\
m'_0 &= \prod_{\nu=0}^{k} m_\nu{}^{A_{\nu 0}} \bmod p \ , \quad w = \sum_{j=1}^{k} {A_{j0}}^3 - A_{-20} - A'_{-1} \bmod q
\end{aligned}
$$

Then, P sends $g'_0, m'_0, w, \tilde{f}'_0, \{f'_\mu\}_{\mu=0,\cdots,k}$to V as a commitment.
Challenge-1: V uniformly and randomly chooses $\{c_i\}_{i=1,\cdots,k}$ from $\mathbb{Z}/q\mathbb{Z}$ and sends it to P.

[6] r'_μ plays the role of λ' in [8].

Response-1: P sends V the following response:

$$r_\nu = \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \ , \quad r'_\nu = \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \bmod q \quad \nu = -2, \cdots, k$$

where $c_0 = 1 \bmod p$.

Commitment-2: P then computes

$$\zeta = \prod_{i=1}^{k} g_i'^{c_i} \bmod p.$$

P uniformly and randomly chooses $\beta \in_R \mathbb{Z}/q\mathbb{Z}$, computes the following commitment, and sends it to V:

$$\eta = \zeta^{x'} \bmod p \ , \quad \eta' = \zeta^{\beta} \bmod p$$
$$y' = {g_0}^{\beta} \bmod p. \tag{8}$$

Challenge-2: V uniformly and randomly chooses c' from $\mathbb{Z}/q\mathbb{Z}$ and sends it to P.

Response-2: P sends V the following response: $r' = c'x' + \beta \bmod q$

Verification: V computes

$$\zeta = \prod_{i=1}^{k} g_i'^{c_i} \bmod p.$$

V accepts the shuffle if the following equations hold for a uniformly and randomly generated $\alpha \in_R \mathbb{Z}/q\mathbb{Z}$:

$$\prod_{\nu=-2}^{k} {f_\nu}^{r_\nu + \alpha r'_\nu} = f'_0 \tilde{f}_0'^{\alpha} \prod_{i=1}^{k} f_i'^{c_i + \alpha {c_i}^2} \pmod p \tag{9}$$

$$\prod_{\nu=0}^{k} {g_\nu}^{r_\nu} = \zeta g'_0 \pmod p \tag{10}$$

$$\prod_{\nu=0}^{k} {m_\nu}^{r_\nu} = \eta \prod_{\mu=0}^{k} {m'_\mu}^{c_\mu} \pmod p \tag{11}$$

$$\sum_{j=1}^{k} (r_j^3 - c_j^3) = r_{-2} + r'_{-1} + w \pmod q \tag{12}$$

$${g_0}^{r'} = y^{c'} y' \pmod p \tag{13}$$

$$\zeta^{r'} = \eta^{c'} \eta' \pmod p \tag{14}$$

The view $View_V^P(X_n, W_n)$ of this protocol is

$$p, q, y, g_0, m_0, \{(g_i, m_i)\}_{i=1,\ldots,k}, \{(g'_i, m'_i)\}_{i=1,\ldots,k},$$
$$\{f_\nu\}_{\nu=-2,\ldots,k}, f'_0, \{f'_i\}_{i=1,\ldots,k}, \tilde{f}'_0, g'_0, m'_0, w, \{c_i\}_{i=1,\ldots,k},$$
$$\{r_\nu\}_{\nu=-2,\ldots,k}, \{r'_\nu\}_{\nu=-2,\ldots,k}, \eta, \eta', y', c', r'.$$

4.4 Properties of the proposed scheme

Theorem 3. *The protocol is complete.*

Theorem 4. *The protocol is special sound as long as the discrete logarithm problem is difficult to solve.*

Theorem 3 and 4 can be proved along the lines with [9]. Proof are given in the appendix of [7].

Theorem 5. *If the decision Diffie-Hellman problem is difficult to solve, the verifiable shuffle-decryption protocol* (P, V, G_R) *is special complete-permutation-hiding.*

Proof. The proof is given in the appendix of [7].

4.5 Threshold Decryption

Although it is possible to achieve threshold decryption with the proposed protocol, it does not work as well as ordinary threshold decryption. If we assume that only honest shufflers participate in the shuffle-decryption protocol, there is no disadvantage when using our protocol. However, if a malicious shuffler quits decryption after some other shufflers have finished their decryptions, our protocol gets into trouble.

Suppose we are decrypting or shuffle-decrypting k ElGamal cipher-texts, λ shufflers have finished their partial decryptions, and one shuffler quits its decryption procedure. In the ordinary threshold decryption protocol, the rest of the shufflers and one substituting (new) shuffler are able to continue the threshold decryption protocol only with little modification. However, computation of $k\lambda$ extra modular exponentiations is required to complete the decryption, and the verifier must compute $k\lambda$ extra modular exponentiations to verify the correctness of the decryption.

In our protocol, shufflers that have finished their partial decryptions need to help other players complete the protocol. Each of the shufflers needs to compute k modular exponentiations to modify the cipher-texts that are already shuffle-decrypted by λ shufflers. Each of them needs to prove the correctness of the above computation which requires another computation of k modular exponentiations. Moreover, the verifier needs to compute an extra $2k\lambda$ modular exponentiations to verify the correctness of the protocol.

5 Efficiency

In this section, we compare the efficiency of the proposed protocol described in Section 4 to (FS) the protocol proposed in [9], (FMMOS) the protocol proposed in [8], and (Groth) the protocol proposed in [13]. We have assumed the lengths of p and q to be 1024 and 160. We have denoted the protocol in Section 4 as (proposed).

Let us first compare them, in Table 1, by the number of exponentiations used in each protocol when the number of ciphertexts is k. "shuffle P" and "shuffle V" denote the number of exponentiations required for P and V to prove and verify a shuffle. "shuffle-decrypt P" and "shuffle-decrypt V" denote the number of exponentiations required for P and V to prove and verify a shuffle-decryption. The numbers for $(FS), (FMMOS)$, and $(Groth)$ are those required to prove a shuffle-decryption in a standard technique

If we adopt the computation tools described in [16], such as the simultaneous multiple exponentiation algorithm and the fixed-base comb method, the number of exponentiations can be heuristically reduced. We estimated that multiple exponentiations cost a 1/3 and fixed-base comb method costs 1/12 (when the number of ciphertexts is large) of that of single exponentiation. Estimates done in this way are in Table 2. Here, "shuffle P", "shuffle V" , "shuffle-decrypt P", and "shuffle-decrypt V" denote the same items as in Table 1.

Table 3 lists the number of communication bits and number of rounds required for protocols. "shuffle" denotes the number of communication bits used when proving a shuffle, "shuffle-decrypt" denotes the number of communication bits used when proving a shuffle-decryption, and "rounds" denotes the number of rounds required for protocols. The numbers for $(FS), (FMMOS)$, and $(Groth)$ include intermediate state data bits, i.e., those of shuffled data.

	(FS)	(FMMOS)	(Groth)	(proposed)
shuffle P	$8k$		$6k$	
shuffle V	$10k$		$6k$	
shuffle-decrypt P	$(9k)$	$9k$	$(7k)$	$8k$
shuffle-decrypt V	$(12k)$	$10k$	$(8k)$	$6k$

Table 1. Numbers of exponentiations required in each protocol

Our protocol and the protocols of [9, 8] require a rather long public parameter F_n. Although the protocol of [13] also requires such a parameter, it can be reduced greatly at the cost of increasing the amount of both computation and communication.

From Tables 2 and 3, we can conclude that computational complexity with our proposed protocol represents a 32% improvement in efficiency over that of (Groth)[13], while communication complexity improves by 47%. Our protocol require two rounds less than that of Groth's [13].

	(FS)	(FMMOS)	(Groth)	(proposed)
shuffle P	$1.4k$		$1.75k$	
shuffle V	$3.3k$		$1.75k$	
shuffle-decrypt P	$(2.4k)$	$1.75k$	$(2.75k)$	$1.9k$
shuffle-decrypt V	$(4.5k)$	$3.3k$	$(3k)$	$2k$

Table 2. Cost of computation required in each protocol

	(FS)	(FMMOS)	(Groth)	(proposed)
shuffle	$5044k$		$1184k$	
shuffle-decrypt	$(6388k)$	$5044k$	$(2528k)$	$1344k$
rounds	3	5	7	5

Table 3. Communication bits required in each protocols

6 Conclusion

In this paper, I have proposed formal definition for the core requirement of unlinkability in verifiable shuffle-decryption. I have also presented a novel method of simultaneously proving both the correctness of both a shuffle and a decryption, and then have proved its security and demonstrated its superior efficiency over that of [13] and [8].

Acknowledgments

The author would like to thank Hiroaki Anada and Satoshi Obana for many helpful discussions.

References

1. M. Abe, *Mix-Networks on Permutation Networks*, Advances in Cryptology — ASIACRYPT '99, LNCS 1716, pp. 258-273, Springer-Verlag, (1999).
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, Vol.24, No.2, pp. 84-88, (1981).
4. R. Cramer, I. Damgård, and B. Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, Crypto '94, LNCS 839, pp. 174-187, (1994).
5. R. Cramer and V. Shoup, *A practical key cryptosystem provably secure against adaptive chosen ciphertext attack*, Advances in Cryptology — Crypto '98, LNCS 1462, pp. 13-25, (1998).
6. A. Fiat and A. Shamir. *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology — CRYPTO '86, LNCS 263, pp. 186–194, (1986).

7. J. Furukawa, *Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability (with appendixes)*, Available online, http://eprint.iacr.org, or from the author via e-mail.
8. J. Furukawa, K. Mori, S. Obana, and K. Sako, *An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling*, Financial Cryptography 2002.
9. J. Furukawa and K. Sako, *An Efficient Scheme for Proving a Shuffle*, Advances in Cryptology — CRYPTO 2001, LNCS 2139 pp. 368-387 (2001).
10. O. Goldreich, *A Uniform-Complexity Treatment of Encryption and Zero-Knowledge*, Journal of Cryptology, Vol. 6, pp. 21-53, (1993).
11. S. Goldwasser and S. Micali, *Probabilistic Encryption*, JCSS, Vol. 28, No. 2, pp. 270-299, (1984).
12. P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels, *Optimistic mixing for exit-polls*, Asiacrypt 2002, LNCS 2501, pp. 451-465 (2002)
13. J. Groth, *A Verifiable Secret Shuffle of Holomorphic Encryptions*, Public Key Cryptography – PKC 2003, LNCS 2567 pp. 145-160 (2003)
14. M. Jakobsson, *A practical mix*, Eurocrypt '98, LNCS 1403, pp. 448-461 (1998)
15. A. Juels and M. Jakobsson, *An optimally robust hybrid mix network*, Proc. of the 20th annual ACM Symposium on Principles of Distributed Computation, 2001
16. A. Menezes, C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, pp. 617-627, (1997).
17. C.A. Neff, *A Verifiable Secret Shuffle and its Application to E-Voting*, ACMCCS 01 pp. 116-125 (2001).
18. M. Ohkubo and M .Abe, *A length-invariant hybrid mix* Asiacrypt 2000, LNCS 1976, pp. 178-191 (2000)
19. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani, *Fault tolerant anonymous channel*, ICICS, LNCS 1334, pp. 440-444 (1997).
20. K. Sako, *Electronic voting schemes allowing open objection to the tally*, Transactions of IEICE, Vol. E77-A No. 1, Jan. (1994).
21. K. Sako and J. Kilian, *Receipt-free mix-type voting scheme –A practical solution to the implementation of voting booth*, Eurocrypt '95, LNCS 921, pp. 393-403 (1995).
22. C. P. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, 4, pp. 161–174, (1991).