An Efficient Scheme for Proving a Shuffle

FURUKAWA Jun and Kazue Sako

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

Abstract. In this paper, we propose a novel and efficient protocol for proving the correctness of a shuffle, without leaking how the shuffle was performed. Using this protocol, we can prove the correctness of a shuffle of n data with roughly $18n$ exponentiations, where as the protocol of Sako-Kilian[SK95] required $642n$ and that of Abe[Ab99] required $22n \log n$. The length of proof will be only $2^{11}n$ bits in our protocol, opposed to $2^{18}n$ bits and $2^{14}n \log n$ bits required by Sako-Kilian and Abe, respectively. The proposed protocol will be a building block of an efficient, universally verifiable mix-net, whose application to voting system is prominent.

Keywords: Mix-net, Permutation, Electronic Voting, Universal Verifiability

1 Introduction

A mix-net[Ch81] scheme is useful for applications which require anonymity, such as voting. The core technique in a mix-net scheme is to execute multiple rounds of shuffling and decryption by multiple, independent mixers, so that the output decryption can not be linked to any of the input encryptions.

To ensure the correctness of the output, it is desirable to achieve the property of universal verifiability. However, proving the correctness of a shuffle without sacrificing unlinkability required a large amount of computation in the prior art. For example, [SK95] adopted a cut-and-choose method to prove the correctness. Abe[Ab99] took an approach to represent a shuffle using multiple pairwise permutations [1]. In practical terms, however, neither scheme is efficient enough to handle a large number of ciphertexts, say on the order of 10,000.

This paper proposes a novel, efficient scheme for proving the correctness of a shuffle. We take a completely different approach than that of [SK95] and [Ab99]. We represent a permutation by a matrix, and introduce two conditions which suffice to achieve a permutation matrix. We then present zero-knowledge proofs to prove the satisfiability of each condition. Moreover, these two proofs can be merged into one proof, resulting in a very efficient proof of a correct shuffle.

We also present here an analysis of the efficiency of our proof. Our proof requires roughly $18n$ exponentiations to prove the correctness of a n-data shuffle,

[1] Another approach, based on a verifiable secret exponent multiplication is described in [Ne01].

where as the protocol of Sako-Kilian[SK95] required $642n$ and that of Abe[Ab99] required $22n \log n$. Using the computation tools in [HAC], the total computation cost necessary in our proof can be reduced to an equivalent of $5n$ exponentiations. The length of a proof will be only $2^{11}n$ bits in our protocol, opposed to $2^{18}n$ bits and $2^{14}n \log n$ bits required by Sako-Kilian and Abe, respectively.

Our paper is organized in the following way. In Section 2, we present the two conditions on a permutation matrix. In Section 4 we give zero-knowledge proofs for each of the two conditions, and discuss how these proofs are combined to achieve to prove the whole shuffle. In Section 5 we describe our protocol and in Section 6, we compare the efficiency of our protocol to prior work.

2 Basic Idea

2.1 Shuffling

Informally speaking, a shuffling is a procedure which on input of n ciphertexts $(E_1, E_2, \ldots, E_n)$, outputs n ciphertexts $(E'_1, E'_2, \ldots, E'_n)$ where:

- there exists a permutation ϕ s.t $D(E'_i) = D(E_{\phi^{-1}(i)})$ for all i. Here, D is a decryption algorithm for ciphertexts.
- Without the knowledge of D or ϕ, $(E_1, E_2, \ldots, E_n)$, and $(E'_1, E'_2, \ldots, E'_n)$ reveal no information on the permutation ϕ.

We consider the use of ElGamal cryptosystems, with public keys (p, q, g, y) and secret key $X \in \mathbf{Z_q}$ s.t. $y = g^X \bmod p$. [2]

Given n ciphertexts $\{E_i\} = \{(g_i, m_i)\}$, where all $\{g_i\}$ and $\{m_i\}$ have the order q, shuffled ciphertexts $\{E'_i\} = \{(g'_i, m'_i)\}$ can be obtained by

$$\begin{aligned} g'_i &= g^{r_i} \cdot g_{\phi^{-1}(i)} \bmod p \\ m'_i &= y^{r_i} \cdot m_{\phi^{-1}(i)} \bmod p \end{aligned} \tag{1}$$

using randomly generated $\{r_i\}$.

2.2 Permutation Matrix

We define a matrix (A_{ij}) to be a permutation matrix if it can be written as follow using some permutation function ϕ.

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

Using this permutation matrix, the equation (1) is equivalent to

$$(g'_i, m'_i) = (g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}}, y^{r_i} \prod_{j=1}^{n} m_j^{A_{ji}}) \bmod p. \tag{2}$$

In order to prove the correctness of the shuffle, we need to show the following two things.

[2] We assume, as usual, p and q are two primes s.t. $p = kq + 1$, where k is an integer, and g is an element that generates a subgroup $\mathbf{G_q}$ of order q in $\mathbf{Z_p^*}$.

1. For each pair $\{(g'_i, m'_i)\}$, the same r_i and (A_{ij}) has been used.
2. (A_{ij}) used is a permutation matrix.

The first property can be efficiently shown using a standard technique[Br93]. The contribution of this paper is to present a novel technique to prove the second property.

At first, we concentrate on proving the existence of a permutation matrix (A_{ij}) and $\{r_i\}$ when given $\{g_i\}$ and $\{g'_i\}$, s.t.

$$g'_i = g^{r_i} \prod_{j=1}^{n} {g_j}^{A_{ji}} \bmod p. \tag{3}$$

We thus need to prove the existence of such a permutation matrix. We begin by looking at necessary conditions which suffices to achieve a permutation matrix. The following is the key observation used to construct the proposed protocol.

Theorem 1. *A matrix $(A_{ij})_{(i,j=1,\ldots,n)}$ is a permutation matrix if and only if, for all i, j, and k, both*

$$\sum_{h=1}^{n} A_{hi}A_{hj} = \begin{cases} 1 \bmod q, & \text{if } i = j \\ 0 \bmod q, & \text{if } i \neq j \end{cases} \tag{4}$$

$$\sum_{h=1}^{n} A_{hi}A_{hj}A_{hk} = \begin{cases} 1 \bmod q & \text{if } i = j = k \\ 0 \bmod q & \text{if otherwise} \end{cases} \tag{5}$$

hold.

Notation 1 *For convenience, we define δ_{ij} and $\delta_{ijk}(i, j, k = 1, \ldots, n)$ to be, respectively,*

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{and} \quad \delta_{ijk} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{if otherwise.} \end{cases}$$

Proof. We first show that there is exactly one non-zero element in each row vector of (A_{ij}) and then, the same for each column vector.

Let C_i be a i-th column vector of the matrix $(A_{ij})_{(i,j=1,\ldots,n)}$. Then, from Equation (4), we see $(C_i, C_j) = \delta_{ij}$ where (A, B) is inner product of vectors A and B. This implies that $rank(A_{ij}) = n$, that is, there is at least one non-zero element in each row and each column. Next we consider a vector $C_i \odot C_j (i \neq j)$ where the operator $\odot$ is defined as $(a_1 \ldots a_n) \odot (b_1 \ldots b_n) = (a_1 b_1 \ldots a_n b_n)$. Define a vector $\hat{C} = \sum_{l=0}^{n} \kappa_l C_l$ for an arbitrary κ_l. From the fact $(\hat{C}, C_i \odot C_j) = \sum_{l=1} \kappa_l \delta_{lij} = 0$ and linear combinations of $\{C_l\}$ generate the space ${\mathbf{Z_q}}^n$, we obtain $C_i \odot C_j = \mathbf{0}$. This means for any h, i and j s.t. $i \neq j$, either $A_{hi} = 0$ or $A_{hj} = 0$. Therefore, the number of non-zero elements in each row vector of (A_{ij}) is at most 1, and thus exactly 1.

From the above observations, the matrix (A_{ij}) contains exactly n non-zero elements. Since $C_i \neq \mathbf{0}$ for all i, the number of non-zero element in each column

vector is also 1. Thus, there is exactly one non-zero element in each row vector and each column vector of the matrix (A_{ij}) if Equations (4) and (5) hold.

The unique non-zero element e_i in $i-th$ row must be $e_i^2 = 1 \bmod q$ from Equation (4) and $e_i^3 = 1 \bmod q$ from Equation (5). This leads to $e_i = 1$ and that matrix $(A_{ij})_{(i,j=1,\ldots,n)}$ is a permutation matrix over $\mathbf{Z_q}$.

2.3 Outline of main protocol

Using Theorem 1, the main protocol can be constructed by the following proofs

Proof-1 a proof that given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as eq.(3) using integers $\{r_i\}$ and a matrix that satisfies the first condition.

Proof-2 a proof that given $\{g_i\}$ and $\{g'_i\}$, $\{g'_i\}$ can be expressed as eq.(3) using integers $\{r_i\}$ and a matrix that satisfies the second condition.

Proof-3 a proof that integers $\{r_i\}$ and the matrix used in the above two proofs are identical.

Proof-4 For each pair (g'_i, m'_i), the same r_i and $\{A_{ij}\}$ has been used.

In the Section 4, we provide protocols for **Proof-1** and **Proof-2**.

3 Security of the protocol

We will prove that the main protocol is sound and zero-knowledge under computational assumption. More specifically, for the property of soundness, we can claim that if a verifier accepts the protocol, then either prover knows the permutation or he knows integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability. For the zero-knowledge property, we can construct a simulator and claim that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then this distinguisher can be used to solve the decisional Diffie-Hellman problem. We note that to make a shuffle secret, we already assume the hardness of the decisional Diffie-Hellman problem.

In the course of reduction, we use the following arguments. First, we define the following set.

Definition 1. *Define R_n^m to be the set of tuples of $n \times m$ elements in $\mathbf{G_q}$:*

$$I = (x_1^{(1)}, \ldots, x_1^{(m)}, x_2^{(1)}, \ldots, x_2^{(m)}, \ldots, x_n^{(1)}, \ldots, x_n^{(m)}).$$

We then define the subset D_n^m of R_n^m to be the set of tuples I satisfying

$$\log_{x_1^{(1)}} x_1^{(i)} = log_{x_j^{(1)}} x_j^{(i)} \bmod p$$

for all $i(i = 2, 3, .., m)$ and $j(j = 2, ..., n)$.

Definition 2. *We define the problem of distinguishing instances uniformly chosen from R_n^m and those from D_n^m by DDH_n^m.*

Note that the decisional Diffie-Hellman problem can be denoted as DDH_2^2. We claim that for any n and m the difficulty of DDH_n^m equals to the decisional Diffie-Hellman problem, by proving the following.

Lemma 1. *For any $n(\geq 2)$ and $m(\geq 2)$, if DDH_n^m is easy, then DDH_n^2 is easy.*

Lemma 2. *If for any $n(\geq 2)$, DDH_n^2 is easy then the decisional Diffie-Hellman is easy.*

Proofs for Lemma 1 and 2 are sketched in Appendix A

4 Proof-1 and Proof-2

In this section, we give two proofs that will be the building blocks of the main protocol.

4.1 Proving the first condition (Proof-1)

The following protocol proves that given $\{g_i\}$ and $\{g'_i\}$, the prover knows $\{r_i\}$ and $\{A_{ij}\}$ s.t.

$$g'_i = g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}} \bmod p$$

$$\sum_{h=1}^{n} A_{hi} A_{hj} = \delta_{ij} \bmod q.$$

The main idea is to issue $s = \sum_{j=1}^{n} r_j c_j$ and $s_i = \sum_{j=1}^{n} A_{ij} c_j$ as a response to a challenge $\{c_j\}$ and let the verifier check

$$\sum_{i=1}^{n} {s_i}^2 = \sum_{j=1}^{n} {c_j}^2 \bmod q$$

$$g^s \prod_{i=1}^{n} {g_i}^{s_i} = \prod_{j=1}^{n} {g'_j}^{c_j} \bmod p.$$

However, this apparently leaks information on A_{ij}, so we need to add randomizers and commitments. By making the response $s = \sum_{j=1}^{n} r_j c_j + \alpha$ and $s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i$ using randomizers $\{\alpha_i\}$ and α, a verifier needs to check the following equation:

$$\sum_{i=1}^{n} {s_i}^2 = \sum_{j=1}^{n} {c_j}^2 + \sum_{j=1}^{n} B_j c_j + D \bmod q$$

where B_j and D are quadratic polynomials of $\{A_{ij}\}$ and α_i. Therefore these B_j and D, together with $g^\alpha \prod_{i=1}^{n} g_i^{\alpha_i}$, will be also sent in advance to enable

verification. We further add another randomizer σ and modify the verification equation to be

$$\sum_{i=1}^{n} {s_i}^2 + \sigma s = \sum_{j=1}^{n} {c_j}^2 + \sum_{j=1}^{n} (B_j + \sigma r_j) c_j + (D + \sigma\alpha) \bmod q.$$

In order to hide the actual value of $\sigma, \{B_j + \sigma r_j\}$ and $D + \sigma\alpha$, this verification is computed over exponents. The below gives a complete description of the **Proof-1**.

<u>Proof-1</u>

Input:$p, q, g, \{g_i\}, \{g'_i\}$.

1. Prover ($\mathcal{P}$) generates random numbers $\sigma, \alpha, \{\alpha_i\}_{(i=1,\ldots,n)} \in_R \mathbf{Z_q}$ and computes

$$\begin{aligned} w &= g^{\sigma} \bmod p \\ g' &= g^{\alpha} \prod_{j=1}^{n} {g_j}^{\alpha_j} \bmod p \\ \dot{w}_i &= g^{\sum_{j=1}^{n} 2\alpha_j A_{ji} + \sigma r_i} \quad (= g^{B_i + \sigma r_i}) \bmod p \quad i = 1, \ldots, n \\ \dot{w} &= g^{\sum_{j=1}^{n} {\alpha_j}^2 + \sigma\alpha} \quad (= g^{D+\sigma\alpha}) \bmod p \end{aligned} \tag{6}$$

   and sends $w, g', \{\dot{w}_i\}, \dot{w}$ $(i = 1, \ldots, n)$ to $\mathcal{V}$.
2. $\mathcal{V}$ sends back randomly chosen $\{c_i\}_{(i=1,\ldots,n)} \in_R \mathbf{Z_q}$ as a challenge.
3. $\mathcal{P}$ computes $s = \sum_{j=1}^{n} r_j c_j + \alpha \bmod q$
   and $s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q (i = 1, \ldots, n)$ and sends to $\mathcal{V}$.
4. $\mathcal{V}$ verifies the following:

$$g^s \prod_{j=1}^{n} {g_j}^{s_j} = g' \prod_{j=1}^{n} {g'_j}^{c_j} \bmod p \tag{7}$$

$$w^s g^{\sum_{j=1}^{n} (s_j^2 - c_j^2)} = \dot{w} \prod_{j=1}^{n} {\dot{w}_j}^{c_j} \bmod p \tag{8}$$

.

<u>Properties of Proof-1</u>

Theorem 2. **Proof-1** *is complete. That is, if $\mathcal{P}$ knows $\{r_i\}$ and $\{A_{ij}\}$ satisfying the first condition, $\mathcal{V}$ always accepts.*

Theorem 3. *If $\mathcal{V}$ accepts* **Proof-1** *with a non-negligible probability, then $\mathcal{P}$ either knows both $\{r_i\}$ and $\{A_{ij}\}$ satisfying the first condition, or can generate integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

A sketch of Proof: Theorem 3 can be proved from the following lemmas, proofs of which are sketched in Appendix B.

Lemma 3. *If $\mathcal{V}$ accepts* **Proof-1** *with non-negligible probability, then $\mathcal{P}$ knows $\{A_{ij}\}$, $\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6).*

Lemma 4. *Assume $\mathcal{P}$ knows $\{A_{ij}\}$, $\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6). If $\mathcal{P}$ knows $\{s_i\}$ and s which satisfy Equation (7), and either $s \neq \sum_{j=1}^{n} r_j c_j + \alpha$ or $s_i \neq \sum_{j=1}^{n} A_{ij} c_j + \alpha_i$ for some i hold, then $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

Lemma 5. *Assume $\mathcal{P}$ knows $\{A_{ij}\}$, $\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6). If Equations (7) and (8) hold with non-negligible probability, then either Equation (4) hold or $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

□

Theorem 4. *We can construct a simulator of* **Proof-1** *such that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then we can solve the decisional Diffie-Hellman problem.*

A sketch of Proof: Given in Appendix B. □

4.2 Proving the second condition(Proof-2)

Analogous to **Proof-1**, the proof for the fact the prover knows $\{r_i\}$ and $\{A_{ij}\}$ s.t.

$$g'_i = g^{r_i} \prod_j g_j^{A_{ji}} \bmod p$$

$$\sum_{h=1}^{n} A_{hi} A_{hj} A_{hk} = \delta_{ijk} \bmod q$$

for $\{g_i\}$ and $\{g'_i\}$, is given as **Proof-2**.

<u>Proof-2</u>

Input:$p, q, g, \{g_i\}, \{g'_i\}$.

1. Prover ($\mathcal{P}$) generates random numbers $\rho, \tau, \alpha, \{\alpha_i\}, \lambda, \{\lambda_i\} \in_R \mathbf{Z_q}$ $(i = 1, \ldots, n)$ and computes

$$t = g^{\tau}, v = g^{\rho}, u = g^{\lambda}, u_i = g^{\lambda_i} \bmod p \quad i = 1, \ldots, n$$

$$g' = g^{\alpha} \prod_{j=1}^{n} {g_j}^{\alpha_j} \bmod p$$

$$\dot{t}_i = g^{\sum_{j=1}^{n} 3\alpha_j A_{ji} + \tau\lambda_i} \bmod p \quad i = 1, \ldots, n$$
$$\dot{v}_i = g^{\sum_{j=1}^{n} 3{\alpha_j}^2 A_{ji} + \rho r_i} \bmod p \quad i = 1, \ldots, n$$
$$\dot{v} = g^{\sum_{j=1}^{n} {\alpha_j}^3 + \tau\lambda + \rho\alpha} \bmod p$$

and sends $t, v, u, \{u_i\}, g', \{\dot{t}_i\}, \{\dot{v}_i\}, \dot{v}$ $(i = 1, \ldots, n)$, to $\mathcal{V}$.

2. $\mathcal{V}$ sends back randomly chosen $\{c_i\}_{(i=1,\ldots,n)} \in_R \mathbf{Z_q}$ as challenge.
3. $\mathcal{P}$ computes $s = \sum_{j=1}^{n} r_j c_j + \alpha \bmod q$, $s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q (i = 1, \ldots, n)$, and $\lambda' = \sum_{j=1}^{n} \lambda_j c_j^2 + \lambda \bmod q$ and sends to $\mathcal{V}$.
4. $\mathcal{V}$ verifies the following:

$$g^s \prod_{j=1}^{n} {g_j}^{s_j} = g' \prod_{j=1}^{n} {g'_j}^{c_j} \bmod p$$
$$g^{\lambda'} = u \prod_{j=1}^{n} {u_j}^{{c_j}^2} \bmod p$$
$$t^{\lambda'} v^s g^{\sum_{j=1}^{n} (s_j^3 - c_j^3)} = \dot{v} \prod_{j=1}^{n} {\dot{v}_j}^{c_j} {\dot{t}_j}^{c_j^2} \bmod p$$

Properties of Proof-2

We claim the following properties of **Proof-2**, which can be proved analogously to that of **Proof-1**.

Theorem 5. **Proof-2** *is complete. That is, if $\mathcal{P}$ knows $\{r_i\}$ and $\{A_{ij}\}$ satisfying the second condition, $\mathcal{V}$ always accepts.*

Theorem 6. *If $\mathcal{V}$ accepts* **Proof-2** *with a non-negligible probability, then $\mathcal{P}$ either knows both $\{r_i\}$ and $\{A_{ij}\}$ satisfying the second condition, or can generate integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

Theorem 7. *We can construct a simulator of* **Proof-2** *such that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then we can solve the decisional Diffie-Hellman problem.*

4.3 Constructing the Main Protocol

In this subsection, we explain how our main protocol is constructed using these **proof-1** and **Proof-2**. It should be noted, that these proofs did not have the ordinary soundness property. That is, a prover knowing integers satisfying $g^a \prod_{i=1}^{n} g_i^{a_i} = 1$ can deceive verifiers as if he had shuffled correctly. Since $\{g_i\}$ is originally chosen by those who encrypted the messages, there is no control to assure that the prover does not know the relations among them. Therefore, we fix a set of basis $\{\tilde{g}, \tilde{g}_1, \ldots \tilde{g}_n\}$ independent from the input ciphertexts, in a way we can assure the relations among the basis unknown. In fact, under

Discrete Logarithm Assumption, we can make it computationally infeasible to obtain such $\{a_i\}$ and a if we generate $\{\tilde{g}, \tilde{g}_1, \ldots \tilde{g}_n\}$ randomly[Br93]. This way it also suffices the requirement that the verifier should not know $\log_g \tilde{g}$ for zero-knowledge property.

We require the prover to perform the same permutation on the set of fixed basis as he did on the input ciphertexts. The prover proves that the permutation on the fixed basis $\{\tilde{g}, \tilde{g}_1, \ldots \tilde{g}_n\}$ is indeed a permutation, and that he indeed applied the same permutation to the input ciphertext.

Using the above methodology, we need not provide **Proof-3** described in the Subsection 2.3. If a prover knows two different representations of an element using $\{\tilde{g}, \tilde{g}_1, \ldots \tilde{g}_n\}$, it means that he knows the relations among the base $\{\tilde{g}, \tilde{g}_1, \ldots \tilde{g}_n\}$ which is against the assumption. **Proof-4** is achieved using the standard techniques described in [Br93]. Therefore we are now equipped with building blocks to prove the correctness of a shuffle.

5 The Main Protocol

In the previous subsection we illustrated our protocol as a combination of **proof-1** and **proof-2**, mainly for comprehensiveness. The proofs can be executed in parallel, resulting in a three-round protocol with reduced communication complexity.

<u>Main Protocol</u>

Input:$p, q, g, y, \tilde{g}$, $\{\tilde{g}_i\}$, $\{(g_i, m_i)\}$, $\{(g'_i, m'_i)\}$.

1. Prover ($\mathcal{P}$) generates the following random numbers:
   $\sigma, \rho, \tau, \alpha, \alpha_i, \lambda, \lambda_i \in_R \mathbf{Z}_q$ $(i = 1, \ldots, n)$
2. $\mathcal{P}$ computes the following:

$$t = g^\tau, v = g^\rho, w = g^\sigma, u = g^\lambda, u_i = g^{\lambda_i} \bmod p \quad i = 1, \ldots, n$$

$$\tilde{g}'_i = \tilde{g}^{r_i} \prod_{j=1}^{n} \tilde{g}_j{}^{A_{ji}} \bmod p \quad i = 1, \ldots, n \tag{9}$$

$$\tilde{g}' = \tilde{g}^\alpha \prod_{j=1}^{n} \tilde{g}_j{}^{\alpha_j} \bmod p \tag{10}$$

$$g' = g^\alpha \prod_{j=1}^{n} g_j{}^{\alpha_j} \bmod p$$

$$m' = y^\alpha \prod_{j=1}^{n} m_j{}^{\alpha_j} \bmod p$$

$$\dot{t}_i = g^{\sum_{j=1}^{n} 3\alpha_j A_{ji} + \tau\lambda_i} \bmod p \quad i = 1, \ldots, n$$

$$\dot{v}_i = g^{\sum_{j=1}^{n} 3\alpha_j{}^2 A_{ji} + \rho r_i} \bmod p \quad i = 1, \ldots, n$$

$$\dot{v} = g^{\sum_{j=1}^{n} {\alpha_j}^3 + \tau\lambda + \rho\alpha} \bmod p$$
$$\dot{w}_i = g^{\sum_{j=1}^{n} 2\alpha_j A_{ji} + \sigma r_i} \bmod p \quad i = 1, \ldots, n$$
$$\dot{w} = g^{\sum_{j=1}^{n} {\alpha_j}^2 + \sigma\alpha} \bmod p.$$

3. $\mathcal{P}$ sends the following to the verifier $\mathcal{V}$:
   $t, v, w, u, \{u_i\}, \{\tilde{g}'_i\}, \tilde{g}', g', m', \{\dot{t}_i\}, \{\dot{v}_i\}, \dot{v}, \{\dot{w}_i\}, \dot{w}$ $(i = 1, \ldots, n)$.
4. $\mathcal{V}$ sends back randomly chosen $\{c_i\}_{(i=1,\ldots,n)} \in_R \mathbf{Z_q}$ as a challenge.
5. $\mathcal{P}$ computes the following and sends them to $\mathcal{V}$.

$$s = \sum_{j=1}^{n} r_j c_j + \alpha, \quad s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \mod q \quad i = 1, \ldots, n$$
$$\lambda' = \sum_{j=1}^{n} \lambda_j c_j^2 + \lambda \bmod q$$

6. $\mathcal{V}$ verifies the following:

$$\tilde{g}^s \prod_{j=1}^{n} {\tilde{g}_j}^{s_j} = \tilde{g}' \prod_{j=1}^{n} {\tilde{g}'_j}^{c_j} \bmod p \tag{11}$$
$$g^s \prod_{j=1}^{n} {g_j}^{s_j} = g' \prod_{j=1}^{n} {g'_j}^{c_j} \bmod p \tag{12}$$
$$y^s \prod_{j=1}^{n} {m_j}^{s_j} = m' \prod_{j=1}^{n} {m'_j}^{c_j} \bmod p \tag{13}$$
$$g^{\lambda'} = u \prod_{j=1}^{n} {u_j}^{{c_j}^2} \bmod p \tag{14}$$
$$t^{\lambda'} v^s g^{\sum_{j=1}^{n} (s_j^3 - c_j^3)} = \dot{v} \prod_{j=1}^{n} {\dot{v}_j}^{c_j} {\dot{t}_j}^{c_j^2} \bmod p \tag{15}$$
$$w^s g^{\sum_{j=1}^{n} (s_j^2 - c_j^2)} = \dot{w} \prod_{j=1}^{n} {\dot{w}_j}^{c_j} \bmod p \tag{16}$$

Theorem 8. *Main Protocol is complete. That is, if $\mathcal{P}$ knows $\{r_i\}$ and $\{A_{ij}\}$ satisfying the both conditions of Theorem 1, $\mathcal{V}$ always accepts.*

Theorem 9. *If $\mathcal{V}$ accepts Main Protocol with a non-negligible probability, then $\mathcal{P}$ knows $\{r_i\}$ and permutation matrix (A_{ij}) satisfying Equations (2), or can generate non-trivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} {\tilde{g}_i}^{a_i} = 1$ with overwhelming probability.*

Theorem 10. *We can construct a simulator of Main Protocol such that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then we can solve the decisional Diffie-Hellman problem.*

Proofs for Theorem 9 and 10 are sketched in Appendix C.

6 Discussions

In this section, we compare the efficiency of the proposed protocol described in Section 5 to the SK95 protocol in [SK95] and MiP-2 protocol in [Ab99]. To enable a fair comparison, we assume the security parameter of [SK95] to be 160 and lengths of p and q to be 1024 and 160 respectively.

We first compare them by the number of exponentiations used in each protocol, in the case of shuffling n ciphertexts. These are $22(n \log n - n + 1)$ for Abe's protocol, $642n$ for the SK95 protocol, and $18n + 18$ for the proposed protocol. If we adopt computation tools described in [HAC], such as the simultaneous multiple exponentiation algorithm and the fixed-base comb method, the number of exponentiations can be heuristically reduced to $11.2(n \log n - n + 1)$,$64n$, and $4.84n + 4.5$, respectively. The total number of bits needing to be transfered during the protocols is $13,248(n \log n - n + 1)$,$353,280n$, and $5,280n + 13,792$. The rounded-up numbers are shown in Table 1.

	Abe (MiP-2)	SK95	This Paper
No. exponentiations	$22n \log n$	$642n$	$18n$
(heuristically adjusted)	$11n \log n$	$64n$	$5n$
No. communication bits	$2^{14} n \log n$	$2^{18} n$	$2^{11} n$

Table 1. Comparison of three protocols

7 Conclusion

In this paper, we presented a novel method to prove the correctness of a shuffle, and demonstrated its efficiency. The proposed method requires only $18n$ exponentiations for shuffling n ciphertexts, where as previous methods required 35 times more, or required a higher order, $O(n \log n)$.

The proposed protocol can be used to build an efficient, universally verifiable voting system where the number of voters can scale up to the order of 10,000.

Acknowledgments

The authors would like to thank Tatsuaki Okamoto, Masayuki Abe, and Satoshi Obana for many helpful discussions.

References

[Ab99] M. Abe, *Mix-Networks on Permutation Networks*, Asiacrypt '99, LNCS 1716, 258-273 (1999)

[Br93] S. Brands, *An Efficient Off-line Electronic Cash System Based On The Representation Problem*, CWI Technical Report CS-R9323, (1993)

[Ch81] D. Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, Vol.24, No.2 84-88 (1981)

[CDS94] R. Cramer, I. Damgård and B. Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, Crypto '94, LNCS 839, 174-187 (1994)

[HAC] A. Menezes, C. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 617-619

[Ne01] C.A. Neff, *Verifiable, Secret Shuffles of ElGamal Encrypted Data*, Initial version circulated Mar. 2000, current version submitted to ACMCCS 01

[OKST97] W. Ogata, K. Kurosawa, K. Sako and K. Takatani, *Fault tolerant anonymous channel*, 1st International Conference on Information and Communications Security (ICICS), LNCS 1334, 440-444 (1997)

[SK95] K. Sako and J. Kilian, *Receipt-free mix-type voting scheme –A practical solution to the implementation of voting booth*, Eurocrypt 95, LNCS 921, 393-403 (1995)

A DDH_n^m and DDH

Lemma 1. *For any $m(\geq 2)$ and $n(\geq 2)$, if DDH_n^m is easy, then DDH_n^2 is easy.*

Proof. We claim that if DDH_n^m is easy, then either DDH_n^{m-1} is easy or DDH_n^2 is easy. By induction we can prove the correctness of the lemma.

In order to prove the claim, we define the subset M_n^m of R_n^m to be the set of tuples

$$I = (x_1^{(1)}, \ldots, x_1^{(m)}, x_2^{(1)}, \ldots, x_2^{(m)}, \ldots, x_n^{(1)}, \ldots, x_n^{(m)})$$

satisfying

$$\log_{x_1^{(1)}} x_1^{(i)} = log_{x_j^{(1)}} x_j^{(i)} \bmod p$$

for all $i(i = 2, 3, .., m-1)$ and $j(j = 2, ..., n)$, but whether or not

$$\log_{x_1^{(1)}} x_1^{(m)} = log_{x_j^{(1)}} x_j^{(m)} \bmod q$$

holds for all $j(j = 2, ..., n)$ is arbitrary. Therefore, the set D_n^m is a subset of M_n^m.

It is clear that if DDH_n^m is easy, then we can either distinguish between the instances chosen uniformly from R_n^m and M_n^m or the instances chosen uniformly from M_n^m and D_n^m. In the former case, it means DDH_n^{m-1} is easy. We claim in the following that in the latter case DDH_n^2 is easy.

Assume M_n^m and D_n^m are distinguishable. For any $I_n^2 \in R_n^2$ s.t.

$$I_n^2 = (x_1^{(1)}, x_1^{(2)}, x_2^{(1)}, x_2^{(2)}, \ldots, x_n^{(1)}, x_n^{(2)})$$

we transform it to $I_n^m \in R_n^m$

$$I_n^m = (x'^{(1)}_1, \ldots, x'^{(m)}_1, x'^{(1)}_2, \ldots, x'^{(m)}_2, \ldots, x'^{(1)}_n, \ldots, x'^{(m)}_n)$$

where

$$x'^{(i)}_j = \begin{cases} x^{(1)}_j & j = 1, \ldots, n \text{ (if } i = 1) \\ (x^{(1)}_j)^{z_i} \bmod p & j = 1, \ldots, n \text{ (if } 2 \le i \le m-1) \\ x^{(2)}_j & j = 1, \ldots, n \text{ (if } i = m) \end{cases}$$

with randomly chosen $\{z_i\}_{(i=2,\ldots,m-1)}$ in $\mathbf{Z_q}$.

If I^2_n is chosen uniformly from D^2_n, then I^m_n is distributed uniformly in D^m_n, and if I^2_n is chosen uniformly from R^2_n, then I^m_n is distributed uniformly in M^m_n. Therefore if D^m_n and M^m_n is distinguishable, then we can solve DDH^2_n.

Lemma 2. *If DDH^2_n $(n \ge 2)$is easy then the decisional Diffie-Hellman problem (DDH^2_2) is easy.*

Proof. For any $I^2_2 = (x^{(1)}_1, x^{(2)}_1, x^{(1)}_2, x^{(2)}_2) \in R^2_2$, we transform it to $I^2_n \in R^2_n$

$$I^2_n = (x'^{(1)}_1, x'^{(2)}_1, x'^{(1)}_2, x'^{(2)}_2, \ldots, x'^{(1)}_n, x'^{(2)}_n)$$

where

$$x'^{(1)}_1 = x^{(1)}_1,\ x'^{(2)}_1 = x^{(2)}_1,\ x'^{(1)}_2 = x^{(1)}_2,\ x'^{(2)}_2 = x^{(2)}_2$$
$$x'^{(1)}_j = (x^{(1)}_1)^{z_j} \cdot (x^{(1)}_2)^{w_j},\quad x'^{(2)}_j = (x^{(2)}_1)^{z_j} \cdot (x^{(2)}_2)^{w_j} \bmod p \quad j = 3, \ldots, n$$

with randomly chosen $\{z_j\}$ and $\{w_j\}(j = 3, \ldots, n)$ in $\mathbf{Z_q}$.

If I^2_2 is chosen uniformly from D^2_2, then I^2_n is distributed uniformly in D^2_n, and if I^2_2 is chosen uniformly from R^2_2, then I^2_n is distributed uniformly in R^2_n. Therefore if DDH^2_n is easy, then so is DDH^2_2.

B Properties of Proof-1

In this section, we sketch the proofs of the following theorems.

Theorem 3 (soundness). *If $\mathcal{V}$ accepts* **Proof-1** *with a non-negligible probability, then $\mathcal{P}$ either knows both $\{r_i\}$ and $\{A_{ij}\}$ satisfying the first condition, or can generate integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^n {g_i}^{a_i} = 1$ with overwhelming probability.*

Theorem 4 (zero-knowledge). *We can construct a simulator of* **Proof-1** *such that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then we can solve the decisional Diffie-Hellman problem.*

B.1 Soundness

It is clear that Theorem 3 holds if Lemmas 3, 4 and 5 hold. We therefore prove the lemmas.

Lemma 3. *If $\mathcal{V}$ accepts* **Proof-1** *with non-negligible probability, then $\mathcal{P}$ knows $\{A_{ij}\}$, $\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6).*

A sketch of Proof: Define $\mathcal{C}_p$ as the space which is spanned by the vector $(1, c_1, c_2, \ldots, c_n)$ made of the challenges to which $\mathcal{P}$ can compute responses $s, \{s_i\}_{(i=1,\ldots,n)}$ such that Equation (7) holds. If the $\dim(\mathcal{C}_p) = n+1$, $\mathcal{P}$ can choose $n+1$ challenges which are linearly independent and obtain $\{A_{ij}\}_{(i,j=1,\ldots,n)}$, $\{r_i\}_{(i=1,\ldots,n)}$, $\{\alpha_i\}_{(i=1,\ldots,n)}$, and α which satisfies the relation:

$$s = \sum_{j=1}^{n} r_j c_j + \alpha, \quad s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q \quad i = 1, \ldots, n$$

Such $\{A_{ij}\}$, $\{r_i\}$,$\{\alpha_i\}$, and α satisfies Equations (3) and (6) . If, $\dim(\mathcal{C}_p) < n+1$. The probability that $\mathcal{V}$ generates a challenge in $\mathcal{C}_p$ is at most $q^{n-1}/q^n = 1/q$, which is negligible. □

Lemma 4. *Assume $\mathcal{P}$ knows $\{A_{ij}\}$,$\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6). If $\mathcal{P}$ knows $\{s_i\}$ and s which satisfy Equation (7), and either $s \neq \sum_{j=1}^{n} r_j c_j + \alpha$ or $s_i \neq \sum_{j=1}^{n} A_{ij} c_j + \alpha_i$ for some i hold, then $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

Proof. The following gives a non-trivial representation of 1 using $g, \{g_i\}$.

$$g^{\sum_{j=1}^{n} s_j c_j + \alpha - s} \prod_{i=1}^{n} {g_i}^{\sum_{j=1}^{n} A_{ij} c_j + \alpha_i - s_i} = 1 \bmod p.$$

Lemma 5. *Assume $\mathcal{P}$ knows $\{A_{ij}\}$, $\{r_i\}$, $\{\alpha_i\}$, and α satisfying Equations (3) and (6). If Equations (7) and (8) hold with non-negligible probability, then either Equation (4) hold or $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability.*

A sketch of Proof: From Lemma 4, If Equation (7) holds, then either

$$\begin{cases} s = \sum_{j=1}^{n} r_j c_j + \alpha \bmod q \\ s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q \quad i = 1, \ldots, n \end{cases}$$

holds or $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $g^a \prod_{i=1}^{n} {g_i}^{a_i} = 1$ with overwhelming probability. We concentrate on the former case. If Equation (8) holds, then

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (\sum_{h=1}^{n} A_{hi} A_{hj} - \delta_{ij}) c_i c_j + \sum_{i=1}^{n} \left\{ (\sum_{j=1}^{n} 2\alpha_j A_{ji} + \sigma r_i) - \psi_i \right\} c_i$$
$$+ \left\{ (\sum_{j=1}^{n} \alpha_j^2 + \sigma\alpha) - \psi \right\} = 0 \quad \bmod q$$

where $\psi_i = \sum_{j=1}^{n} 2\alpha_j A_{ji} + \sigma r_i, \psi = \sum_{j=1}^{n} {\alpha_j}^2 + \sigma\alpha \bmod q$. If Equation (4) does not hold for some i and j, then the probability that Equation (8) holds is negligible. □

B.2 Zero-knowledge

A sketch of Proof: We first give a construction of the simulator. We then prove that if there exists such a distinguisher then we can solve DDH^2_{n+1}. From Lemma 2, it means it is equivalent to solving the decisional Diffie-Hellman problem.

The construction of the Simulator

We will construct the simulator $\mathcal{S}$ of the **Proof-1** with the input $p, q, g, \{g_i\}$, $\{g'_i\}$ as follows.

The simulator $\mathcal{S}$ first generates $s, \{s_i\}, \{c_i\} \in_R \mathbf{Z_q}$, $w, \{\dot{w}_i\} \in_R \mathbf{G_q}$ randomly. Then it computes $g', \dot{w}$ as the following.

$$g' = g^s \prod_{j=1}^{n} {g_j}^{s_j} {g'_j}^{-c_j} \bmod p$$

$$\dot{w} = w^s g^{\sum_{j=1}^{n}(s_j^2 - c_j^2)} \prod_{j=1}^{n} \dot{w}_j^{-c_j} \bmod p$$

The output of $\mathcal{S}$ is $(w, g', \{\dot{w}_i\}, \dot{w}, \{c_i\}, s, \{s_i\})$

A distinguisher of D^2_{n+1} and R^2_{n+1}

We will then construct a distinguisher $\mathcal{D}'$ who can distinguish between the uniform instances of D^2_{n+1} and R^2_{n+1} if $\mathcal{S}$ can not simulate the **Proof-1**.

Let's say the instance $I = (x_1^{(1)}, x_1^{(2)}, \ldots, x_{n+1}^{(1)}, x_{n+1}^{(2)})$ was chosen uniformly from either D^2_{n+1} or R^2_{n+1}. Then this distinguisher will first generate $g_1, g_2, ..g_n$ as the constants used in **Proof-1** and let $g = x_1^{(1)}$.

It will then generate a random permutation matrix (A_{ji}) and compute

$$g'_i = x_{i+1}^{(1)} \prod_{j=1}^{n} g_j^{A_{ji}} \bmod p. \quad (i = 1, \ldots, n)$$

We note that $\{g'_i\}$ gives a random permutation of $\{g_i\}$.

Based on $g, \{g_i\}, \{g'_i\}$, the distinguisher $\mathcal{D}'$ is going to act as a simulator $\mathcal{S}'$ which simulates the simulator $\mathcal{S}$. More specifically, the simulator $\mathcal{S}'$ randomly generates $s, \{s_i\}, \{c_i\} \in_R \mathbf{Z_q}$ and computes

$$w = x_1^{(2)}$$

$$g' = g^s \prod_{j=1}^{n} {g_j}^{s_j} {g'_j}^{-c_j} \bmod p$$

$$\alpha_j = s_j - \sum_{k=1}^{n} A_{jk} c_k \bmod q \quad j = 1, \ldots, n$$

$$\dot{w}_i = x_{i+1}^{(2)} \prod_{j=1}^{n} g^{2\alpha_j A_{ji}} \bmod p \quad i = 1, \ldots, n$$

$$\dot{w} = w^s g^{\sum_{j=1}^{n}(s_j^2 - c_j^2)} \prod_{j=1}^{n} \dot{w}_j^{-c_j} \bmod p.$$

The simulator $\mathcal{S}'$ outputs

$$w, g', \{\dot{w}_i\}, \dot{w}, \{c_i\}, s, \{s_i\}.(i = 1, \ldots, n)$$

Lemma 6. *Simulator $\mathcal{S}'$ perfectly simulates* **Proof-1** *when $I \in_R D_{n+1}^2$.*

Sketch: We let

$$\log_{x_1^{(1)}} x_{i+1}^{(1)} = r_i, \quad \log_{x_1^{(1)}} x_1^{(2)} = \sigma.$$

Then it is clear that by randomly choosing $\{s_i\}$ and s, it gives the same distribution of the output as when $\{\alpha_i\}$ and α were first chosen randomly, and verifier honestly chooses random challenge $\{c_i\}$.

Lemma 7. *Simulator $\mathcal{S}'$ perfectly simulates $\mathcal{S}$ when $I \in_R R_{n+1}^2$.*

Since $\{x_i^{(2)}\}_{(i=1,\ldots,n+1)}$ are randomly chosen, it gives the same distribution when $\dot{w}_i$ and w are randomly chosen.

□

Therefore, if there exists a distinguisher $\mathcal{D}$ that distinguishes the output of the simulator $\mathcal{S}$ and a real transcript of **Proof-1**, then this distinguisher can be used to solve DDH_{n+1}^2.

C Properties of the Main Protocol

In this section, we discuss the properties of the main protocol. The completeness property is clear. We provide proofs for the soundness and the zero-knowledge property.

C.1 Soundness

Theorem 9. *If $\mathcal{V}$ accepts Main Protocol with a non-negligible probability, then $\mathcal{P}$ knows $\{r_i\}$ and permutation matrix (A_{ij}) satisfying Equations (2), or can generate non-trivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} \tilde{g}_i{}^{a_i} = 1$ with overwhelming probability.*

A sketch of Proof:

We can show $\mathcal{P}$'s knowledge of $\{A_{ij}\}, \{r_i\}, \{\alpha_i\}$, and α satisfying Equations (9) and (10) from the satisfiability of Equation (11), similar to Lemma 3. From the satisfiability of Equations (11) and (16), and additionally that of Equations

(14) and (15), we can prove that the $\{A_{ij}\}$ satisfies the both conditions of Theorem 1, in a similar manner as proving Lemma 5. Thus Theorem 1 ensures that (A_{ij}) is a permutation matrix. The following lemma ensures that the same permutation matrix was applied to both $\{g_i\}$ and $\{m_i\}$ to achieve $\{g'_i\}$ and $\{m'_i\}$, yielding the correctness of the shuffle. □

Lemma 8. *Assume $\mathcal{P}$ knows $\{A_{ij}\}, \{r_i\}, \{\alpha_i\}$, and α satisfying Equations (9) and (10), and $\{s_i\}$ and s satisfying Equation (11). If Equations (12) and (13) hold with non-negligible probability, then either the relationships*

$$\begin{cases} g' = g^{\alpha} \prod_{j=1}^{n} {g_j}^{\alpha_j} \bmod p \\ g'_i = g^{r_i} \prod_{j=1}^{n} {g_j}^{A_{ji}} \bmod p \quad i = 1, \ldots, n \\ m' = y^{\alpha} \prod_{j=1}^{n} {m_j}^{\alpha_j} \bmod p \\ m'_i = y^{r_i} \prod_{j=1}^{n} {m_j}^{A_{ji}} \bmod p \quad i = 1, \ldots, n \end{cases} \tag{17}$$

hold or $\mathcal{P}$ can generate nontrivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} {\tilde{g}_i}^{a_i} = 1$ with overwhelming probability.

A sketch of Proof: Similarly to Lemma 4, we can ensure that

$$\begin{cases} s = \sum_{j=1}^{n} r_j c_j + \alpha \bmod q \\ s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q \quad i = 1, \ldots, n \end{cases}$$

hold from the satisfiability of Equation (11) unless $\mathcal{P}$ can generate non-trivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} {\tilde{g}_i}^{a_i} = 1$.

If Equation (12) holds, then

$$1 = \frac{g^{\alpha} \prod_{i=1}^{n} {g_i}^{\alpha_i}}{g'} \prod_{j=1}^{n} \left(\frac{g^{r_j} \prod_{i=1}^{n} {g_i}^{A_{ij}}}{g'_j} \right)^{c_j} \bmod p.$$

If first two equations on Equations (17) does not hold, then the probability that Equation (12) hold is negligible. The same thing can be said for m', $\{m'_i\}_{(i=1,\ldots,n)}$ from the satisfiability of (13). □

C.2 Zero-knowledge

Theorem 10. *We can construct a simulator of Main Protocol such that if there is a distinguisher who can distinguish between a real transcript from the protocol and an output from the simulator, then we can solve the decisional Diffie-Hellman problem.*

A sketch of Proof: We first give a construction of the simulator. We then prove that if there exists such a distinguisher then we can solve DDH^5_{n+1}. From Lemma 1 and 2, it means it is equivalent to solving the decisional Diffie-Hellman problem.

The construction of the Simulator

We will construct the simulator $\mathcal{S}$ of the main protocol with the input $p, q, g, y, \tilde{g}$, $\{\tilde{g}_i\}$, $\{(g_i, m_i)\}$, $\{(g'_i, m'_i)\}$ as follows.

The simulator $\mathcal{S}$ first generates $s, \{s_i\}, \{c_i\}, \lambda' \in_R \mathbf{Z_q}$, t, v, w, $\{u_i\}$, $\{\dot{t}_i\}$, $\{\dot{v}_i\}$, $\{\dot{w}_i\}$, $\{\tilde{g}'_i\} \in_R \mathbf{G_q}$ randomly. Then it computes $\tilde{g}', g', m', u, \dot{v}, \dot{w}$ as the following.

$$u = g^{\lambda'} \prod_{j=1}^{n} {u_j}^{-c_j^2} \bmod p$$

$$\tilde{g}' = \tilde{g}^s \prod_{j=1}^{n} {\tilde{g}_j}^{s_j} {\tilde{g}'_j}^{-c_j} \bmod p$$

$$g' = g^s \prod_{j=1}^{n} {g_j}^{s_j} {g'_j}^{-c_j} \bmod p$$

$$m' = y^s \prod_{j=1}^{n} {m_j}^{s_j} {m'_j}^{-c_j} \bmod p$$

$$\dot{v} = t^{\lambda'} v^s g^{\sum_{j=1}^{n}(s_j^3 - c_j^3)} \prod_{j=1}^{n} \dot{t}_j^{-c_j^2} {\dot{v}_j}^{-c_j} \bmod p$$

$$\dot{w} = w^s g^{\sum_{j=1}^{n}(s_j^2 - c_j^2)} \prod_{j=1}^{n} \dot{w}_j^{-c_j} \bmod p.$$

The output of $\mathcal{S}$ is
$\big(t, v, w, u, \{u_i\}, \{\tilde{g}'_i\}, \tilde{g}', g', m', \{\dot{t}_i\}, \{\dot{v}_i\}, \dot{v}, \{\dot{w}_i\}, \dot{w}, \{c_i\}, s, \{s_i\}, \lambda'\big)$.

A distinguisher of D^5_{n+1} and R^5_{n+1}

We will then construct a distinguisher $\mathcal{D}'$ who can distinguish between the uniform instances of D^5_{n+1} and R^5_{n+1} if $\mathcal{S}$ can not simulate the main protocol.

Let's say the instance I

$$I = (x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(5)}, \ldots, x_{n+1}^{(1)}, x_{n+1}^{(2)}, \ldots, x_{n+1}^{(5)})$$

was chosen uniformly from either D^5_{n+1} or R^5_{n+1}. Then this distinguisher will first generate $g_1, m_1, g_2, m_2, ..., g_n, m_n, \tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_n$ as the constants used in Main Protocol and let $X \in_R \mathbf{Z_q}, g = x_1^{(1)}, \tilde{g} = x_1^{(2)}, y = g^X \bmod p$. It will then generate a random permutation A_{ji} and a secret key $X \in_R \mathbf{Z_q}$ and compute

$$(g'_i, m'_i) = (x_{i+1}^{(1)} \prod_{j=1}^{n} g_j^{A_{ji}},\ (x_{i+1}^{(1)})^X \prod_{j=1}^{n} m_j^{A_{ji}}) \bmod p. \quad (i = 1, \ldots, n)$$

We note that $\{(g'_i, m'_i)\}$ gives a random shuffle of $\{(g_i, m_i)\}$.

Based on $g, y, \tilde{g}, \{\tilde{g}_i\}, \{(g_i, m_i)\}$ and $\{(g'_i, m'_i)\}$ the distinguisher $\mathcal{D}'$ is going to act as a simulator $\mathcal{S}'$ which simulates the simulator $\mathcal{S}$. More specifically, the simulator $\mathcal{S}'$ randomly generates

$$s, \{s_i\}, \{c_i\}, \lambda', \{\beta_i\} \in_R \mathbf{Z_q} \quad i = 1, \ldots, n$$

and computes

$$\begin{aligned}
t &= x_1^{(3)},\ v = x_1^{(4)},\ w = x_1^{(5)} \\
u_i &= (x_{i+1}^{(1)})^{\beta_i} \bmod p \quad i = 1, \ldots, n \\
u &= g^{\lambda'} \prod_{j=1}^{n} u_j^{-c_j^2} \bmod p \\
\tilde{g}'_i &= x_{i+1}^{(2)} \prod_{j=1}^{n} \tilde{g}_j^{A_{ji}} \bmod p \quad i = 1, \ldots, n \\
\tilde{g}' &= \tilde{g}^s \prod_{j=1}^{n} \tilde{g}_j{}^{s_j} \tilde{g}'_j{}^{-c_j} \bmod p \\
g' &= g^s \prod_{j=1}^{n} g_j{}^{s_j} g'_j{}^{-c_j} \bmod p \\
m' &= y^s \prod_{j=1}^{n} m_j{}^{s_j} m'_j{}^{-c_j} \bmod p \\
\alpha_j &= s_j - \sum_{k=1}^{n} A_{jk} c_k \bmod q \quad j = 1, \ldots, n \\
\dot{t}_i &= (x_{i+1}^{(3)})^{\beta_i} \prod_{j=1}^{n} g^{3\alpha_j A_{ji}} \bmod p \quad i = 1, \ldots, n \\
\dot{v}_i &= x_{i+1}^{(4)} \prod_{j=1}^{n} g^{3\alpha_j{}^2 A_{ji}} \bmod p \quad i = 1, \ldots, n \\
\dot{w}_i &= x_{i+1}^{(5)} \prod_{j=1}^{n} g^{2\alpha_j A_{ji}} \bmod p \quad i = 1, \ldots, n \\
\dot{v} &= t^{\lambda'} v^s g^{\sum_{j=1}^{n}(r_j^3 - c_j^3)} \prod_{j=1}^{n} (\dot{t}_j^{-c_j^2} \dot{v}_j^{-c_j}) \bmod p \\
\dot{w} &= w^s g^{\sum_{j=1}^{n}(s_j^2 - c_j^2)} \prod_{j=1}^{n} \dot{w}_j^{-c_j} \bmod p.
\end{aligned}$$

The simulator $\mathcal{S}'$ outputs
$\big(t, v, w, u, \{u_i\}, \{\tilde{g}'_i\}, \tilde{g}', g', m', \{\dot{t}_i\}, \{\dot{v}_i\}, \dot{v}, \{\dot{w}_i\}, \dot{w}, \{c_i\}, s, \{s_i\}, \lambda'\big)$.

Lemma 9. *Simulator $\mathcal{S}'$ perfectly simulates Main Protocol when $I \in_R D_{n+1}^5$.*

Sketch: We let

$$\log_{x_1^{(1)}} x_{i+1}^{(1)} = r_i, \ \ \log_{x_1^{(1)}} (x_{i+1}^{(1)})^{\beta_i} = \lambda_i$$
$$\log_{x_1^{(1)}} x_1^{(3)} = \tau, \ \ \log_{x_1^{(1)}} x_1^{(4)} = \rho, \ \ \log_{x_1^{(1)}} x_1^{(5)} = \sigma.$$

This gives for $i = 1, .., n$,

$$x_{i+1}^{(1)} = g^{r_i}, (x_{i+1}^{(1)})^{\beta_i} = g^{\lambda_i}, x_{i+1}^{(2)} = \tilde{g}^{r_i},$$
$$(x_{i+1}^{(3)})^{\beta_i} = g^{\tau\lambda_i}, \ x_{i+1}^{(4)} = g^{\rho r_i}, \ x_{i+1}^{(5)} = g^{\sigma r_i}.$$

Therefore, it is clear that by randomly choosing $s, \{s_i\}, \lambda'$ and $\{\beta_i\}$, it gives the same distribution of the output as when $\alpha, \{\alpha_i\}, \{\lambda_i\}$ and λ were first chosen randomly, and verifier honestly chooses random challenge $\{c_i\}$.

Lemma 10. *Simulator $\mathcal{S}'$ perfectly simulates $\mathcal{S}$ when $I \in_R R_{n+1}^5$*

Sketch: Since $x_i^{(2)}, x_i^{(3)}, x_i^{(4)}, x_i^{(5)} (i = 1, 2, ..., n+1)$ and $\beta_i (i = 1, 2, ..., n)$ are randomly chosen, it gives the same distribution when $\tilde{g}, t, v, w, \{\tilde{g}_i'\}, \{\dot{t}_i\}, \{\dot{v}_i\}, \{\dot{w}_i\}$ and $\{u_i\}$ are randomly chosen for $i = 1, 2, ..., n$.

□

Therefore, if there exists a distinguisher $\mathcal{D}$ that distinguishes the output of the simulator $\mathcal{S}$ and a real transcript of Main Protocol, then this distinguisher can be used to solve DDH_{n+1}^5.

D Alternative Notation

We present here an alternative notation of the variables. Since we have discussed the basis $\{g, g_1, \ldots, g_n\}$ throughout the paper, we can think of representing g by g_0. Similarly y by m_0 and $\tilde{g}$ by $\tilde{g}_0$. We can include the value of randomizers $\{r_i\}$, α_i and α, in the matrix by defining $A_{0i} = r_i$, $A_{i0} = \alpha_i$, and $A_{00} = \alpha$.

Treating a public key in a similar manner with input variables may be awkward, but it gives a compact representation to some of the variables, e.g,

$$g_\mu' = \prod_{\nu=0}^{n} {g_\nu}^{A_{\nu\mu}}, m_\mu' = \prod_{\nu=0}^{n} {m_\nu}^{A_{\nu\mu}}, \tilde{g}_\mu' = \prod_{\nu=0}^{n} {\tilde{g}_\nu}^{A_{\nu\mu}} \quad \mu = 0, \ldots, n.$$

Further suggestions for the alternative notation follows:

$$g_0' = g', m_0' = m', \tilde{g}_0' = \tilde{g}', s_0 = s, c_0 = 1, \lambda_0 = \lambda, u_0 = u$$
$$s_\mu = \sum_{\nu=0}^{n} A_{\mu\nu} c_\nu, \lambda' = \sum_{\nu=0}^{n} \lambda_\nu c_\nu^2, \prod_{\nu=0}^{n} g_\nu^{s_\nu} = \prod_{\nu=0}^{n} {g'}_\nu^{c_\nu}, g^{\lambda'} = \prod_{\nu=0}^{n} u_\nu^{c_\nu^2} \quad \mu = 0, \ldots, n.$$