Improved Zero-Knowledge Argument of Encrypted Extended Permutation

Yi Liu[1,3], Qi Wang[1,2], and Siu-Ming Yiu[3]

[1] Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
liuy7@mail.sustech.edu.cn
wangqi@sustech.edu.cn
[2] National Center for Applied Mathematics (Shenzhen),
Southern University of Science and Technology, Shenzhen 518055, China
[3] Department of Computer Science,
The University of Hong Kong, Pokfulam, Hong Kong SAR, China
smyiu@cs.hku.hk

Abstract. Extended permutation (EP) is a generalized notion of the standard permutation. Unlike the one-to-one correspondence mapping of the standard permutation, EP allows to replicate or omit elements as many times as needed during the mapping. EP is useful in the area of secure multi-party computation (MPC), especially for the problem of private function evaluation (PFE). As a special class of MPC problems, PFE focuses on the scenario where a party holds a private circuit C while all other parties hold their private inputs $x_1, \ldots, x_n$, respectively. The goal of PFE protocols is to securely compute the evaluation result $C(x_1, \ldots, x_n)$, while any other information beyond $C(x_1, \ldots, x_n)$ is hidden. EP here is introduced to describe the topological structure of the circuit C, and it is further used to support the evaluation of C privately. For an actively secure PFE protocol, it is crucial to guarantee that the private circuit provider cannot deviate from the protocol to learn more information. Hence, we need to ensure that the private circuit provider correctly performs an EP. This seeks the help of the so-called *zero-knowledge argument of encrypted extended permutation* protocol. In this paper, we provide an improvement of this protocol. Our new protocol can be instantiated to be non-interactive while the previous protocol should be interactive. Meanwhile, compared with the previous protocol, our protocol is significantly (*e.g.*, more than 3.4×) faster, and the communication cost is only around 24% of that of the previous one.

Keywords: ElGamal encryption · Extended permutation · Private function evaluation · Zero-knowledge.

1 Introduction

The notion of *extended permutation* (EP) is a generalized notion of the standard permutation. Different from the one-to-one correspondence mapping of the standard permutation, EP allows replication and omission of elements during the mapping. An EP π maps elements in a set $\{1, \dots, M\}$ to a set $\{1, \dots, N\}$ for positive integers M and N. Here, for every $y \in \{1, \dots, N\}$, there exists exactly one $x \in \{1, \dots, M\}$, such that $\pi(x) = y$. We note that π may not be a function, while π^{-1} is indeed a function.

EP is a very useful notion in many areas. In particular, EP is implicitly or explicitly used in the area of secure multi-party computation (MPC) [27]. In the setting of MPC, EP could be introduced to illustrate the topological structure of circuits. More concretely, EP can be used to describe the connections between wires of a circuit, and thus the topology of the circuit. To describe a circuit using EP, we divide the wires of the circuit into two types: *incoming* wires (IW) and *outgoing* wires (OW). All input wires of gates in the circuit are incoming wires, while the input wires of the circuit and the output wires of gates are outgoing wires. An example for a circuit C using such a naming rule is given in Fig. 1. It

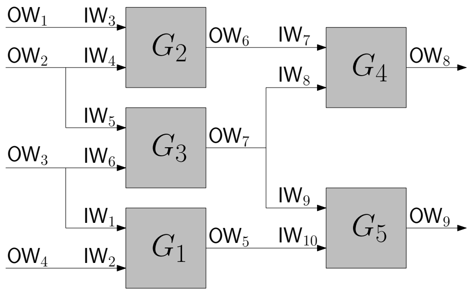


Fig. 1: An illustration of a circuit C, where wires are denoted by incoming wires (IW) and outgoing wires (OW).

is easy to see that every incoming wire connects to exactly one outgoing wire. Meanwhile, an outgoing wire may connect to one or multiple incoming wires, or has no connection to any incoming wires. It is clear that for a circuit, its outgoing wires correspond to the domain of an EP, and its incoming wires correspond to the range of an EP. Therefore, after indexing the incoming wires and outgoing wires of a circuit, we can extract an EP that describes the topology of the circuit. In Fig. 2, we provide the corresponding EP for the circuit C in Fig 1. [4] Moreover, given an EP (together with the numbers of gates, inputs, and outputs), we can easily reconstruct the topological structure of the corresponding circuit.

EP is especially useful for the problem of (general-purpose) private function evaluation (PFE) [1]. PFE is a special class of MPC problems. It focuses on designing a protocol for the scenario where a party holds a private circuit C, while other parties possess their own private inputs $x_1, \dots, x_n$. The goal of PFE is to privately evaluate C on $x_1, \dots, x_n$, *i.e.*, to compute the evaluation result

[4] Since OW_8 and OW_9, as output wires of the circuit C, have no connections to other wires, we can simply omit them in the EP.

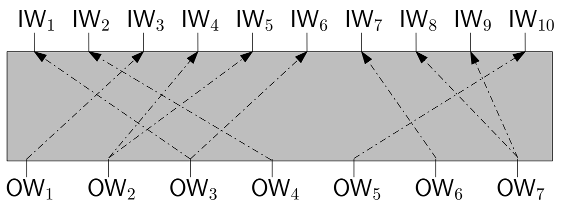


Fig. 2: The extended permutation corresponding to the circuit C in Fig. 1.

$C(x_1, \ldots, x_n)$. After the execution of PFE, parties receive the evaluation result $C(x_1, \ldots, x_n)$ while information beyond $C(x_1, \ldots, x_n)$ is hidden. Note that this is different from traditional secure function evaluation problem, in which the circuit C is publicly known. In fact, PFE problem can be reduced to securely evaluating a universal circuit [26, 19, 18, 22, 13, 28, 2, 23], such that the description of the circuit C is used as inputs to the universal circuit. However, using universal circuits leads to a logarithmic blow-up. In other words, the universal circuit for evaluating a circuit C with size n has size at least $\Theta(n \log n)$, where the constant factor (*e.g.*, 12) and the low-order terms are significant. Starting from the original work of Katz and Malka [17], another line of research focuses on designing PFE protocols while avoiding the usage of universal circuits, such as [24, 20, 25, 5, 15, 4]. This line of work has *linear* complexity in the size of the circuit n. It was shown [17, 15] that they outperformed the state-of-the-art PFE protocol based on universal circuits theoretically and experimentally. The basic idea for this line of work is to use EP. More concretely, the party holding the private circuit C derives an EP from C, and *obliviously* performs an EP on a set of outgoing wires to establish the connections between outgoing wires and incoming wires. Then parties are able to follow the results from the EP to evaluate C on private inputs while keeping C hidden.

Although this line of work usually has good performance, only the work in [25] is secure against malicious adversaries, and all other results only work in the semi-honest model. One of the main challenges for designing an actively secure PFE protocol is to guarantee that the private function owner performs a valid EP on elements representing outgoing wires. In the setting of [25], the private circuit owner performs an EP on a set of encrypted elements locally and re-randomizes all encrypted elements in the resulting list. Then the private circuit owner is required to publish the resulting encrypted list and prove that the resulting encrypted list is derived from a valid EP on the encrypted elements in a zero-knowledge manner. The protocol for proving the validity of this result is called *zero-knowledge argument of encrypted extended permutation*, and it is also the efficiency bottleneck of the protocol [25].

1.1 Contribution

In this paper, we provide an improved version of the zero-knowledge argument of encrypted extended permutation protocol. Both our protocol and the original protocol in [25] are designed based on the ElGamal encryption scheme [11]. It is possible to extend our ideas to other encryption schemes. We note that our protocol can be instantiated to be non-interactive while the previous protocol should be interactive. Compared with the original work [25], the communication cost of our protocol is only around 24% of that of [25]. For computation cost, our protocol is significantly (*e.g.*, more than $3.4\times$) faster than the previous protocol. Moreover, protocols based on our protocol, such as the linear actively secure PFE protocol in [25], can also gain better performance.

1.2 Overview of Our Idea

Before the full description of our protocol, we here briefly provide an overview of our idea. We denote the EP π by a mapping $\pi : \{1, \ldots, M\} \to \{1, \ldots, N\}$. Informally, given two lists of ciphertexts $\vec{\alpha} = [\alpha_1, \ldots, \alpha_M]$ and $\vec{c} = [c_1, \ldots, c_N]$, the goal of the prover in our protocol is to prove that there exists an EP π, such that the encrypted element of c_i is the same as the encrypted element of $\alpha_{\pi^{-1}(i)}$. A formal definition for the relation corresponding to our protocol will be given in Section 2. The idea of our protocol is to decompose a valid EP into four steps: extension, placement, replication, and finalization, and then the prover shows their validity respectively. The four steps are described in the following.

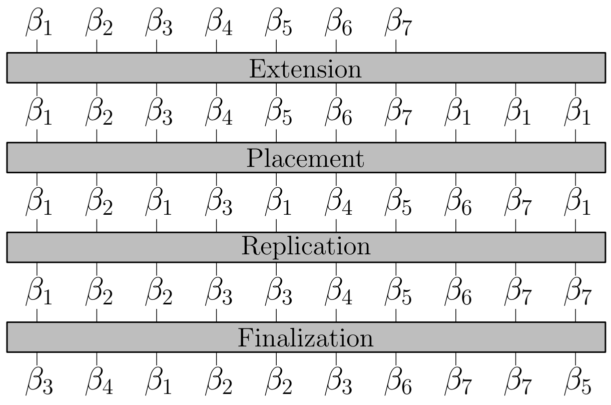
Extension If $M < N$, we know that the length of the resulting ciphertext list $\vec{c}$ is longer than that of the original ciphertext list $\vec{\alpha}$. Therefore, all parties append $N - M$ ciphertexts at the end of $\vec{\alpha}$ as dummy ciphertexts. To ensure that the dummy ciphertexts are meaningless while the resulting new list is derived from a valid EP performed on the original list, all parties could append $N - M$ ciphertext α_1 at the end of $\vec{\alpha}$. If $M \geq N$, we can safely skip this extension step.

Placement If the encrypted element of a ciphertext in $\vec{\alpha}$ does not appear in the resulting list $\vec{c}$ (in an encrypted form), *i.e.*, this element is omitted according to the mapping of the EP π, the prover can label this ciphertext also as a dummy ciphertext. The prover now permutes the list, such that for each ciphertext encrypting the (non-omitted) element in the original list, if it is mapped to k different outputs according to π, $k - 1$ dummy ciphertexts are placed after this ciphertext. If $M > N$, extra dummy ciphertexts are moved to the end of the list. Then all ciphertexts are re-randomized.

Replication The prover replaces all dummy ciphertexts except extra dummy ciphertexts with their first non-dummy ciphertext. In other words, if a non-omitted element is mapped to k different outputs according to π, its corresponding ciphertext is replicated $k - 1$ times thereafter. Then all ciphertexts are re-randomized.

Finalization If $M > N$, parties can remove the last $M - N$ extra dummy ciphertexts from the list. Now the prover can permute the list to their final place according to π. Finally, all ciphertexts are re-randomized to derive $\vec{c}$.

Then the prover is required to prove that each step is executed correctly in the protocol. We give an illustration of these four steps for the EP corresponding to the circuit C (Fig. 1) in Fig 3, where we use β_i's to denote the encrypted elements of ciphertexts in the list $\vec{\alpha}$.



β_1 β_2 β_3 β_4 β_5 β_6 β_7
Extension
β_1 β_2 β_3 β_4 β_5 β_6 β_7 β_1 β_1 β_1
Placement
β_1 β_2 β_1 β_3 β_1 β_4 β_5 β_6 β_7 β_1
Replication
β_1 β_2 β_2 β_3 β_3 β_4 β_5 β_6 β_7 β_7
Finalization
β_3 β_4 β_1 β_2 β_2 β_3 β_6 β_7 β_7 β_5

Fig. 3: The four steps for the proof corresponding to the circuit C in Fig. 1.

The organization for the rest of this paper is as follows. In Section 2, we present preliminaries for our further presentation. Then we provide a formal description for our main protocol in Section 3. Subsequently, sub-protocols inside our main protocol are given in Section 4. Finally, performance of our protocol and comparisons between our protocol and the original work [25] are presented in Section 5, from both communication and computation aspects.

2 Preliminaries

In this paper, the security of protocols is proved under standard security definitions (see [14, 21] for more information). This paper mainly focuses on constructing a public-coin honest-verifier zero-knowledge protocol (see [12]). Note that this kind of protocols can be compiled by the Fiat–Shamir heuristic [9] to be non-interactive and secure against malicious verifiers with *low overhead*.

We use the notation $\|S\|$ to denote the number of bits required to represent elements in the set S. We write $x \leftarrow_{\$} S$ to indicate that an element x is uniformly sampled from the set S. Define $[n] = \{1, \ldots, n\}$. The function $\max(\cdot, \cdot)$ takes as input two values and returns the maximum of these two values. We say that a function f in a variable κ mapping natural numbers to $[0, 1]$ is *negligible* if $f(\kappa) = \mathcal{O}(\kappa^{-c})$ for every constant $c > 0$.

We give the formal definition of EP in the following.

Definition 1 (Extended Permutation [24]). *For positive integers M and N, a mapping $\pi : [M] \to [N]$ is an extended permutation (EP) if for every $y \in [N]$, there exists exactly one $x \in [M]$, such that $\pi(x) = y$. We often denote x by $\pi^{-1}(y)$.*

Here, we give a brief description of the ElGamal encryption scheme [11]. This encryption scheme is over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. It is semantically secure under the decisional Diffie-Hellman (DDH) assumption (see [16]) for $\mathbb{G}$. The description of the scheme is in the following.

Key Generation The algorithm KGen takes as input the security parameter 1^κ, picks $s \leftarrow_\$ \mathbb{Z}_q$, and sets $h \leftarrow g^s$. Then the algorithm outputs the public key $\mathsf{pk} \leftarrow (\mathbb{G}, q, g, h)$ and the private key $\mathsf{sk} \leftarrow s$.

Encryption The algorithm Enc takes as input a message $m \in \mathbb{G}$ and a public key pk, and returns the ciphertext $c \leftarrow (c^{(0)} = g^r, c^{(1)} = mh^r)$ for a random coin $r \leftarrow_\$ \mathbb{Z}_q$.

Decryption The algorithm Dec takes as input a ciphertext $c = (c^{(0)}, c^{(1)})$ and a key pair $(\mathsf{pk}, \mathsf{sk})$, and returns the plaintext $m \leftarrow c^{(1)}/(c^{(0)})^s$.

Remark 1. For the ElGamal encryption scheme with $\mathsf{pk} = (\mathbb{G}, q, g, h)$, it is easy for a prover to prove in zero-knowledge that two ElGamal ciphertexts encrypt the same value. Without loss of generality, we denote two ciphertexts by $c_1 = (c_1^{(0)}, c_1^{(1)}) = (g^{r_1}, mh^{r_1})$ and $c_2 = (c_2^{(0)}, c_2^{(1)}) = (g^{r_2}, mh^{r_2})$, such that they encrypt the same value m.

When we compute $c_3 \leftarrow (c_1^{(0)}(c_2^{(0)})^{-1}, c_1^{(1)}(c_2^{(1)})^{-1}) = (g^{r_1-r_2}, h^{r_1-r_2})$, the resulting ciphertext c_3 indeed encrypts 1. Therefore, c_1 and c_2 encrypt the same value if and only if c_3 encrypts 1. Let $r = r_1 - r_2$. If a prover knows r, she is able to prove that c_1 and c_2 encrypt the same value via proving that $(g, h, c_3^{(0)}, c_3^{(1)})$ is a Diffie-Hellman (DH) tuple. More concretely, it is equivalent for the prover to prove in zero-knowledge that there exists a value $r \in \mathbb{Z}_q$, such that $c_3^{(0)} = g^r$ and $c_3^{(1)} = h^r$.

In this paper, we aim to provide a zero-knowledge protocol for the relation R_{EncEP} based on the ElGamal encryption scheme:

$$\begin{aligned} R_{\mathsf{EncEP}} = \{(\mathbb{G}, q, g, h, \{(\alpha_i^{(0)}, \alpha_i^{(1)})\}_{i\in[M]}, \{(c_i^{(0)}, c_i^{(1)})\}_{i\in[N]}) \mid \exists \{r_i\}_{i\in[N]}, \pi, s.t. \\ c_i^{(0)} = \alpha_{\pi^{-1}(i)}^{(0)} g^{r_i} \wedge c_i^{(1)} = \alpha_{\pi^{-1}(i)}^{(1)} h^{r_i} \wedge \pi \text{ is an extended permutation}\} \end{aligned}$$

Our construction utilizes a zero-knowledge protocol $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ for the relation R_{Shuffle} based on the ElGamal encryption scheme:

$$\begin{aligned} R_{\mathsf{Shuffle}} = \{(\mathbb{G}, q, g, h, \{(c_i^{(0)}, c_i^{(1)})\}_{i\in[\ell]}, \{(c_i'^{(0)}, c_i'^{(1)})\}_{i\in[\ell]}) \mid \exists \{r_i\}_{i\in[\ell]}, \pi, s.t. \\ c_i'^{(0)} = c_{\pi(i)}^{(0)} g^{r_i} \wedge c_i'^{(1)} = c_{\pi(i)}^{(1)} h^{r_i} \wedge \pi \text{ is a permutation}\} \end{aligned}$$

We note that there exist efficient (non-interactive) protocols with sub-linear communication cost that can be used to instantiate $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$, such as the protocol in [3].

3 Our Main Protocol

Based on the idea introduced in Section 1.2, we provide a full description of our main protocol in this section. The sub-protocols inside our protocol are given in Section 4.

The zero-knowledge protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ for R_{EncEP} between a prover P and a verifier V is given as follows.

Public Inputs: A group $\mathbb{G}$ of order q with generator g, where DDH assumption holds. The public key of the ElGamal encryption scheme $\mathsf{pk} = (\mathbb{G}, q, g, h)$. Two lists of ElGamal ciphertexts $\vec{\alpha} = [\alpha_1, \dots, \alpha_M]$ and $\vec{c} = [c_1, \dots, c_N]$ corresponding to pk. Each ciphertext α_i (resp. c_i) is of the form $\alpha_i = (\alpha_i^{(0)}, \alpha_i^{(1)}) \in \mathbb{G}^2$ (resp. $c_i = (c_i^{(0)}, c_i^{(1)}) \in \mathbb{G}^2$).

Witness: An EP $\pi : [M] \to [N]$ and a list $R = [r_1, \dots, r_N]$, where $r_i \in \mathbb{Z}_q$.

Statement: There exists an EP π and a list $R = [r_1, \dots, r_N]$, such that $c_i^{(0)} = \alpha_{\pi^{-1}(i)}^{(0)} g^{r_i}$ and $c_i^{(1)} = \alpha_{\pi^{-1}(i)}^{(1)} h^{r_i}$.

Protocol Description

1. **Extension.** Both parties append $\max(N - M, 0)$ ciphertexts α_1 to the list $\vec{\alpha}$ as dummy ciphertexts. The new list is denoted by $\vec{e} = [e_1, \dots, e_N]$. Let $N' = \max(M, N)$.
2. **Placement.** If the index i of a ciphertext in $\vec{e}$ satisfies $i \le M$ and $\{j \mid i = \pi^{-1}(j)\} = \emptyset$, *i.e.*, this encrypted element is omitted after the EP, P also labels this ciphertext as a dummy ciphertext. P now permutes the list $\vec{e}$, such that for each non-dummy ciphertext in $\vec{e}$ with index j, if $|\pi(j)| = k$, $k-1$ dummy ciphertexts are placed after this ciphertext. The condition $|\pi(j)| = k$ means that this encrypted element is mapped to k different outputs according to π. If $M > N$, extra dummy ciphertexts are moved to the end of the list. This permutation is denoted by π' and the resulting list of ciphertexts is denoted by $\vec{\hat{p}} = [\hat{p}_1, \dots, \hat{p}_{N'}]$, where $\hat{p}_i = e_{\pi'(i)}$.
   Then P picks $r'_i \leftarrow_{\$} \mathbb{Z}_q$ for $i \in [N']$ and computes the ElGamal ciphertext $p_i \leftarrow (\hat{p}_i^{(0)} g^{r'_i}, \hat{p}_i^{(1)} h^{r'_i})$ for $i \in [N']$. We denote the resulting list by $\vec{p} = [p_1, \dots, p_{N'}]$. P sends $\vec{p}$ to V.
3. **Replication.** P replaces all dummy ciphertexts except extra dummy ciphertexts by the nearest non-dummy ciphertexts before each of them. In other words, if a non-omitted element is mapped to k different outputs according to π, its corresponding ciphertext is replicated $k-1$ times thereafter. We define a function $\omega : [N] \to [N]$ that maps an input index i to the index of a *non-dummy* ciphertext j, such that j is the maximum index of non-dummy ciphertext in $\vec{p}$ that satisfies $j \le i$. We note that for a dummy ciphertext with index i, $\omega(i)$ is the index of the ciphertext that replaces it during this replication procedure.
   Let the resulting list be $\vec{\hat{\rho}} = [\hat{\rho}_1, \dots, \hat{\rho}_{N'}]$. We have $\hat{\rho}_i = p_{\omega(i)}$ for $i \in [N]$. P picks $r''_i \leftarrow_{\$} \mathbb{Z}_q$, and computes the ElGamal ciphertext $\rho_i \leftarrow (\hat{\rho}_i^{(0)} g^{r''_i}, \hat{\rho}_i^{(1)} h^{r''_i})$

for each $i \in [N]$. This resulting list is denoted by $\vec{\rho} = [\rho_1, \ldots, \rho_{N'}]$. Note that if $N - M < 0$, the last $N - M$ ciphertext are still the extra dummy ciphertexts in $\vec{p}$. P sends (the first N elements of) $\vec{\rho}$ to V.

4. **Finalization.** V obtains $\vec{\rho}$. If $N - M < 0$, both parties remove the last $M - N$ extra ciphertexts from $\vec{\rho}$. No matter whether we need to remove extra ciphertexts or not, we denote the current list of ciphertexts by $\vec{\rho}'$. P permutes all ciphertexts to their final location as prescribed by π. We denote this permutation by π'' and the resulting list by $\vec{\hat{f}} = [\hat{f}_1, \ldots, \hat{f}_N]$, where $\hat{f}_i = \rho_{\pi''(i)}$. Then P computes $\hat{r}_i \leftarrow r_i - r''_{\pi''(i)} - r'_{\omega(\pi''(i))} \bmod q$ for $i \in [N]$. It is easy to verify that $(\hat{f}_i^{(0)} g^{\hat{r}_i}, \hat{f}_i^{(1)} h^{\hat{r}_i}) = c_i$.

The remaining work is to show that these four steps are executed correctly. Since the extension step is done by both parties, P only needs to show that what she has done is correct in the last three steps. Namely, P needs to prove in zero-knowledge that $\vec{p}$ and $\vec{c}$ are derived from valid shuffles applied to $\vec{e}$ and $\vec{\rho}'$, respectively, and $\vec{\rho}$ is derived from a valid dummy ciphertext replacement (replication) applied to $\vec{p}$. Hence, P and V together follow the detailed procedure below to prove the correctness of P's operations in these last three steps.

5. P uses the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ to prove that $\vec{p}$ is derived from a valid shuffle applied to $\vec{e}$ with witness $(\{r'_i\}_{i \in [N']}, \pi')$.
6. P uses the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ to prove that $\vec{c}$ is derived from a valid shuffle applied to $\vec{\rho}'$ with witness $(\{\hat{r}_i\}_{i \in [N]}, \pi'')$.
7. To prove that $\vec{\rho}$ is derived from a valid dummy ciphertext replacement from $\vec{p}$, P needs to prove that the plaintext of ρ_1 is equal to p_1, and that the plaintext of each ρ_i is equal to that of ρ_{i-1} or that of p_i for $i = 2, \ldots, N$. According to Remark 1, the goal can be translated to prove the correctness of the corresponding DH tuple. Both parties compute two ciphertexts

$$\gamma_{i,0} \leftarrow (\rho_i^{(0)}(\rho_{i-1}^{(0)})^{-1}, \rho_i^{(1)}(\rho_{i-1}^{(1)})^{-1})$$

and

$$\gamma_{i,1} \leftarrow (\rho_i^{(0)}(p_i^{(0)})^{-1}, \rho_i^{(1)}(p_i^{(1)})^{-1})$$

for $i = 2, \ldots, N$, together with

$$\gamma_{1,0} = \gamma_{1,1} \leftarrow (\rho_1^{(0)}(p_1^{(0)})^{-1}, \rho_1^{(1)}(p_1^{(1)})^{-1}) .$$

For $i = 2, \ldots, N$, if the plaintext of ρ_i is equal to that of ρ_{i-1}, the random coin of $\gamma_{i,0}$ is $\nu_{i,0} = r''_i - r''_{i-1} \bmod q$, and we let $b_i = 0$. If the plaintext of ρ_i is equal to that of p_i, the random coin of $\gamma_{i,1}$ is $\nu_{i,1} = r''_i$, and we let $b_i = 1$. In addition, the random coin for both $\gamma_{1,0}$ and $\gamma_{1,1}$ is $\nu_{1,0} = \nu_{1,1} = r''_1$.
P computes $\{\nu_{i,b_i}\}_{i \in [N]}$ and uses the protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ to prove the following statement:

There exists a set of elements $\{\nu_{i,b_i}\}_{i \in [N]}$, where $b_i \in \{0, 1\}$ and $\nu_{i,b_i} \in \mathbb{Z}_q$, such that $\gamma_{i,b_i}^{(0)} = g^{\nu_{i,b_i}}$ and $\gamma_{i,b_i}^{(1)} = h^{\nu_{i,b_i}}$ for all $i \in [N]$.

8. If all the executions of $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{DH}}$ output accept, V outputs accept. Otherwise, V outputs reject.

In what follows, we present a theorem for the security of the protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$.

Theorem 1. *The protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ is a zero-knowledge argument of knowledge for the relation R_{EncEP}.*

Proof. It is easy to verify the completeness of the protocol. We first show that $(\hat{f}_i^{(0)} g^{\hat{r}_i}, \hat{f}_i^{(1)} h^{\hat{r}_i}) = c_i$. More concretely, we can verify that

$$\begin{aligned}
\hat{f}_i^{(0)} g^{\hat{r}_i} &= \rho^{(0)}_{\pi''(i)} g^{r_i - r''_{\pi''(i)} - r'_{\omega(\pi''(i))}} \\
&= \rho^{(0)}_{\pi''(i)} g^{-r''_{\pi''(i)}} g^{-r'_{\omega(\pi''(i))}} g^{r_i} \\
&= \hat{\rho}^{(0)}_{\pi''(i)} g^{r''_{\pi''(i)}} g^{-r''_{\pi''(i)}} g^{-r'_{\omega(\pi''(i))}} g^{r_i} \\
&= \hat{\rho}^{(0)}_{\pi''(i)} g^{-r'_{\omega(\pi''(i))}} g^{r_i} \\
&= p^{(0)}_{\omega(\pi''(i))} g^{-r'_{\omega(\pi''(i))}} g^{r_i} \\
&= \hat{p}^{(0)}_{\omega(\pi''(i))} g^{r'_{\omega(\pi''(i))}} g^{-r'_{\omega(\pi''(i))}} g^{r_i} \\
&= \hat{p}^{(0)}_{\omega(\pi''(i))} g^{r_i} \\
&= e^{(0)}_{\pi'(\omega(\pi''(i))} g^{r_i} \\
&= \alpha^{(0)}_{\pi^{-1}(i)} g^{r_i}
\end{aligned}$$

Similarly, we have $\hat{f}_i^{(1)} g^{\hat{r}_i} = \alpha^{(1)}_{\pi^{-1}(i)} h^{r_i}$. Then, if the prover P honestly proves that all the operations conducted in the four steps are correct using related parameters derived in the operations, the completeness of the protocol directly follows from the completeness of the protocols $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{DH}}$.

Then we show that the protocol achieves the zero-knowledge property. For an adversary $\mathcal{A}$ controlling the verifier V, we construct a simulator $\mathcal{S}$ that internally runs V and simulates V's view. $\mathcal{S}$ firstly sets $N' \leftarrow \max(M, N)$ as in the protocol. For the step of placement, $\mathcal{S}$ randomly picks $p_i \leftarrow_{\$} \mathbb{G}^2$ for $i \in [N']$ and sends the list $\vec{p} = [p_1, \ldots, p_{N'}]$ to $\mathcal{A}$. For the step of replication, $\mathcal{S}$ randomly generates $\rho_i \leftarrow_{\$} \mathbb{G}^2$ for $i \in [N']$ and sends $\vec{\rho} = [\rho_1, \ldots, \rho_{N'}]$ to $\mathcal{A}$. Then $\mathcal{S}$ invokes the simulator $\mathcal{S}_{\mathsf{Shuffle}}$ for the protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Shuffle}}$ twice, for both Steps 5 and 6, to simulate the view of $\mathcal{A}$ in the execution of $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Shuffle}}$. $\mathcal{S}$ computes elements of $\{\gamma_{i,b}\}_{i \in [N], b \in \{0,1\}}$ as in the protocol and uses the simulator $\mathcal{S}_{\mathsf{DH}}$ for the protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{DH}}$ to simulate the view of $\mathcal{A}$ in the execution of $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{DH}}$. Finally, $\mathcal{S}$ outputs what $\mathcal{A}$ outputs to complete the simulation.

In the simulation, we note that elements in the lists $\vec{p}$ and $\vec{\rho}$ are all randomly generated ciphertexts, while those elements in a real execution are based on the extended permutation π. However, since the ElGamal encryption scheme in the protocol is semantically secure under the DDH assumption, all computationally bounded adversaries cannot distinguish these simulated ciphertexts from ciphertexts generated in a real execution except for a negligible probability. The other difference between the simulation and the real execution of the protocol is for the sub-protocols $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$. Because both the sub-protocols $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ are also zero-knowledge, $\mathcal{A}$'s view simulated by the corresponding simulators $\mathcal{S}_{\mathsf{Shuffle}}$ and $\mathcal{S}_{\mathsf{DH}}$ is computationally indistinguishable from a real execution. Therefore, the protocol is zero-knowledge.

We analyze the soundness of the protocol as follows. The prover P follows the four steps to perform the extended permutation on the original list of ciphertexts $\vec{\alpha}$ and derive the list of resulting ciphertexts $\vec{c}$. Intuitively, the protocols $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ guarantee that no new ciphertexts except encrypted values inside $\vec{\alpha}$ are added to the resulting list of ciphertexts $\vec{c}$, and thus a valid extended permutation is performed on the encrypted values in $\vec{\alpha}$. Our goal now is to extract the extended permutation π and random coins $\{r_i\}_{i\in[N]}$. In the following, we construct an extractor $\mathcal{E}$ that internally runs the prover P^* and extracts the corresponding witness.

The extractor $\mathcal{E}$ runs the prover P^* as a subroutine. Then $\mathcal{E}$ uses the extractor $\mathcal{E}_{\mathsf{Shuffle}}$ for the sub-protocol $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ in Steps 5 and 6 to extract the witness in these two execution of $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$. Namely, $\mathcal{E}$ uses $\mathcal{E}_{\mathsf{Shuffle}}$ to extract witness $(\{r'_i\}_{i\in[N']}, \pi')$ and $(\{\hat{r}_i\}_{i\in[N]}, \pi'')$ in Steps 5 and 6, respectively. Meanwhile, $\mathcal{E}$ invokes the extractor $\mathcal{E}_{\mathsf{DH}}$ for the sub-protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ in Step 7 to extract the random coins $\{\nu_{i,b_i}\}_{i\in[N]}$ of the ciphertexts $\{\gamma_{i,b_i}\}_{i\in[N]}$ (encrypting 1) and corresponding $\{b_i\}_{i\in[N]}$.

$\mathcal{E}$ can reconstruct the corresponding mapping ω. Then $\mathcal{E}$ iteratively assigns the value of $\omega(i)$ as follows. Let $\omega(1) = 1$. Then for $i = 2, \ldots, N$, let

$$\omega(i) = \begin{cases} \omega(i-1) & \text{if } b_i = 0\,, \\ i & \text{if } b_i = 1\,. \end{cases}$$

Meanwhile, $\mathcal{E}$ can effectively compute the random coins $\{r''_i\}_{i\in[N]}$. Firstly, $\mathcal{E}$ sets $r''_1 \leftarrow \nu_{1,b_1}$. Then $\mathcal{E}$ iteratively assigns the value of r''_i as follows. For $i = 2, \ldots, N$,

$$r''_i = \begin{cases} r''_{i-1} + \nu_{i,b_i} \bmod q & \text{if } b_i = 0\,, \\ \nu_{i,b_i} & \text{if } b_i = 1\,. \end{cases}$$

Now, $\mathcal{E}$ can derive all the random coins for the extended permutation via computing

$$r_i = \hat{r}_i + r''_{\pi''(i)} + r'_{\omega(\pi''(i))} \bmod q$$

for $i \in [N]$. Since $\mathcal{E}$ has obtained π', π'', and ω, $\mathcal{E}$ can reconstruct the extended permutation π as

$$\pi(i) = {\pi''}^{-1} \circ \omega^{-1} \circ {\pi'}^{-1}(i)\,.$$

Therefore, the extractor successfully derives the extended permutation π and the list $R = [r_1, \ldots, r_N]$, and the soundness of the protocol is then proved. □

4 Sub-Protocols

As we have mentioned in Section 2, there exist efficient (non-interactive) protocols with sub-linear communication cost that can be used to instantiate the zero-knowledge protocol for shuffle ($\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$). In this paper, we use the protocol in [3] as $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$.

Now we provide the sub-protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ inside our main protocol. We note that this zero-knowledge protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ is for the relation R_{DH}:

$$R_{\mathsf{DH}} = \{(\mathbb{G}, q, g, h, \{(\gamma_i^{(0)}, \gamma_i^{(1)})\}_{i\in[\ell]} \mid \exists\{\nu_{i,b_i}\}_{i\in[\ell]}, \text{where } b_i \in \{0,1\} \ s.t. \\ \forall i \ (\gamma_{i,b_i}^{(0)} = g^{\nu_{i,b_i}} \wedge \gamma_{i,b_i}^{(1)} = h^{\nu_{i,b_i}})\} \, .$$

In the following, we describe the protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ for the relation R_{DH} between a prover P and a verifier V utilizing the idea introduced in [7] and [8]. This protocol is honest-verifier zero-knowledge and can be compiled by the Fiat–Shamir heuristic [9] to be non-interactive and secure against malicious verifiers as we have mentioned in Section 2.

Public Inputs: A group $\mathbb{G} = \langle g \rangle$ of order q. Another generator h for $\mathbb{G}$. A set of elements $\{(\gamma_{i,b}^{(0)}, \gamma_{i,b}^{(1)})\}_{i\in[\ell],b\in\{0,1\}}$, where $(\gamma_{i,b}^{(0)}, \gamma_{i,b}^{(1)}) \in \mathbb{G}^2$.

Witness: A list $[\nu_{1,b_1}, \ldots, \nu_{\ell,b_\ell}]$, where $\nu_{i,b_i} \in \mathbb{Z}_q$ and $b_i \in \{0,1\}$.

Statement: Given ciphertexts $\{(\gamma_{i,b}^{(0)}, \gamma_{i,b}^{(1)})\}_{i\in[\ell],b\in\{0,1\}}$, there exist $\{\nu_{i,b_i}\}_{i\in[\ell]}$, where $\nu_{i,b_i} \in \mathbb{Z}_q$ and $b_i \in \{0,1\}$, such that $\gamma_{i,b_i}^{(0)} = g^{\nu_{i,b_i}}$ and $\gamma_{i,b_i}^{(1)} = h^{\nu_{i,b_i}}$ for all $i \in [\ell]$.

Protocol Description

1. For $i \in [\ell]$:
   (a) P picks $e_{i,1-b_i} \leftarrow_\$ \mathbb{Z}_q$ and $z_{i,1-b_i} \leftarrow_\$ \mathbb{Z}_q$.
   (b) P computes
   $$a_{i,1-b_i}^{(0)} \leftarrow g^{z_{i,1-b_i}} (\gamma_{i,1-b_i}^{(0)})^{-e_{i,1-b_i}}$$
   and
   $$a_{i,1-b_i}^{(1)} \leftarrow h^{z_{i,1-b_i}} (\gamma_{i,1-b_i}^{(1)})^{-e_{i,1-b_i}}$$
   to simulate a valid transcript.
   (c) P picks $x_{i,b_i} \leftarrow_\$ \mathbb{Z}_q$. Then P computes $a_{i,b_i}^{(0)} = g^{x_{i,b_i}}$ and $a_{i,b_i}^{(1)} = h^{x_{i,b_i}}$.

   P sends $\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell],b\in\{0,1\}}$ to V.
2. V chooses $e, \theta \leftarrow_\$ \mathbb{Z}_q$ and sends them to P.
3. For $i \in [\ell]$:
   (a) P computes $e_{i,b_i} \leftarrow e - e_{i,1-b_i} \bmod q$.
   (b) P computes $z_{i,b_i} \leftarrow x_{i,b_i} + \nu_{i,b_i} e_{i,b_i} \bmod q$.
   (c) P computes $z_0 \leftarrow \sum_{i=1}^{\ell} z_{i,0}\theta^i \bmod q$ and $z_1 \leftarrow \sum_{i=1}^{\ell} z_{i,1}\theta^i \bmod q$.

   P sends $\{e_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, z_0, and z_1 to V.

4. V verifies the following equations:

$$e_{i,0} + e_{i,1} \equiv e \bmod q$$

for $i \in [\ell]$, and

$$g^{z_b} = \prod_{i=1}^{\ell} (a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{e_{i,b}})^{\theta^i}$$

and

$$h^{z_b} = \prod_{i=1}^{\ell} (a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{e_{i,b}})^{\theta^i}$$

for $b \in \{0,1\}$. If all equations hold, V outputs accept. Otherwise, V outputs reject.

Theorem 2. *The protocol $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$ is an honest-verifier zero-knowledge proof of knowledge for the relation R_{DH}.*

Proof. For the completeness of the protocol, we can verify that:

$$\begin{aligned} g^{z_{i,b_i}} &= g^{x_{i,b_i} + \nu_{i,b_i} e_{i,b_i}} \\ &= g^{x_{i,b_i}} g^{\nu_{i,b_i} e_{i,b_i}} \\ &= a_{i,b_i}^{(0)} (\gamma_{i,b_i}^{(0)})^{e_{i,b_i}} \end{aligned}$$

and

$$\begin{aligned} h^{z_{i,b_i}} &= h^{x_{i,b_i} + \nu_{i,b_i} e_{i,b_i}} \\ &= h^{x_{i,b_i}} h^{\nu_{i,b_i} e_{i,b_i}} \\ &= a_{i,b_i}^{(1)} (\gamma_{i,b_i}^{(1)})^{e_{i,b_i}} . \end{aligned}$$

Meanwhile, for the verification related to $1 - b_i$ in Step 4, values $z_{i,1-b_i}$, $e_{i,1-b_i}$, $a_{i,1-b_i}^{(0)}$, and $a_{i,1-b_i}^{(1)}$ generated in Step 1 are specified to satisfy the equation for the verification. Therefore, we have

$$\begin{aligned} g^{z_b} &= g^{\sum_{i=1}^{\ell} z_{i,b} \theta^i} \\ &= \prod_{i=1}^{\ell} (g^{z_{i,b}})^{\theta^i} \\ &= \prod_{i=1}^{\ell} (a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{e_{i,b}})^{\theta^i} \end{aligned}$$

and

$$\begin{aligned} h^{z_b} &= h^{\sum_{i=1}^{\ell} z_{i,b}\theta^i} \\ &= \prod_{i=1}^{\ell} (h^{z_{i,b}})^{\theta^i} \\ &= \prod_{i=1}^{\ell} (a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{e_{i,b}})^{\theta^i} \end{aligned}$$

for $b \in \{0,1\}$. Now, it is easy to see that the protocol is complete.

To show that the protocol achieves the honest-verifier zero-knowledge property, we construct a simulator $\mathcal{S}$ to simulate the view of the verifier V. The simulator $\mathcal{S}$ first picks the challenges $e, \theta \leftarrow\!\!\$\, \mathbb{Z}_q$. Then $\mathcal{S}$ selects $e_{i,0} \leftarrow\!\!\$\, \mathbb{Z}$ and computes $e_{i,1} \leftarrow e - e_{i,0} \bmod q$. $\mathcal{S}$ generates $z_{i,b} \leftarrow\!\!\$\, \mathbb{Z}$ and computes

$$a_{i,b}^{(0)} \leftarrow g^{z_{i,b}} (\gamma_{i,b}^{(0)})^{-e_{i,b}}$$

and

$$a_{i,b}^{(1)} \leftarrow g^{z_{i,b}} (\gamma_{i,b}^{(1)})^{-e_{i,b}}$$

for $i \in [\ell]$ and $b \in \{0,1\}$. $\mathcal{S}$ also computes $z_0 \leftarrow \sum_{i=1}^{\ell} z_{i,0}\theta^i$ and $z_1 \leftarrow \sum_{i=1}^{\ell} z_{i,1}\theta^i$. We note that the distribution of

$$(\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell], b\in\{0,1\}}, e, \theta, \{e_{i,b}\}_{i\in[\ell], b\in\{0,1\}}, z_0, z_1)$$

in this simulation is perfectly indistinguishable from that of a real execution. This is due to the fact that given random $e, \theta \in \mathbb{Z}_q$, elements in $\{z_{i,b}\}_{i\in[\ell], b\in\{0,1\}}$ are uniformly random both in a real execution and in the simulation. Meanwhile, the distributions of each pair $(e_{i,0}, e_{i,1})$ satisfying $e_{i,0} + e_{i,1} \equiv e \bmod q$ in a real execution and in the simulation are identical. Conditioned on these values, z_0, z_1, and elements in $\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell], b\in\{0,1\}}$ are uniquely determined by the verification equations. Thus, the distribution of simulated proofs is identical to that of real proofs.

For soundness, we construct an extractor $\mathcal{E}$ that internally runs P^* and executes the protocol with P^*. If the transcript is accepting, $\mathcal{E}$ has to extract a witness. Therefore, $\mathcal{E}$ rewinds P^* to the challenge phase (Step 2) and runs it again with different challenges to obtain ℓ pair of accepting transcripts with the same $\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell], b\in\{0,1\}}$, such that each pair is with different $\{\theta_{[j]}\}_{j\in[\ell]}$, and both transcripts in each pair are with challenges e and $\bar{e}(\neq e)$, respectively. Note that the rewinding scheme follows the strategy used in [6]. Let these pairs be of the form

$$(\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell], b\in\{0,1\}}, e, \theta_{[j]}, \{e_{i,b}^{[j]}\}_{i\in[\ell], b\in\{0,1\}}, \{z_b^{[j]}\}_{b\in\{0,1\}})$$

and

$$(\{(a_{i,b}^{(0)}, a_{i,b}^{(1)})\}_{i\in[\ell], b\in\{0,1\}}, \bar{e}, \theta_{[j]}, \{\bar{e}_{i,b}^{[j]}\}_{i\in[\ell], b\in\{0,1\}}, \{\bar{z}_b^{[j]}\}_{b\in\{0,1\}})$$

for $j \in [\ell]$. Note that the extractor $\mathcal{E}$ will obtain 2ℓ transcripts, and it runs in expected polynomial time. Since $e \neq \bar{e}$, for each $j \in [\ell]$, we must have $e_{i,0}^{[j]} \neq \bar{e}_{i,0}^{[j]}$ or $e_{i,1}^{[j]} \neq \bar{e}_{i,1}^{[j]}$. Let b_i be the value that $e_{i,b_i}^{[j]} \neq \bar{e}_{i,b_i}^{[j]}$ for $i \in [\ell]$. If we have both $e_{i,0}^{[j]} \neq \bar{e}_{i,0}^{[j]}$ and $e_{i,1}^{[j]} \neq \bar{e}_{i,1}^{[j]}$, b_i could be equal to either 0 or 1. According to the accepting transcripts, we have

$$g^{z_b^{[j]}} = \prod_{i=1}^{\ell} (a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{e_{i,b}^{[j]}})^{\theta_{[j]}^i} \ , \quad h^{z_b^{[j]}} = \prod_{i=1}^{\ell} (a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{e_{i,b}^{[j]}})^{\theta_{[j]}^i} \ ,$$

for $j \in [\ell]$ and $b \in \{0,1\}$. Therefore, there should be some $\{z_{i,b}^{[j]}\}_{i\in[\ell]}$, such that

$$a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{e_{i,b}^{[j]}} = g^{z_{i,b}^{[j]}} \text{ for } i \in [\ell] \ , \text{ and } z_b^{[j]} = \sum_{i=1}^{\ell} z_{i,b}^{[j]} \theta_{[j]}^i \ .$$

For the system of equations $z_b^{[j]} = \sum_{i=1}^{\ell} z_{i,b}^{[j]} \theta_{[j]}^i$ for $j \in [\ell]$, we can efficiently solve the unique solution $\{z_{i,b}^{[j]}\}_{i\in[\ell]}$. This is due to the fact that the corresponding Vandermonde matrix of θ is of full rank. It is straightforward to see that this unique solution $\{z_{i,b}^{[j]}\}_{i\in[\ell]}$ should also satisfy $a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{e_{i,b}^{[j]}} = h^{z_{i,b}^{[j]}}$. Hence, we obtain $\{z_{i,b}^{[j]}\}_{i\in[\ell]}$ for $b \in \{0,1\}$. Similarly, we know that

$$g^{\bar{z}_b^{[j]}} = \prod_{i=1}^{\ell} (a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{\bar{e}_{i,b}^{[j]}})^{\theta_{[j]}^i} \ , \quad h^{\bar{z}_b^{[j]}} = \prod_{i=1}^{\ell} (a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{\bar{e}_{i,b}^{[j]}})^{\theta_{[j]}^i}$$

for $j \in [\ell]$. We can use the same approach to computing $\{\bar{z}_{i,b}^{[j]}\}_{i\in[\ell]}$ for $b \in \{0,1\}$, such that

$$a_{i,b}^{(0)} (\gamma_{i,b}^{(0)})^{\bar{e}_{i,b}^{[j]}} = g^{\bar{z}_{i,b}^{[j]}} \ , \quad a_{i,b}^{(1)} (\gamma_{i,b}^{(1)})^{\bar{e}_{i,b}^{[j]}} = h^{\bar{z}_{i,b}^{[j]}} \ , \quad \text{and } \bar{z}_b^{[j]} = \sum_{i=1}^{\ell} \bar{z}_{i,b}^{[j]} \theta_{[j]}^i \ .$$

Given a pair of equations $a_{i,b_i}^{(0)} (\gamma_{i,b_i}^{(0)})^{e_{i,b_i}^{[j]}} = g^{z_{i,b_i}^{[j]}}$ and $a_{i,b_i}^{(0)} (\gamma_{i,b_i}^{(0)})^{\bar{e}_{i,b_i}^{[j]}} = g^{\bar{z}_{i,b_i}^{[j]}}$, there should be some x_{i,b_i} and ν_{i,b_i}, such that

$$a_{i,b_i}^{(0)} = g^{x_{i,b_i}} \ , \quad \gamma_{i,b_i}^{(0)} = g^{\nu_{i,b_i}} \ ,$$

and

$$x_{i,b_i} + \nu_{i,b_i} e_{i,b_i} = z_{i,b_i} \ , \quad x_{i,b_i} + \nu_{i,b_i} \bar{e}_{i,b_i} = \bar{z}_{i,b_i} \ .$$

According to the assignment of b_i, we know $e_{i,b_i} \neq \bar{e}_{i,b_i}$. Thus, the extractor $\mathcal{E}$ can easily compute x_{i,b_i} and ν_{i,b_i} from the last two equations, and finally obtain $\{x_{i,b_i}\}_{i\in[\ell]}$ and $\{\nu_{i,b_i}\}_{i\in[\ell]}$ from pairs of equations for all $i \in [\ell]$. It is easy to verify that these extracted elements also satisfy

$$a_{i,b_i}^{(1)} = h^{x_{i,b_i}} \ , \quad \gamma_{i,b_i}^{(1)} = h^{\nu_{i,b_i}} \ .$$

Hence, the extractor $\mathcal{E}$ successfully extracts $\{\nu_{i,b_i}\}_{i\in[\ell]}$, where $b_i \in \{0,1\}$ and $\nu_{i,b_i} \in \mathbb{Z}_q$, such that $\gamma_{i,b_i}^{(0)} = g^{\nu_{i,b_i}}$ and $\gamma_{i,b_i}^{(1)} = h^{\nu_{i,b_i}}$ for all $i \in [\ell]$. The soundness of the protocol follows. □

5 Analysis

In this section, we analyze the performance of our protocol. In Table 1, we present the communication cost for one execution of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ with parameters M, N and $N' = \max(M, N)$. We give the communication cost of the two executions of the sub-protocol $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ inside $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$, respectively. The row of "remaining" is for the communication cost of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ excluding the cost of sub-protocols $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{DH}}$. We note that $\|\mathbb{G}\| > \|\mathbb{Z}_q\|$.

Table 1: Communication cost of each part in our protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ with parameter M, N and $N' = \max(M, N)$.

Protocol	From P to V	From V to P
1st $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ [3]	$(11\sqrt{N'} + 5)\|\mathbb{G}\| + (5\sqrt{N'} + 9)\|\mathbb{Z}_q\|$	$8\|\mathbb{Z}_q\|$
2nd $\Pi_{\mathsf{zk}}^{\mathsf{Shuffle}}$ [3]	$(11\sqrt{N} + 5)\|\mathbb{G}\| + (5\sqrt{N} + 9)\|\mathbb{Z}_q\|$	$8\|\mathbb{Z}_q\|$
$\Pi_{\mathsf{zk}}^{\mathsf{DH}}$	$4N\|\mathbb{G}\| + (2N + 2)\|\mathbb{Z}_q\|$	$2\|\mathbb{Z}_q\|$
Remaining	$(2N' + 2N)\|\mathbb{G}\|$	0

In Table 2, we then present the comparison of communication cost between our protocol and the previous protocol [25] (in the honest-verifier zero-knowledge setting). Here our comparison follows the fact that the parameters satisfy $N > M$ in most applications of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$. Therefore, we simply let $N' = \max(M, N) = N$ in the comparison. We remark that the protocol in [25] is not public-coin, and interaction is needed for the protocol execution. Alternatively, in our protocol, all messages sent from the verifier are uniformly random, *i.e.*, the protocol is public-coin. Therefore, we can simply leverage the Fiat-Shamir heuristic to make our protocol non-interactive. Now the communication cost of our protocol only involves the bits sent from the prover P to the verifier V. From Table 2, we can see that the (non-interactive) communication cost of our protocol is around $8N\|\mathbb{G}\|$, while the total communication cost of the (interactive) protocol in [25] is around $34N\|\mathbb{G}\|$ bits. Therefore, the communication cost of our protocol is only around 24% of that of the protocol in [25].

Table 2: Communication cost comparison between the original protocol [25] and the protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ is this paper with parameters M and N.

Protocol	From P to V	From V to P
[25]	$\sim (32N\|\mathbb{G}\| + 12N\|\mathbb{Z}_q\|)$	$\sim (2N\|\mathbb{G}\| + 10N\|\mathbb{Z}_q\|)$
This paper	$\sim ((8N + 22\sqrt{N})\|\mathbb{G}\| + (2N + 10\sqrt{N})\|\mathbb{Z}_q\|)$	$18\|\mathbb{Z}_q\|$

Note that in our protocol, we use the protocol in [3] as the zero-knowledge protocol for shuffle twice, while the protocol in [25] adopts the zero-knowledge

Table 3: Comparison of computation cost between the original protocol [25] and the protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ in this paper with parameters M and N except zero-knowledge protocols for shuffle.

Protocols	Time P Expos	Time V Expos
[25]	$\sim 37N$	$\sim 32N$
This paper	$\sim 10N$	$\sim 4N$

protocol for shuffle introduced in [10] twice (for ElGamal ciphertext list of the same length N). We denote the protocols in [3] and [10] by BG and FS, respectively. It is shown [3] that BG significantly outperforms FS from both communication and computation aspects. According to the analysis in [3], BG's argument size is only $1/94$ that of FS's, and BG has $3.4\times$ faster running time. We count the total number of exponentiations in $\mathbb{G}$ performed by P and V for the original protocol [25] and our protocol, except those performed by the zero-knowledge protocols for shuffle, in Table 3. We can see that without considering the zero-knowledge protocols for shuffle, the computation cost of our protocol is about 27% of that of the protocol in [25] for provers and 12.5% of that for verifiers. Therefore, our protocol should be much faster than the original protocol in [25]. In addition, we would like to note that the communication cost from FS in [10] is around $(10N\|\mathbb{G}\| + 4N\|\mathbb{Z}_q\|)$ bits. This means that the communication cost of our whole protocol outperforms that of the protocol in [10] even when the communication cost of FS in [10] is not considered.

Acknowledgments. We thank the reviewers for their detailed and helpful comments. Y. Liu and Q. Wang partially supported by the Shenzhen fundamental research programs under Grant no. 20200925154814002 and Guangdong Provincial Key Laboratory (Grant No. 2020B121201001). Y. Liu and S.-M. Yiu were also partially supported by ITF, Hong Kong (ITS/173/18FP).

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. J. Cryptol. **2**(1), 1–12 (1990)
2. Alhassan, M.Y., Günther, D., Kiss, Á., Schneider, T.: Efficient and scalable universal circuits. J. Cryptol. **33**(3), 1216–1271 (2020)
3. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 263–280. Springer (2012)
4. Bicer, O., Bingol, M.A., Kiraz, M.S., Levi, A.: Highly efficient and re-executable private function evaluation with linear complexity. IEEE Transactions on Dependable and Secure Computing pp. 1–1 (2020)
5. Bingöl, M.A., Biçer, O., Kiraz, M.S., Levi, A.: An efficient 2-party private function evaluation protocol based on half gates. Comput. J. **62**(4), 598–613 (2019)

6. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 327–357. Springer (2016)
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Lecture Notes in Computer Science, vol. 740, pp. 89–105. Springer (1992)
8. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 174–187. Springer (1994)
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
10. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 368–387. Springer (2001)
11. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984)
12. Goldreich, O.: The Foundations of Cryptography - Volume 1: Basic Techniques. Cambridge University Press (2001)
13. Günther, D., Kiss, Á., Schneider, T.: More efficient universal circuit constructions. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 443–470. Springer (2017)
14. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010)
15. Holz, M., Kiss, Á., Rathee, D., Schneider, T.: Linear-complexity private function evaluation is practical. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 401–420. Springer (2020)
16. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. CRC Press (2014)
17. Katz, J., Malka, L.: Constant-round private function evaluation with linear complexity. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 556–571. Springer (2011)

18. Kiss, Á., Schneider, T.: Valiant's universal circuit is practical. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 699–728. Springer (2016)
19. Kolesnikov, V., Schneider, T.: A practical universal circuit construction and secure evaluation of private functions. In: Tsudik, G. (ed.) Financial Cryptography and Data Security, 12th International Conference, FC 2008, Cozumel, Mexico, January 28-31, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5143, pp. 83–97. Springer (2008)
20. Laud, P., Willemson, J.: Composable oblivious extended permutations. In: Cuppens, F., García-Alfaro, J., Zincir-Heywood, A.N., Fong, P.W.L. (eds.) Foundations and Practice of Security - 7th International Symposium, FPS 2014, Montreal, QC, Canada, November 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8930, pp. 294–310. Springer (2014)
21. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology **16**(3), 143–184 (2003)
22. Lipmaa, H., Mohassel, P., Sadeghian, S.S.: Valiant's universal circuit: Improvements, implementation, and applications. IACR Cryptol. ePrint Arch. **2016**, 17 (2016), http://eprint.iacr.org/2016/017
23. Liu, H., Yu, Y., Zhao, S., Zhang, J., Liu, W.: Pushing the limits of valiant's universal circuits: Simpler, tighter and more compact. IACR Cryptol. ePrint Arch. **2020**, 161 (2020), https://eprint.iacr.org/2020/161
24. Mohassel, P., Sadeghian, S.S.: How to hide circuits in MPC an efficient framework for private function evaluation. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 557–574. Springer (2013)
25. Mohassel, P., Sadeghian, S.S., Smart, N.P.: Actively secure private function evaluation. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 486–505. Springer (2014)
26. Valiant, L.G.: Universal circuits (preliminary report). In: Chandra, A.K., Wotschke, D., Friedman, E.P., Harrison, M.A. (eds.) Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA. pp. 196–203. ACM (1976)
27. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. pp. 160–164. IEEE Computer Society (1982)
28. Zhao, S., Yu, Y., Zhang, J., Liu, H.: Valiant's universal circuits revisited: An overall improvement and a lower bound. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 401–425. Springer (2019)