Making Private Function Evaluation Safer, Faster, and Simpler

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

[1] Research Institute of Trustworthy Autonomous Systems & 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. In the problem of two-party *private function evaluation* (PFE), one party $\mathsf{P_A}$ holds a *private function* f and (optionally) a private input x_A, while the other party $\mathsf{P_B}$ possesses a private input x_B. Their goal is to evaluate f on x_A and x_B, and one or both parties may obtain the evaluation result $f(x_A, x_B)$ while no other information beyond $f(x_A, x_B)$ is revealed.

In this paper, we revisit the two-party PFE problem and provide several enhancements. We propose the *first* constant-round actively secure PFE protocol with linear complexity. Based on this result, we further provide the *first* constant-round publicly verifiable covertly (PVC) secure PFE protocol with linear complexity to gain better efficiency. For instance, when the deterrence factor is $\epsilon = 1/2$, compared to the passively secure protocol, its communication cost is very close and its computation cost is around 2.6×. In our constructions, as a by-product, we design a specific protocol for proving that a list of ElGamal ciphertexts is derived from an *extended permutation* performed on a given list of elements. It should be noted that this protocol greatly improves the previous result and may be of independent interest. In addition, a reusability property is added to our two PFE protocols. Namely, if the same function f is involved in multiple executions of the protocol between $\mathsf{P_A}$ and $\mathsf{P_B}$, then the protocol could be executed more efficiently from the second execution. Moreover, we further extend this property to be *global*, such that it supports multiple executions for the same f in a reusable fashion between $\mathsf{P_A}$ and *arbitrary* parties playing the role of $\mathsf{P_B}$.

Keywords: Extended permutation · Private function evaluation · Publicly verifiable covert security · Secure two-party computation.

1 Introduction

The two-party *private function evaluation* (PFE) problem considers the scenario where a party $\mathsf{P_A}$ holds a *private function* f and (optionally) a private input x_A while the other party $\mathsf{P_B}$ has another private input x_B. These two parties intend to compute $f(x_A, x_B)$ without the existence of a third party. Finally, one or both parties may obtain $f(x_A, x_B)$, while they cannot deduce any other information beyond their specified outputs during the interaction. As a special case of secure computation, note that PFE is different from the notion of standard *secure function evaluation* (SFE). The key difference is that the function f is *commonly known* by participants in SFE, while f should *remain private* in PFE, in the sense that *everything* about the function, except an upper bound on its size and the lengths of both input and output, is hidden.

Both data and algorithms are valuable in numerous real-world scenarios, such as medical and commercial applications. For instance, we consider the following business scenario between a traditional enterprise and an algorithm-driven company. The traditional enterprise has a dataset, while the algorithm-driven company holds a powerful data mining algorithm that can process this dataset. On the one hand, the algorithm-driven company does not intend to disclose the algorithm. On the other hand, since the dataset may contain sensitive data, the traditional enterprise is unwilling to reveal the dataset to others. We note that this dilemma can be solved by a PFE protocol that allows the traditional enterprise to receive the result of privately running the algorithm on the dataset.

It is trivial to design a PFE protocol based on fully homomorphic encryption (FHE) schemes [17]. However, the efficiency of FHE schemes is still prohibitive, and researchers attempted to design PFE in the setting of traditional multi-party computation (MPC). In the literature, some PFE protocols specify a *limited* set of functions, such as polynomials [13, 35, 32] and low-depth circuits [38], while others are *general-purpose*, focusing on functions implemented by arbitrary (polynomial-size) circuits [1]. In this paper, we work on general-purpose PFE protocols, and thus the PFE protocols mentioned in the rest of this paper are assumed to be general-purpose.

To construct general-purpose PFE protocols, there exist two main approaches. The first approach reduces the PFE problem to the problem of secure computation for universal circuits (UC) (see [40, 27, 25, 30, 19, 42, 2, 31]). UC refers to a sequence of circuits $\mathsf{U} = \{\mathsf{U}_n\}_{n\in\mathbb{N}}$, each of which can take as input (the description of) a circuit C of size n and a valid input x, and output $C(x) \leftarrow \mathsf{U}_n(C, x)$. Therefore, we can combine UC with traditional MPC techniques, such as Yao's garbled circuits [41, 29], to obtain PFE protocols. The major goal of this line of work is to reduce the size of UC and improve the traditional MPC techniques. However, a noted barrier of UC-based PFE protocol is that a (Boolean) UC has (optimal) size $|\mathsf{U}_n| = \Theta(n \log n)$ [40], where the constant factor (more than 12 for the state-of-the-art result [31]) and the low-order terms are significant. Hence, when the size of a circuit used for evaluation is relatively large, the considerable expansion of its size caused by the use of UC makes UC-based PFE prohibitive.

The second approach avoids the usage of UC. In 2011, Katz and Malka [23] proposed a constant-round passively secure two-party PFE protocol applied on Boolean circuits, and the protocol achieves linear complexity in circuit size. This linear-complexity PFE protocol has asymptotically less computation and communication complexity than UC-based PFE protocols that have complexity $\Theta(n \log n)$. Very recently, an implementation [21] of the passively secure PFE protocol [23] showed that this protocol outperforms the state-of-the-art UC-based PFE protocol not only in communication but also in total running time, *e.g.*, it is $\sim 3.3\times$ faster in a LAN and $\sim 7.0\times$ faster in a WAN for private circuits of size 10^6. Subsequently, the work [33] introduced a general framework for designing PFE protocols. This general framework captures the idea of [23] and provides a slight improvement in communication cost. In addition, a PFE protocol based on *oblivious evaluation of switching networks* (OSN) was provided in [33] and was later improved in [9]. However, it is shown [2, 8] that OSN-based PFE protocols have $\Theta(n \log n)$ computation and communication complexities limit their usage when the size of circuits is considerable. More recently, a passively secure re-executable PFE protocol with linear complexity was proposed in [8]. With this reusability property, it is shown [8] that this protocol has significantly better performance than the PFE protocol in [23] and [33] when the protocol is executed any number (more than one) of times for the same function by the same two parties.

Since parties may deviate from the protocol to gain more advantages, such as learning the other party's input and affecting the output of the protocol, it is more realistic to consider PFE protocols that are secure under stronger security models. Unfortunately, even though the line of work for PFE protocols with linear complexity has better performance theoretically and experimentally, existing protocols are mainly focused on the semi-honest model, and very few results managed to provide protocols in stronger security models.

To the best of our knowledge, only two papers considered PFE protocols with linear complexity that are secure against malicious adversaries. The seminal work [23] introduced how to compile their passively secure PFE protocol to be secure against malicious $\mathsf{P_B}$, *i.e.*, the party that provides the private input x_B, via specific efficient zero-knowledge protocols. However, the security of the compiled protocol is not full-fledged, and the function provider $\mathsf{P_A}$ is required to be semi-honest. The subsequent work [34] proposed an actively secure PFE framework with linear complexity based on the results in [33]. However, the number of rounds in this protocol is equal to the number of gates for the evaluated circuit. This will simply become a bottleneck when the size of the circuit is considerable.

Besides the malicious model, there is no PFE protocol with linear complexity in other security models. We notice that the *publicly verifiable covert* (PVC) model is particularly useful for many scenarios that PFE protocols may apply to. *Covert security* was introduced by Aumann and Lindell [4]. It serves as a compromise between semi-honest and malicious security definitions, and thereby provides a more realistic security guarantee than semi-honest security and has significantly less overhead than malicious security. Informally, a malicious party is still allowed to covertly deviate from the protocol execution in this model. However, this misbehavior will be detected by honest parties with a certain probability ϵ, which is called *deterrence factor*. The fear of being caught will deter participants from acting maliciously and deviating from the protocol. The PVC security notion that enhances the covert security model was introduced by Asharov and Orlandi in 2012 [3]. PVC security guarantees that once the misbehavior of a malicious party is caught, honest parties could generate a publicly verifiable certificate to persuade others, including those outside the protocol, that the malicious party is cheating. Meanwhile, it should be guaranteed that this malicious party learns no information about the inputs of honest parties even when the certificate is given. This notion significantly strengthens the covert security model especially when parties' reputations are important. A general PVC-secure two-party computation protocol was proposed in [3] based on garbled circuits and the Signed-OT technique. Then the Signed-OT protocol was improved in [26] to obtain a more efficient PVC-secure protocol. Subsequently, an elegant protocol [22] using a derandomized approach was proposed in 2019. Avoiding the use of costly Signed-OT, this protocol is more efficient than the previous protocols. In the meantime, another protocol [43] introduced a notion called financially secure computation that combines a PVC-secure protocol with blockchain. Very recently, a compiler that can transform a two-party passively secure protocol to a PVC-secure protocol was introduced in [15]. It is easy to see that PVC security is useful for two-party PFE protocols in many realistic scenarios. Note that all existing results for two-party PVC security [3, 26, 22, 15] are only designed for SFE, *i.e.*, the function f is *publicly known*. Although UC can be integrated into these frameworks to derive a PVC-secure PFE protocol, so far there is no PVC-secure PFE protocol with linear complexity.

Therefore, the following question is open so far:

> *Can we construct a constant-round actively secure and a constant-round PVC-secure PFE protocols with linear complexity in the two-party setting while avoiding strong primitives such as FHE?*

In this paper, we answer this question. In addition, we borrow the idea of [8] to realize a reusability property for our protocols and further extend it *globally*. A comparison of main properties for all PFE protocols *with linear complexity* is summarized in Table 1.

1.1 Our Results

We summarize our results and main contributions in this paper as follows.

Active security. We provide the *first* constant-round actively secure PFE protocol with linear complexity in the two-party setting. More precisely, we design a constant-round two-party PFE protocol that is secure against malicious function owner $\mathsf{P_A}$ and semi-honest private input provider $\mathsf{P_B}$. Then by leveraging classical MPC results for security against malicious $\mathsf{P_B}$ providing private input values, such as the approach used in [23], we can automatically

Table 1: Comparison of the main properties for all PFE protocols with linear complexity.

Paper	Security	# Round	Reusable?
[23]	Passive	Constant	No.
[33]	Passive	Constant	No.
[34]	Active	# Gates	No.
[8]	Passive	Constant	Yes, for two parties.
This paper	Active	Constant	Yes, global reusability.
This paper	PVC	Constant	Yes, global reusability.

obtain the desirable actively secure protocol. Our protocol is composed of an *initiation* phase and an *evaluation* phase.

PVC security. Based on the techniques of our actively secure PFE protocol, we design the *first* constant-round PVC-secure PFE protocol with linear complexity in the two-party setting to gain much better efficiency. This protocol inherits the two-phase structure. It is noted that the additional overhead to achieve PVC security is very light from both computation and communication aspects, *e.g.*, when the deterrence factor is $\epsilon = 1/2$, compared to the passively secure protocol, its communication cost is very close and its computation cost is around 2.6×.

Efficiency improvement. We provide the sub-protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ as a core component for our actively secure and PVC-secure protocols. This protocol is designed for proving that a list of ElGamal [16] ciphertexts is derived from an *extended permutation* (see Definition 3) performed on a given list of elements. A generic construction for such a purpose was originally given in [34], and it is left open whether it is possible to construct such a protocol in a specific approach to gaining better performance. Our protocol answers this open problem, and improves the generic construction [34] significantly: the communication cost of our protocol is less than 1/56 of the generic construction, and the computation cost is less than 36%.

Reusability (simplified follow-up executions). The reusability property is added to both of our two PFE protocols. When two specified parties intend to evaluate the same private function f on different private inputs, they only need to go through the initiation phase at one time and then execute the evaluation phase multiple times with different inputs. Moreover, we extend this property *globally*. Namely, once an initiation for a private f is performed by the function owner $\mathsf{P_A}$, *arbitrary* private input providers playing the role of $\mathsf{P_B}$ can benefit from the reusability property for f.

2 Preliminaries

We use $|S|$ to denotes the size of a set S and $\|S\|$ to denote the number of bits required to represent elements in the set S. We write $x \leftarrow_{\$} S$ for uniformly sampling an element x from the set S. For a positive integer n, let $[n] = \{1, \ldots, n\}$. For a bit string x, we use $x[i]$ to represent the ith bit of x. We write a vector named a as $\vec{a} = (a_1, \ldots, a_n)$, and use $\vec{0}$ and $\vec{1}$ to denote a vector where all entries are equal to 0 and 1 when its dimension is clear in the context, respectively. Let $\vec{a}\vec{b} = (a_1 b_1, \ldots, a_n b_n)$ denote the Hadamard product of two vectors $\vec{a}$ and $\vec{b}$, $\vec{a} \circ \vec{b} = (a_1, \ldots, a_{n_a}, b_1, \ldots, b_{n_b})$ the concatenation of vectors, $\vec{a}^{\mathrm{T}}\vec{b} = \sum_i a_i b_i$ the inner product, and $\vec{g}^{\vec{a}} = \prod_i g_i^{a_i}$ the multi-exponentiation. For a scalar c and a vector $\vec{a}$, the scalar product is $c\vec{a} = (ca_1, \ldots, ca_n)$.

Let κ be the computational security parameter, and κ is written in unary as input to all algorithms. 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 say that $1 - f$ is *overwhelming* if f is negligible.

Given a $\mathsf{seed} \in \{0, 1\}^\kappa$, we can use a pseudorandom function with seed as the key in the CTR mode to derive sufficiently many pseudorandom numbers and use them as random coins for operations in protocols.

We use Com to denote the (non-interactive) commitment scheme. We write decom as the random coins for a commitment, which can be used to open this commitment. The commitment scheme Com achieves (computationally) binding and hiding properties. We will use a signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ that is existentially unforgeable under chosen-message attacks (EUF-CMA) for our PVC-secure protocol in Section 4.

The oblivious transfer (OT) functionality $\mathcal{F}_{\mathsf{OT}}$ is presented below. Let Π_{OT} be the protocol that securely realizes a parallel version of $\mathcal{F}_{\mathsf{OT}}$.

Functionality $\mathcal{F}_{\mathsf{OT}}$

Private inputs: $\mathsf{P_A}$ has input $x \in \{0, 1\}^\lambda$ and $\mathsf{P_B}$ has input $\{(A_{i,0}, A_{i,1})\}_{i \in [\lambda]}$.

Upon receiving $x \in \{0, 1\}^\lambda$ from $\mathsf{P_A}$ and $\{(A_{i,0}, A_{i,1})\}_{i \in [\lambda]}$ from $\mathsf{P_B}$, the functionality sends $\{A_{i,x[i]}\}_{i \in [\lambda]}$ to $\mathsf{P_A}$.

The security of our protocol relies on the decisional Diffie-Hellman (DDH) assumption as follows.

Definition 1. *The decisional Diffie-Hellman (DDH) assumption in a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order $q \in \Theta(2^\kappa)$ is that given (g^a, g^b) for $a, b \leftarrow_\$ \mathbb{Z}_q$, g^{ab} is computationally indistinguishable from a random element in $\mathbb{G}$.*

Under the DDH assumption, we have the following lemma.

Lemma 1 ([36]). *Under the DDH assumption for the cyclic group $\mathbb{G}$ of prime order $q \in \Theta(2^\kappa)$, for any positive integer $n = \mathsf{poly}(\kappa)$, given $g_1, \ldots, g_n \leftarrow_\$ \mathbb{G}$, we have that $(g_1^{\alpha_1}, \ldots, g_n^{\alpha_n})$ is computationally indistinguishable from $(g_1^{\alpha}, \ldots, g_n^{\alpha})$ for $\alpha, \alpha_1, \ldots, \alpha_n \leftarrow_\$ \mathbb{Z}_q$.*

It is well-known that the DDH assumption implies the discrete logarithm assumption, which is equivalent to the following assumption.

Definition 2. *The discrete logarithm relation assumption in a cyclic group $\mathbb{G}$ of prime order $q \in \Theta(2^\kappa)$ is that for any positive integer $n = \mathsf{poly}(\kappa)$, given*

$g_1, \ldots, g_n \leftarrow_\$ \mathbb{G}$, *it is computationally hard to find* $a_1, \ldots, a_n \in \mathbb{Z}_q$, *such that* $\exists a_i \neq 0 \in \mathbb{Z}_q \wedge \prod_{i=1}^{n} g_i^{a_i} = 1$. *We call* $\prod_{i=1}^{n} g_i^{a_i} = 1$ *a nontrivial discrete logarithm relation.*

We use the ElGamal encryption scheme in our protocol. This encryption scheme is over the cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, and it is indistinguishable under chosen plaintext attack (IND-CPA) under the DDH assumption for $\mathbb{G}$. We provide the description of algorithms for the scheme as follows.

Key Generation. This algorithm 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. This algorithm 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 $r \leftarrow_\$ \mathbb{Z}_q$.

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

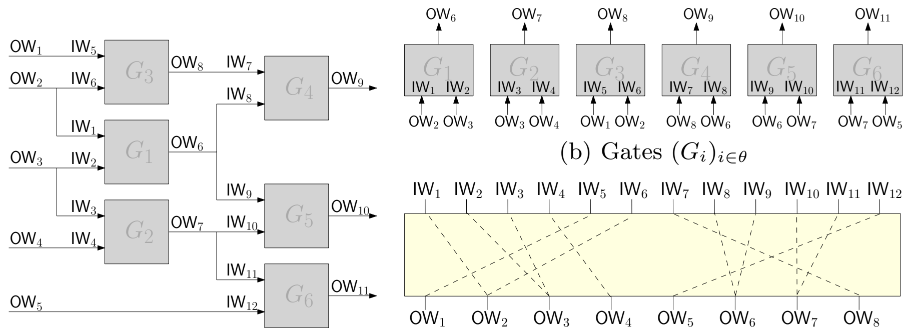
The ElGamal encryption scheme is multiplicatively homomorphic, such that the multiplication result of two ciphertexts is the ciphertext of the multiplication result of the two corresponding plaintexts. Computing the power of a ciphertext c also derives the ciphertext for the power of the corresponding plaintext of c.

2.1 Circuit Representation and Extended Permutation

Here, we introduce an approach to describing Boolean circuits with arbitrary fan-out (see an example circuit C_f in Fig. 1). For a circuit, we call a wire *outgoing wire* (denoted by OW) if it is an input wire of the circuit or output wire of a gate. Meanwhile, a wire is called *incoming wire* (denoted by IW) if it is the input wire of a gate. Outgoing wires are connected with incoming wires, in the sense that each incoming wire connects with exactly one outgoing wire while an outgoing wire may connect with an arbitrary number (including 0) of incoming wires. Suppose that a circuit consists of θ gates, n input bits, and m output bits. Then this circuit has $n + \theta$ outgoing wires and 2θ incoming wires. For a gate G_i, its output wire is the outgoing wire OW_{n+i} and its two input wires are the incoming wires IW_{2i-1} and IW_{2i}. The last m gates are the output gates of the circuit. Fig. 1(b) lists all gates $(G_i)_i$ inside the circuit C_f. A formal description of the connections between incoming wires and outgoing wires is captured by [33] via extended permutation.

Definition 3 ([33]). *For positive integers N and M, a mapping $\pi : [N] \to [M]$ is an extended permutation (EP) if for every $x \in [N]$, there exists one $y \in [M]$, such that $y = \pi(x)$.*

Given an index of an incoming wire, π maps it to the index of the outgoing wire that connects with this incoming wire (see example in Fig. 1(c)). Note that different from the one-to-one correspondence mapping of the standard permutation, EP allows to replicate or omit elements during the mapping.



(a) Circuit C_f assembled by $(G_i)_{i\in\theta}$ (b) Gates $(G_i)_{i\in\theta}$ (c) Wire connections and EP π_f

Fig. 1: A circuit C_f and the illustration of its wire connections and EP π_f.

Given a set of gates $(G_i)_{i\in[\theta]}$, the circuit owner $\mathsf{P_A}$ holding the description of a circuit C_f can follow the (randomized) procedure below to assign $(G_i)_{i\in[\theta]}$ to positions of gates in C_f and derive an EP π_f from the resulting circuit assembled by this set of gates.

1. Sort indices for non-output gate positions of C_f in a topological order, such that if the output wire of the ith gate is connected with the input wire of the jth gate, then i must be smaller than j. The indices of output gates are from $\theta - m + 1$ to θ.
2. Pick a random standard permutation π_R. For non-output gates with indices $i \in [\theta - m]$, the position for the ith gate of C_f is assigned to gate $G_{\pi_R(i)}$.
3. For all output gates with indices $i = \theta - m + 1, \ldots, \theta$, assign gate G_i to the position of the ith gate.
4. Extract the EP π_f for connections of outgoing wires and incoming wires from the resulting circuit.

When we consider a circuit that only includes one type of gates, *e.g.*, NAND gates, the circuit can be exactly described by the corresponding EP. Now given π_f, it is easy to derive the description of the circuit. Our protocol indeed leverages this idea and assumes that circuits only consist of NAND gates for simplicity.

2.2 Building Blocks

In Table 2, we present two zero-knowledge ideal functionalities $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ associated with the relations R_{DH} and R_{EncEP} for the cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q as building blocks for our protocols. We will introduce how to instantiate them in Section 3.

3 PFE Protocol for Active Security

In this section, we introduce our constant-round two-party PFE protocol. This protocol is secure against malicious $\mathsf{P_A}$ and semi-honest $\mathsf{P_B}$. Note that it is

Table 2: Relations and their zero-knowledge ideal functionalities.

Relation	Functionality
$R_{\mathsf{DH}} = \{(\mathbb{G}, q, \{g_i\}_{i\in[N]}, \{h_i\}_{i\in[N]}) \mid \exists x, s.t. \bigwedge_{i\in[N]}(h_i = g_i^x)\}$	$\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$
$R_{\mathsf{EncEP}} = \{(\mathbb{G}, q, g, h, \{g_i\}_{i\in[M]}, \{(c_i^{(0)}, c_i^{(1)})\}_{i\in[N]}) \mid \exists \{r_i\}_{i\in[N]}, \pi, s.t.$ $c_i^{(0)} = g^{r_i} \wedge c_i^{(1)} = g_{\pi(i)}h^{r_i} \wedge \pi$ is an EP. $\}$	$\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$

straightforward to obtain a constant-round actively secure PFE protocol with linear complexity by leveraging classical MPC results, such as the approach used in [23], to compile the protocol to be secure against malicious (circuit grabler) $\mathsf{P_B}$ providing private input values.

In PFE, a party $\mathsf{P_A}$ has a private Boolean circuit input C_f (implementing a function f) and private input $x_A \in \{0,1\}^{n_A}$, whereas the other party $\mathsf{P_B}$ has private input $x_B \in \{0,1\}^{n_B}$. We present the ideal functionality $\mathcal{F}_{\mathsf{activePFE}}$ for our protocol in the following. Here we consider the more general case that the circuit holder $\mathsf{P_A}$ has an input $x_A \in \{0,1\}^{n_A}$, and it is not difficult to modify the protocols to the case that $\mathsf{P_A}$ has the private input C_f only. For the sake of simplicity, we assume that only one party will receive the evaluation result. It is also possible to modify the protocol such that both parties can receive the final result (see [20, Section 2.5.2]).

Functionality $\mathcal{F}_{\mathsf{activePFE}}$

Pre-agreement: The circuit consists of θ gates, m output wires, and $n(= n_A + n_B)$ input wires.
Private inputs: $\mathsf{P_A}$ has a Boolean circuit input C_f and input $x_A \in \{0,1\}^{n_A}$, whereas the other party $\mathsf{P_B}$ has input $x_B \in \{0,1\}^{n_B}$.

1. If an input of the form abort_i from the party P_i for $i \in \{A, B\}$ is received, the ideal functionality sends $\perp$ to both parties and terminates.
2. If an input circuit C_f satisfying the pre-agreement from $\mathsf{P_A}$ is received, store C_f.
3. If $x_A \in \{0,1\}^{n_A}$ from $\mathsf{P_A}$ and $x_B \in \{0,1\}^{n_B}$ from $\mathsf{P_B}$ are received and an input circuit C_f is stored, the ideal functionality computes $C_f(x_A, x_B)$.
   (a) If P_i (which is corrupted by $\mathcal{A}$) is allowed to learn $C_f(x_A, x_B)$, then it sends $C_f(x_A, x_B)$ to P_i.
   (b) Otherwise, the ideal functionality sends $\mathsf{nothing}$ to the corrupted P_i. Then if the message $\mathsf{continue}$ from $\mathcal{A}$ is received, the ideal functionality sends $C_f(x_1, x_2)$ to the honest party. Otherwise, if abort_i is received from $\mathcal{A}$ on behalf of the corrupted P_i, it sends $\perp$ to the honest party.

3.1 Full Description of the Protocol

We now give a full description of our protocol $\mathit{\Pi}_{\mathsf{activePFE}}$. Our protocol consists of two phases: initiation and evaluation. In the initiation phase, two parties prepare required data for later evaluations of C_f. Then given the preprocessed data from the initiation phase, $\mathsf{P_A}$ and $\mathsf{P_B}$ evaluate C_f on their inputs x_A and x_B in the evaluation phase. At the end of the protocol, parties obtain their outputs specified by $\mathcal{F}_{\mathsf{activePFE}}$, *i.e.*, the evaluation result $C_f(x_A, x_B)$ or $\mathsf{nothing}$. For the first execution of the protocol, $\mathsf{P_A}$ and $\mathsf{P_B}$ together execute the initiation phase and evaluation phase sequentially. Then, if the two parties would like to evaluate the same circuit C_f on different inputs, they now only need to execute the evaluation phase using the information previously generated in the initiation phase. This reusability property will be further extended to global reusability (see Remark 2). Note that in our protocols, we consider the Boolean circuit C_f only consists of NAND gates for simplicity. We use the cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q as above.

Here, we briefly present the main flow of the protocol. In the initiation phase, $\mathsf{P_A}$ derives an EP from her private circuit C_f and establishes connections of wire labels between incoming and outgoing wires, while $\mathsf{P_B}$'s tasks are to assist $\mathsf{P_A}$ and ensure that $\mathsf{P_A}$ honestly follows the protocol. Then in the evaluation phase, different from the standard paradigm of garbled circuits, we let $\mathsf{P_B}$ *obliviously* garble (all gates of) the circuit for $\mathsf{P_A}$. Then $\mathsf{P_A}$ can evaluate the corresponding garbled circuit, since she knows the topology of her circuit and the connections of wire labels established in the initiation phase.

In this initiation phase, $\mathsf{P_B}$ first chooses a list G of $M = n + \theta - m$ different elements from $\mathbb{G}$ and sends G to $\mathsf{P_A}$. This list G will be used to derive the labels of outgoing wires except those that are output wires of the circuit. After receiving the list G, $\mathsf{P_A}$ generates an ElGamal encryption key pair. Then $\mathsf{P_A}$ derives the EP π_f from the circuit C_f following the procedure in Section 2.1. Now $\mathsf{P_A}$ performs the EP π_f on G and encrypts all elements of the resulting list to obtain the list Φ, where the ith encrypted elements in Φ are of the form $g_{\pi(i)}$. The list Φ is then sent to $\mathsf{P_B}$. The EP here is to establish the connections between the outgoing wires (except output wires of the circuit since they do not connect with other wires) and the incoming wires for the further generation of wire labels, and the resulting list is encrypted to hide the EP from $\mathsf{P_B}$. Then $\mathsf{P_A}$ picks a list $T = [t_1, \ldots, t_N]$ for $t_i \in \mathbb{Z}_q$ as the *blinding factors*. Using the homomorphic property, $\mathsf{P_A}$ can compute the t_ith power of the plaintext of c_i for all c_i's in Φ and obtain the resulting list Φ', where the ith element is the encryption of $g_{\pi_f(i)}^{t_i}$. We note that here t_i is used to blind the encrypted values in Φ, such that $\mathsf{P_B}$ still does not know the base $g_{\pi_f(i)}$ when the element $g_{\pi_f(i)}^{t_i}$ is given later, and thus π_f and C_f are hidden. Finally, $\mathsf{P_A}$ helps $\mathsf{P_B}$ to decrypt all elements of Φ' to derive $P = [p_1, \ldots, p_N]$, where $p_i = g_{\pi_f(i)}^{t_i}$. In Fig. 2, we give an illustration of the procedure that the circuit owner $\mathsf{P_A}$ will go through in the initiation phase for the previous example (Fig. 1).

During this procedure, to gain active security, it is important that $\mathsf{P_A}$ should prove in zero-knowledge that her operations are valid using the building blocks in Section 2.2. After the initiation phase, $\mathsf{P_B}$ holds the two lists G and P, while $\mathsf{P_A}$ holds the list T, together with lists G and P.

At the beginning of the evaluation phase, $\mathsf{P_B}$ generates labels for all wires. For the output wires of the circuit, $\mathsf{P_B}$ randomly generates wire labels representing 0

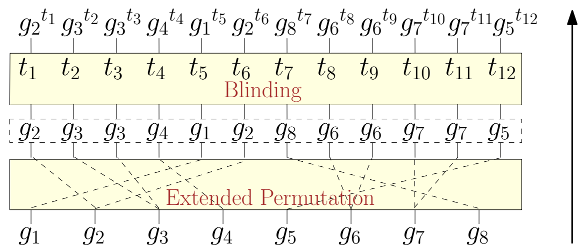


Fig. 2: Procedure of the circuit owner $\mathsf{P_A}$ in the initiation phase. The values in the dotted-line box are encrypted values that are hidden from $\mathsf{P_B}$.

and 1 from $\mathbb{G}$. For labels of other wires, $\mathsf{P_B}$ first picks randomly two values $\alpha_0 \in \mathbb{Z}_q$ and $\alpha_1 \in \mathbb{Z}_q$. Then, all incoming-wire and outgoing-wire labels, except the outgoing wires that are output wires of the circuit (whose have been generated), are generated via computing the values in the lists P and G to the power of α_0 and α_1, respectively, for values 0 and 1. Here, each element p_i in P is for an incoming wire IW_i, and the pair of its wire labels is computed via $(v_i^0, v_i^1) \leftarrow (p_i^{\alpha_0}, p_i^{\alpha_1})$, *i.e.*, $(v_i^0, v_i^1) = (g_{\pi_f(i)}^{t_i \alpha_0}, g_{\pi_f(i)}^{t_i \alpha_1})$. Similarly, for an outgoing wire OW_i, the pair of wire labels $(w_i^0, w_i^1) \leftarrow (g_i^{\alpha_0}, g_i^{\alpha_1})$ is computed using g_i in G. $\mathsf{P_B}$ now can garble all θ gates of the circuit that are composed solely of NAND gates for $\mathsf{P_A}$ one by one using these labels via a classical approach for garbling gates as we will introduce later. Then $\mathsf{P_B}$ sends these garbled gates to $\mathsf{P_A}$. Note that $\mathsf{P_B}$ is unaware of the EP π_f (and the topology of C_f). An illustration for wire labels with respect to garbled gates for the previous example (Fig. 1) is given in Fig. 3. Note that all input-wire labels of the circuit are generated and possessed by $\mathsf{P_B}$,

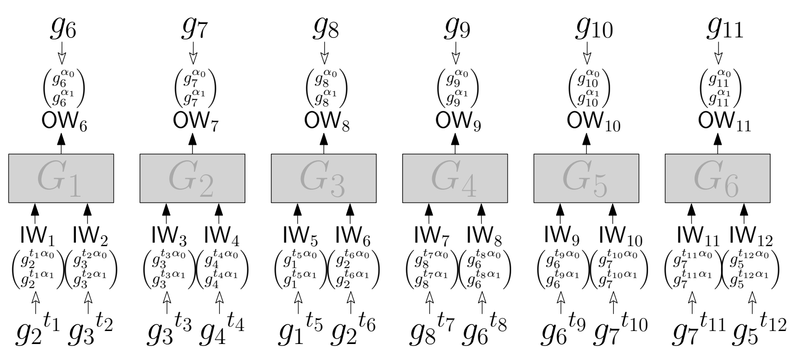


Fig. 3: Wire labels with respect to garbled gates for the circuit C_f.

and thus $\mathsf{P_B}$ picks out the input-wire labels corresponding to his input x_B and sends his garbled inputs to $\mathsf{P_A}$. Meanwhile, $\mathsf{P_A}$ could retrieve the garbled inputs corresponding to her input x_A from $\mathsf{P_B}$ through OT. This approach inherits from the standard approach of gabled circuits. Now since $\mathsf{P_A}$ knows the topology of the circuit, the list of blinding factors T, and input-wire labels, she can re-construct the garbled circuit assembled by the received garbled gates and evaluate the garbled circuit using both parties' input-wire labels $\{\mathsf{x}_i\}_{i\in[n]}$.

We now introduce the approach to garbling gates and evaluating the garbled circuit assembled by garbled gates. Two algorithms $(\mathsf{Gb}, \mathsf{Eval})$ are involved here.

The algorithm Gb is invoked by $\mathsf{P_B}$ to generate garbled gates (in a one-by-one manner) for $\mathsf{P_A}$. According to the circuit representation approach in Section 2.1, a gate G_i consists of two input wires, *i.e.*, incoming wires, with indices $2i-1$ and $2i$, and one output wire, *i.e.*, an outgoing wire, with index $n+i$. For such a gate, Gb takes as input the gate index i, the two pairs of input-wire labels (v^0_{2i-1}, v^1_{2i-1}) and (v^0_{2i}, v^1_{2i}), together with the pair of output-wire labels (w^0_{n+i}, w^1_{n+i}), and prepares four ciphertexts: $c_i^{a,b} \leftarrow \mathsf{Enc}^i_{v^a_{2i-1}, v^b_{2i}}(w^{\overline{a\cdot b}}_{n+i})$ for $a, b \in \{0,1\}$ for a dual-key cipher Enc. Gb outputs the set of garbled gates $\{\mathsf{GG}_i\}_{i\in[\theta]}$. Here $\mathsf{GG}_i = \{c_i^{a,b}\}_{a,b\in\{0,1\}}$, where $c_i^{a,b}$ are randomly permuted.

Eval is invoked by $\mathsf{P_A}$ to evaluate the garbled circuit that consists of garbled gates generated by $\mathsf{P_B}$. It takes as input a set of garbled gates $\{\mathsf{GG}_i\}_{i\in[\theta]}$, a set of input-wire labels $\{\mathsf{x}_i\}_{i\in[n]}$, the list of blinding factors $T = \{t_i\}_{i\in[N]}$, and an EP π_f. This algorithm first derives the description of the corresponding circuit C_f from π_f. Now starting from (outgoing-wire) labels $\{\mathsf{x}_i\}_{i\in[n]}$, Eval computes incoming-wire labels from the corresponding outgoing-wire labels and evaluates garbled gates one by one following the topographical order of the circuit to obtain the final output-wire labels. Without loss of generality, for an outgoing wire OW_i, we denote its label in hand by w^b_i, where $b \in \{0,1\}$. Note that each outgoing wire may connect with some incoming wires that are the input wires of some gates. Assume that an incoming wire IW_j is connected with OW_i, *i.e.*, $i = \pi_f(j)$. $\mathsf{P_A}$ can obtain the wire label of IW_j by computing the t_jth power of w^b_i, *i.e.*, $(w^b_i)^{t_j}$. It is easy to verify that $(w^b_i)^{t_j} = g_i^{\alpha_b t_j} = p_j^{\alpha_b} = v^b_j$, *i.e.*, the result is the input-wire (incoming-wire) label we want. When having two input-wire (incoming-wire) labels $(v^b_{2i-1}, v^{b'}_{2i})$, where $b, b' \in \{0,1\}$, for a garbled gate GG_i, the algorithm can decrypt GG_i using these two labels as keys (via a simple reverse approach of Enc) and obtain the non-$\perp$ resulting output-wire (outgoing-wire) label $w^{\overline{b\cdot b'}}_{n+i}$. It is easy to see that the values of the wire b and b' are hidden from $\mathsf{P_A}$ during this procedure. Since Eval follows the topology of the circuit, input-wire labels of a gate are always ready when we proceed to evaluate that gate. Finally, Eval returns the decrypted output-wire labels of the output gates.

The dual-key cipher Enc here can be constructed based on the random oracle (denoted by $\mathsf{H}\colon \mathbb{G}\times\mathbb{G}\times\{0,1\}^* \to \{0,1\}^{\|\mathbb{G}\|\times\tau}$) in a standard way: to garble a gate with index i, we let $\mathsf{Enc}^i_{v^a_{2i-1}, v^b_{2i}}(w^{\overline{a\cdot b}}_{n+i}) = \mathsf{H}(v^a_{2i-1}, v^b_{2i}, i) \oplus w^{\overline{a\cdot b}}_{n+i}$, [4] and further optimizations exist, *e.g.*, a variant of the point-and-permute optimization [6] (see [8]). This garbling scheme is secure under the random oracle model, and we refer readers to see more information in Appendix A.

We provide the formal descriptions of the protocol below.

[4] The operator $\oplus$ here is applied on the bit-representation of the right group element, and τ specifies the length of proper padding to ensure the check of correct decryption.

Protocol $\Pi_{\mathsf{activePFE}}$

Pre-agreement: Both parties agree on a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, where DDH assumption holds. They also have the pre-agreement that C_f consists of θ gates, m output wires, and $n(= n_A + n_B)$ input wires. We denote the number of incoming wires by $N \leftarrow 2\theta$ and the number of outgoing wires except those that are output wires of the circuit by $M \leftarrow n + \theta - m$.
Private inputs: $\mathsf{P_A}$ has a Boolean circuit input C_f and input $x_A \in \{0,1\}^{n_A}$, whereas the other party $\mathsf{P_B}$ has input $x_B \in \{0,1\}^{n_B}$.

Initiation Phase

In this phase, $\mathsf{P_A}$ has private circuit input C_f, while $\mathsf{P_B}$ has no input.

1. $\mathsf{P_B}$ picks $g_i \leftarrow\!\!\$\, \mathbb{G}$ for $i \in [M]$, such that all g_i's are different, and collects them as a list $G = [g_1, \ldots, g_M]$. Then, $\mathsf{P_B}$ sends G to $\mathsf{P_A}$.
2. $\mathsf{P_A}$ picks $s \leftarrow\!\!\$\, \mathbb{Z}_q$ and computes $h \leftarrow g^s$. The public key and private key of the ElGamal encryption then is denoted by $\mathsf{pk} = (\mathbb{G}, q, g, h)$ and $\mathsf{sk} = s$, respectively.
   $\mathsf{P_A}$ derives an EP π_f from C_f. Then $\mathsf{P_A}$ performs π_f on the elements of G and encrypts all resulting elements using pk to derive the list $\Phi = [c_1, c_2, \ldots, c_N]$, where c_i is the encryption of $g_{\pi_f(i)}$ for $i \in [N]$.
   $\mathsf{P_A}$ picks $t_i \leftarrow\!\!\$\, \mathbb{Z}_q$ for $i \in [N]$, such that all t_i's are different, and stores the list $T = [t_1, \ldots, t_N]$ for the evaluation phase. $\mathsf{P_B}$ computes the t_ith power of each plaintext $g_{\pi_f(i)}$ of c_i via the multiplicatively homomorphic property of the ElGamal encryption to obtain c'_i. Let the resulting list $\Phi' = [c'_1, \ldots, c'_N]$.
   $\mathsf{P_A}$ computes the information for decryption of all ciphertexts c'_i (remember that $c'_i = (c'^{(0)}_i, c'^{(1)}_i)$), *i.e.*, $\mathsf{P_A}$ computes $d_i \leftarrow (c'^{(0)}_i)^s$ for $i \in [N]$.
   $\mathsf{P_A}$ sends h, Φ, Φ', and $\{d_i\}_{i\in[N]}$ to $\mathsf{P_B}$. Then $\mathsf{P_A}$ uses the functionalities $\mathcal{F}^{\mathsf{EncEP}}_{\mathsf{zk}}$ to prove to $\mathsf{P_B}$ that she has performed a valid EP on G to obtain Φ. Meanwhile, $\mathsf{P_A}$ uses $\mathcal{F}^{\mathsf{DH}}_{\mathsf{zk}}$ to prove to $\mathsf{P_B}$ her knowledge of s, *i.e.*, the private key, for $(g, \{c'^{(0)}_i\}_{i\in[N]})$ and $(h, \{d_i\}_{i\in[N]})$, together with her knowledge of t_i for the two-tuple ciphertexts c_i and c'_i for all $i \in [N]$.
3. $\mathsf{P_B}$ decrypts all c'_i's to obtain the plaintexts $p_i \leftarrow c'^{(1)}_i \cdot d_i^{-1}$, and stores a list $P = [p_1, \ldots, p_N]$ for the evaluation phase.

Evaluation phase

In this phase, $\mathsf{P_A}$ has private input π_f (for C_f) and x_A, and $\mathsf{P_B}$ has private input x_B. $\mathsf{P_B}$ holds the two lists G and P derived in the initiation phase, while $\mathsf{P_A}$ holds the lists T, G, and P. This phase could be executed multiple times for different input x_A and x_B once the two parties finish the initiation phase.

1. For output wires of the circuit, $\mathsf{P_B}$ picks $w_i^0, w_i^1 \leftarrow\!\!\$\, \mathbb{G}$ for $i = M+1, \ldots, M+m$ as the wire labels. Then $\mathsf{P_B}$ picks $\alpha_0, \alpha_1 \leftarrow\!\!\$\, \mathbb{Z}_q$. For input wires of the circuit and output wires of non-output gates, *i.e.*, all outgoing wires except output wires of the circuit, $\mathsf{P_B}$ computes labels $w_i^0 \leftarrow g_i^{\alpha_0}$ and $w_i^1 \leftarrow g_i^{\alpha_1}$ for $i \in [M]$. For all incoming wires, $\mathsf{P_B}$ computes labels $v_i^0 \leftarrow p_i^{\alpha_0}$ and $v_i^1 \leftarrow p_i^{\alpha_1}$ for $i \in [N]$. $\mathsf{P_B}$ computes $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v^0_{2i-1}, v^1_{2i-1}), (v^0_{2i}, v^1_{2i}), (w^0_{n+i}, w^1_{n+i})\}_{i\in[\theta]})$. Here, for a gate with index i, (v^0_{2i-1}, v^1_{2i-1}) and (v^0_{2i}, v^1_{2i}) are the labels of the two input wires, and (w^0_{n+i}, w^1_{n+i}) are the labels of the output wire.

2. $\mathsf{P_A}$ and $\mathsf{P_B}$ execute $\mathcal{F}_{\mathsf{OT}}$. $\mathsf{P_B}$ uses as input $\{(w_i^0, w_i^1)\}_{i\in[n_A]}$, while $\mathsf{P_A}$ uses as input all bits of $x_A \in \{0,1\}^{n_A}$. At the end, $\mathsf{P_A}$ receives her garbled inputs $\{\mathsf{x}_i = w_i^{x_A[i]}\}_{i\in[n_A]}$.
3. $\mathsf{P_B}$ derives $\mathsf{x}_{n_A+i} \leftarrow w_{n_A+i}^{x_B[i]}$ for $i \in [n_B]$ as his garbled inputs. Then $\mathsf{P_B}$ sends $\mathsf{GC} = \{\mathsf{GG}_i\}_{i\in[\theta]}$ and $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$ to $\mathsf{P_A}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathsf{P_B}$ also sends the garbled output mapping $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[n]}$ to $\mathsf{P_A}$.
4. $\mathsf{P_A}$ computes the garbled output: $\{\mathsf{y}_i\}_{i\in[m]} \leftarrow \mathsf{Eval}(\mathsf{GC}, \{\mathsf{x}_i\}_{i\in[n_A+n_B]}, T, \pi_f)$. If $\mathsf{P_A}$ is allowed to know the evaluation result $y \in \{0,1\}^m$, $\mathsf{P_A}$ can derive and output y from the garbled output mapping he has received. If $\mathsf{P_B}$ is allowed to know the evaluation result, $\mathsf{P_A}$ sends $\{\mathsf{y}_i\}_{i\in[m]}$ to $\mathsf{P_B}$ so that $\mathsf{P_B}$ could derive and output the final result. If the output-wire labels are not consistent with those $\mathsf{P_B}$ generated, $\mathsf{P_B}$ outputs $\perp$.

We present the theorem for the security of the protocol $\Pi_{\mathsf{activePFE}}$ below.

Theorem 1. *If the dual-key cipher is constructed based on the random oracle as above and the DDH assumption of $\mathbb{G}$ holds, the protocol $\Pi_{\mathsf{activePFE}}$ securely realizes $\mathcal{F}_{\mathsf{activePFE}}$ in the presence of malicious $\mathsf{P_A}$ and semi-honest $\mathsf{P_B}$ in the $(\mathcal{F}_{\mathsf{OT}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}})$-hybrid world.*

The proof of this theorem can be found in Appendix B.

We note that there exist protocols that securely realize $\mathcal{F}_{\mathsf{OT}}$ (*e.g.*, [12, 24]), such that these protocols can be executed in parallel with constant-round and have linear complexity in the number of $\mathsf{P_B}$'s input wires $n_A(\leq n \ll \theta)$. Meanwhile, there exist protocols (*e.g.*, [14] that can be compiled by Fiat-Shamir heuristic to be non-interactive) that securely realizes $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$, such that the complexity of the total execution of the protocols is linear in $N(=2\theta)$, *i.e.*, linear in the number of gates θ. In Section 3.2, we will give a realization of $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ that can also be compiled to be non-interactive and has linear complexity. Therefore, the protocol $\Pi_{\mathsf{activePFE}}$ can be instantiated as a constant-round PFE protocol with linear complexity. By leveraging classical MPC results, such as the approach used in [23], our protocol can be compiled to be secure against malicious $\mathsf{P_B}$ and still preserves constant-round and linear complexity. Hence, we obtain a constant-round actively secure PFE protocol in the two-party setting with linear complexity.

Remark 1. The approach in [23] consider the case that $\mathsf{P_A}$ only provides a circuit C_f, while in some scenarios, $\mathsf{P_A}$ may also provide a private input x_A. For this case, we could simply use standard techniques, such as XOR-tree [28], to prevent malicious $\mathsf{P_B}$ launching selective-failure attacks.

In the following theorem, we show that executing the evaluation phase multiple times when the protocol involves the same circuit C_f (and EP π_f) does not sacrifice the security of the protocol $\Pi_{\mathsf{activePFE}}$. The proof of this theorem is put in Appendix C.

Theorem 2. *The evaluation phase of $\Pi_{\mathsf{activePFE}}$ can be securely executed multiple times for a fixed circuit C_f. In other words, the protocol that executes one initiation phase and multiple evaluation phases is secure against malicious $\mathsf{P_A}$ and semi-honest $\mathsf{P_B}$.*

We note that every execution of the evaluation phase in the view of $\mathsf{P_B}$ is to generate a set of new garbled gates, and the efforts to achieve reusability are mainly devoted to preventing $\mathsf{P_A}$ from learning additional information. Therefore, when we use classical MPC results, such as the approach used in [23], for the protocol, it is obvious that this reusability property still holds.

Remark 2. It is important that all messages from $\mathsf{P_B}$ in the initiation phase, including those from $\mathsf{P_B}$ in the protocols that securely realize $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ (in Section 3.2) are all random. Meanwhile, after the initiation phase, $\mathsf{P_B}$ does not possess any private information. Therefore, we can make the initiation phase non-interactive via borrowing the idea of Fiat-Shamir heuristic. Now $\mathsf{P_A}$ can use the random oracle to generate messages of $\mathsf{P_B}$ (using all previous messages), simulate the interaction, and publish her messages in this simulated interaction at one time. Via this approach, the protocol is *globally* reusable for the same circuit C_f. This means that *all* parties playing the role of $\mathsf{P_B}$ can retrieve $\mathsf{P_A}$'s messages and verify the correctness of these published messages. Then it is sufficient for them to directly start the evaluation phase with $\mathsf{P_A}$ for the fixed private circuit C_f multiple times using P and G derived in this simulated interaction. No interaction for initiation phase is needed between $\mathsf{P_A}$ and a potential party playing the role of $\mathsf{P_B}$. This is a new feature, since the reusability of previous PFE protocols with linear complexity [8] is *locally* reusable, such that $\mathsf{P_A}$ needs to interactively perform a setup for a fixed circuit with a specified $\mathsf{P_B}$, and the reusability only works for the two parties that perform such a setup together.

3.2 Realization of Functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$

In this section, we introduce an approach securely realizing the functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$. We would like to note that although EP is a generalization of permutation (shuffle), it seems that its corresponding zero-knowledge protocol cannot be constructed by simply modifying or invoking a shuffle protocol, *e.g.*, [5, 10]. That may be the main reason why [34] failed to provide such a specific protocol for EP by extending shuffle protocols (see Appendix B of [34] for their thoughts on failed attempts) and they only provided a protocol in a generic approach. In what follows, we provide an efficient and specific protocol for $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$.

We firstly introduce the basic idea of our protocol. The job of the prover in $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ is to convince the verifier that the plaintexts of a list of ciphertexts $\Phi = [c_1, \ldots, c_N]$ is derived from an EP that performs on a list of elements $G = [g_1, \ldots, g_M]$. In other words, the plaintext of each ciphertext in Φ is one of the elements in G. Notice that this is equivalent to saying that the plaintext of a ciphertext c_i is $\vec{g}^{\vec{e}_i} = \prod_{j=1}^{M} g_j^{e_{ij}}$, where the vector $\vec{e}_i = (e_{i1}, \ldots, e_{iM})$ is of the form that exact one entry is 1 and all other entries are 0, *i.e.*,

$$e_{ij} = \begin{cases} 1 & \text{if } c_i \text{ encrypts } g_j\,, \\ 0 & \text{otherwise.} \end{cases}$$

The vector $\vec{e}_i$ satisfies such a condition if and only if $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$ and $\vec{e}_i\vec{e}_i = \vec{e}_i$. The condition $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$ implies that the sum of all entries of $\vec{e}_i$ is equal to 1. The condition $\vec{e}_i\vec{e}_i = \vec{e}_i$ implies that $\vec{e}_i(\vec{e}_i - \vec{1}) = \vec{0}$, *i.e.*, each entry of the vector is either 0 or 1. These two conditions conclude that $\vec{e}_i$ is of the form that exact one entry is 1 and all other entries are 0. In addition, the corresponding ciphertext c_i is of the form $(g^{r_i}, \vec{g}^{\vec{e}_i} h^{r_i})$, which is reminiscent of ElGamal or Pedersen [37] commitment schemes and can be regarded as a commitment to the vector $\vec{e}_i$. Therefore, the prover's goal is to prove that each "committed" vector $\vec{e}_i$ satisfies $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$ and $\vec{e}_i\vec{e}_i = \vec{e}_i$, in a zero-knowledge manner. We note that it is possible for the prover to simultaneously prove the conditions for all $\vec{e}_i$'s.

For the proof of the condition $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$, let the verifier pick a challenge $\omega \leftarrow_{\$} \mathbb{Z}_q$. Then using the homomorphic property, both parties compute $C = (\prod_{i=1}^{N}(c_i^{(0)})^{\omega^i}, \prod_{i=1}^{N}(c_i^{(1)})^{\omega^i})$, which can be regarded as a commitment to the vector $\vec{e} = \sum_{i=1}^{N}\omega^i\vec{e}_i$. Since ω is random, if $\sum_{i=1}^{N}\omega^i(\vec{1}^{\mathrm{T}}\vec{e}_i) = \sum_{i=1}^{N}\omega^i$ holds, then $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$ holds for all $i \in [M]$ with an overwhelming probability. Let $\Omega \leftarrow \sum_{i=1}^{N}\omega^i$. Since $\sum_{i=1}^{N}\omega^i(\vec{1}^{\mathrm{T}}\vec{e}_i) = \vec{1}^{\mathrm{T}}\vec{e}$ and $\vec{e}$ is committed in C, it is enough for the prover to prove that $\vec{1}^{\mathrm{T}}\vec{e} = \Omega$ holds if the prover is computationally bounded.

We could follow a similar approach for the proof of the condition $\vec{e}_i\vec{e}_i = \vec{e}_i$. Let the verifier pick a random challenge $x \in \mathbb{Z}_q$. Then, using the homomorphic property, both parties compute $c_{\vec{d}} = (\prod_{i=1}^{N}(c_i^{(0)})^{x^i}, \prod_{i=1}^{N}(c_i^{(1)})^{x^i})$, which can be regarded as a commitment to $\vec{d} = \sum_{i=1}^{N}x^i\vec{e}_i$. Since x is randomly chosen, if $\sum_{i=1}^{N}x^i\vec{e}_i\vec{e}_i - \vec{d} = \vec{0}$ holds, then $\vec{e}_i\vec{e}_i = \vec{e}_i$ holds for all $i \in [N]$ with an overwhelming probability. Moreover, let the verifier pick another random challenge $y \in \mathbb{Z}_q$ and define a bilinear map $* : \mathbb{Z}_q^M \times \mathbb{Z}_q^M \to \mathbb{Z}_q$ by $(a_1, \ldots, a_M) * (b_1, \ldots, b_M) = \sum_{j=1}^{M} a_j b_j y^j$. Similarly, if $\vec{e}_i * \vec{e}_i - \vec{1} * \vec{e}_i = 0$, then $\vec{e}_i\vec{e}_i = \vec{e}_i$ holds with an overwhelming probability. Hence, since the vectors $\vec{e}_i$'s and $\vec{d}$ have been committed in c_i's and $c_{\vec{d}}$, it is enough for the prover to prove that $\sum_{i=1}^{N}x^i\vec{e}_i * \vec{e}_i - \vec{1} * \vec{d} = 0$ holds if the prover is computationally bounded.

It is important to note that all g_i's are generated by $\mathsf{P_B}$, and thus a computationally bounded $\mathsf{P_A}$ cannot find a non-trivial discrete logarithm relation for $\{g_i\}_{i\in[M]}$ except a negligible probability. This guarantees the soundness of the protocols. Now we present the protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ between a prover P and a verifier V below. Two sub-protocols $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ then follow respectively. In these protocols, V always verifies whether the received messages are of correct form, and rejects once they are not. These protocols are all public-coin honest-verifier zero-knowledge, and we can compile them to be non-interactive and secure via Fiat-Shamir heuristic to obtain the protocols we want.

Protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$

Public Inputs: A cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, where DDH assumption holds. The public key of the ElGamal encryption scheme $\mathsf{pk} = (\mathbb{G}, q, g, h)$. A list of elements $G = [g_1, \ldots, g_M]$. A list of ElGamal ciphertexts $\Phi = [c_1, \ldots, c_N]$, where $c_i = (c_i^{(0)}, c_i^{(1)})$. Elements in G and Φ all belong to the group $\mathbb{G}$.

Witness: P has an EP π and a list $R = [r_1, \dots, r_N]$ that are random coins of ciphertexts in Φ, where $r_i \in \mathbb{Z}_q$.

1. For $i \in [N]$, P derives a vector $\vec{e}_i = (e_{i,1}, \dots, e_{i,M}) \in \mathbb{Z}_q^M$ from π, such that the encrypted value of c_i can be represented by $\vec{g}^{\vec{e}_i}$. For the EP π, this vector is of the form where exact one entry is 1 and all other entries are all 0.
2. V picks an element $\omega \leftarrow_{\$} \mathbb{Z}_q$ and sends it to P. Both parties compute $C \leftarrow (C^{(0)} = \prod_{i=1}^N (c_i^{(0)})^{\omega^i}, C^{(1)} = \prod_{i=1}^N (c_i^{(1)})^{\omega^i})$. P computes $\vec{e} \leftarrow \sum_{i=1}^N \omega^i \vec{e}_i$ and $r_{\vec{e}} \leftarrow \sum_{i=1}^N \omega^i r_i$. Both parties compute $\Omega \leftarrow \sum_{i=1}^N \omega^i$. P proves to V the following relation R_{Sum} for $\vec{y} = \vec{1}$ via the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$:
$$\{(\mathbb{G}, q, g, h, G, C, \Omega, \vec{y}) \mid \exists(\vec{e}, r_{\vec{e}}) : C^{(0)} = g^{r_{\vec{e}}} \land C^{(1)} = \vec{g}^{\vec{e}} h^{r_{\vec{e}}} \land \vec{y}^{\mathrm{T}} \vec{e} = \Omega\}.$$
3. V picks two elements $x, y \leftarrow_{\$} \mathbb{Z}_q$ and sends them to P. Both parties compute $c_{\vec{d}_i} \leftarrow (c_{\vec{d}_i}^{(0)} = (c_i^{(0)})^{x^i}, c_{\vec{d}_i}^{(1)} = (c_i^{(1)})^{x^i})$ for $i \in [N]$ and also $c_{\vec{d}} \leftarrow (c_{\vec{d}}^{(0)} = \prod_{i=1}^N (c_i^{(0)})^{x^i}, c_{\vec{d}}^{(1)} = \prod_{i=1}^N (c_i^{(1)})^{x^i})$ and $c_{-\vec{1}} \leftarrow (\prod_{i=1}^M g_i^{-1}, 1)$. P computes $\vec{d}_i \leftarrow x^i \vec{e}_i$ and $r_{\vec{d}_i} \leftarrow x^i r_i$ for $i \in [N]$, $\vec{d} \leftarrow \sum_{i=1}^N \vec{d}_i$, and $r_{\vec{d}} = \sum_{i=1}^N r_{\vec{d}_i}$. Define a bilinear map $* : \mathbb{Z}_q^M \times \mathbb{Z}_q^M \to \mathbb{Z}_q$ by $(a_1, \dots, a_M) * (b_1, \dots, b_M) = \sum_{j=1}^M a_j b_j y^j$. P proves to V the following relation R_{Zero} via the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$:
$$\begin{aligned}&\{(\mathbb{G}, q, g, h, G, \Phi, \{c_{\vec{d}_i}\}_{i \in [N]}, c_{\vec{d}}, c_{-\vec{1}}) \mid \exists(\{\vec{e}_i, r_i, \vec{d}_i, r_{d_i}\}_{i \in [N]}, \vec{d}, r_{\vec{d}}) : \\ &(\forall i \in [N], c_i^{(0)} = g^{r_i} \land c_i^{(1)} = \vec{g}^{\vec{e}_i} h^{r_i} \land c_{\vec{d}_i}^{(0)} = g^{r_{\vec{d}_i}} \land c_{\vec{d}_i}^{(1)} = \vec{g}^{\vec{d}_i} h^{r_{\vec{d}_i}}) \\ &\qquad \land c_{\vec{d}}^{(0)} = g^{r_{\vec{d}}} \land c_{\vec{d}}^{(1)} = \vec{g}^{\vec{d}} h^{r_{\vec{d}}} \land \sum_{i=1}^N \vec{e}_i * \vec{d}_i - \vec{1} * \vec{d} = 0\}.\end{aligned}$$

Theorem 3. *The protocol $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ is an honest-verifier zero-knowledge argument of knowledge for R_{EncEP}.*

The proof of this theorem can be found in Appendix D.

The protocol $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ between a prover P and a verifier V below uses the idea mentioned in [11] for recursing the protocol and halving the statement in each recursion. Thus, $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ has logarithmic communication cost. Throughout this protocol, we assume that the parameter M is a power of 2. If needed, one can easily pad the inputs to ensure that this holds as in [11].

Protocol $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$

Public Inputs: A cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, where DDH assumption holds. The public key of the ElGamal encryption scheme $\mathsf{pk} = (\mathbb{G}, q, g, h)$. An ElGamal ciphertexts $C = (C^{(0)}, C^{(1)})$. An element $\Omega \in \mathbb{Z}_q$. Two vectors $\vec{g} = (g_1, \dots, g_M)$ and $\vec{y} = (y_1, \dots, y_M)$ of length M. Denote the length of vectors $\vec{g}$ and $\vec{y}$ by $\ell = M$. Let $c_{\vec{e}} \leftarrow C^{(1)}$. Both parties initiate an element $c'_{\vec{e}} \leftarrow g^{\Omega}$.
Witness: The prover P has witness $\vec{e}$, $r_{\vec{e}}$.
Statement: There exist $\vec{e}$ and $r_{\vec{e}}$, such that $C^{(0)} = g^{r_{\vec{e}}} \land c_{\vec{e}} = \vec{g}^{\vec{e}} h^{r_{\vec{e}}} \land c'_{\vec{e}} = g^{\vec{y}^{\mathrm{T}} \vec{e}}$.

– V picks $u \leftarrow_{\$} \mathbb{G}$ and sends u to P. P initiates $\rho_{\vec{e}} = 0$, and $\rho'_{\vec{e}} = 0$. Then two parties engage in the procedure below to prove the statement:

There exist $\vec{e}$, $r_{\vec{e}}$, $\rho_{\vec{e}}$, and $\rho'_{\vec{e}}$, such that $C^{(0)} = g^{r_{\vec{e}}} \wedge c_{\vec{e}} = \vec{g}^{\vec{e}} u^{\rho_{\vec{e}}} h^{r_{\vec{e}}} \wedge c'_{\vec{e}} = g^{\vec{y}^{\mathrm{T}}\vec{e}} u^{\rho'_{\vec{e}}}$.

- If $\ell = 1$, we denote the only element in $\vec{e}$, $\vec{g}$, and $\vec{y}$ by $\bar{e}$, $\bar{g}$, and $\bar{y}$, respectively. Let $\gamma \leftarrow g^{\bar{y}}$. Now $c_{\vec{e}}$, $c'_{\vec{e}}$, and $C^{(0)}$ are of the form $c_{\vec{e}} = \bar{g}^{\bar{e}} u^{\rho_{\vec{e}}} h^{r_{\vec{e}}}$, $c'_{\vec{e}} = \gamma^{\bar{e}} u^{\rho'_{\vec{e}}}$, and $C^{(0)} = g^{r_{\vec{e}}}$, respectively. P and V follow the procedure as follows.
  1. P picks $x_1, x_2, x_3, x_4 \leftarrow\$\, \mathbb{Z}_q$, and sends $a_1 \leftarrow \bar{g}^{x_1} u^{x_2} h^{x_3}$, $a_2 \leftarrow \gamma^{x_1} u^{x_4}$, $a_3 \leftarrow g^{x_3}$ to V.
  2. V sends $\alpha \leftarrow\$\, \mathbb{Z}_q$ to P.
  3. P sends $z_1 \leftarrow x_1 + \alpha\bar{e}$, $z_2 \leftarrow x_2 + \alpha\rho_{\vec{e}}$, $z_3 \leftarrow x_3 + \alpha r_{\vec{e}}$, and $z_4 \leftarrow x_4 + \alpha\rho'_{\vec{e}}$ to V.
  4. V verifies whether equations $\bar{g}^{z_1} u^{z_2} h^{z_3} = a_1 c_{\vec{e}}^{\alpha}$, $\gamma^{z_1} u^{z_4} = a_2 (c'_{\vec{e}})^{\alpha}$, and $g^{z_3} = a_3 (C^{(0)})^{\alpha}$ hold. If they all hold, V outputs accept. Otherwise, V outputs reject.
- If $\ell \neq 1$, P and V follow the following procedure.
  1. We write $\vec{e} = \vec{e}_L \circ \vec{e}_R$, $\vec{g} = \vec{g}_L \circ \vec{g}_R$, and $\vec{y} = \vec{y}_L \circ \vec{y}_R$. P computes $v_L \leftarrow \vec{g}_R^{\vec{e}_L} u^{\rho_L}$, $v_R \leftarrow \vec{g}_L^{\vec{e}_R} u^{\rho_R}$, $v'_L \leftarrow g^{\vec{y}_R^{\mathrm{T}}\vec{e}_L} u^{\rho'_L}$, and $v'_R \leftarrow g^{\vec{y}_L^{\mathrm{T}}\vec{e}_R} u^{\rho'_R}$, where $\rho_L, \rho_R, \rho'_L, \rho'_R \leftarrow\$\, \mathbb{Z}_q$. Then P sends v_L, v_R, v'_L, and v'_R to V.
  2. V sends $\alpha \leftarrow\$\, \mathbb{Z}_q$ to P.
  3. P computes $\vec{e}' = \alpha\vec{e}_L + \alpha^{-1}\vec{e}_R$ of length $\ell' = \ell/2$, and also computes $\rho_{\vec{e}'} \leftarrow \rho_{\vec{e}} + \alpha^2\rho_L + \alpha^{-2}\rho_R$ and $\rho'_{\vec{e}'} \leftarrow \rho'_{\vec{e}} + \alpha^2\rho'_L + \alpha^{-2}\rho'_R$. Both parties compute $c_{\vec{e}'} \leftarrow c_{\vec{e}} v_L^{\alpha^2} v_R^{\alpha^{-2}}$ and $c'_{\vec{e}'} \leftarrow c'_{\vec{e}} (v'_L)^{\alpha^2} (v'_R)^{\alpha^{-2}}$, and two vectors $\vec{g}' \leftarrow \vec{g}_L^{\alpha^{-1}} \vec{g}_R^{\alpha}$ and $\vec{y}' \leftarrow \alpha^{-1}\vec{y}_L + \alpha\vec{y}_R$ of length $\ell' = \ell/2$. It is easy to verify that $c_{\vec{e}'} = \vec{g'}^{\vec{e}'} u^{\rho_{\vec{e}'}} h^{r_{\vec{e}}}$ and $c'_{\vec{e}'} = g^{\vec{y'}^{\mathrm{T}}\vec{e}'} u^{\rho'_{\vec{e}'}}$.
  4. Both parties recurse on $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Sum}}$ for the same $C^{(0)}$, $(\mathbb{G}, q, g, h)$, u but using $c_{\vec{e}'}$, $c'_{\vec{e}'}$, $\vec{g}'$, $\vec{y}'$ in place of $c_{\vec{e}}$, $c'_{\vec{e}}$, $\vec{g}$, $\vec{y}$. P in the recursion uses the same $r_{\vec{e}}$, but uses $\rho_{\vec{e}'}$, $\rho'_{\vec{e}'}$, $\vec{e}'$ in place of $\rho_{\vec{e}}$, $\rho'_{\vec{e}}$, $\vec{e}$. We use $\ell' = \ell/2$ in place of ℓ to denote the length of vector $\vec{g}'$, $\vec{y}'$, and $\vec{e}'$.

Theorem 4. *The protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Sum}}$ is an honest-verifier zero-knowledge argument of knowledge for the relation R_{Sum}.*

The proof of this theorem can be found in Appendix E.

The protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero}}$ between a prover P and a verifier V below borrows the idea of the *zero argument* in [5]. We tailor the protocol to support the ElGamal encryption scheme and introduce how to further reduce the communication cost in Remark 3.

Protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero}}$

Public Inputs: A cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, where DDH assumption holds. The public key of the ElGamal encryption scheme $\mathsf{pk} = (\mathbb{G}, q, g, h)$. A list $G = [g_1, \ldots, g_M]$. Two lists of ElGamal ciphertexts $\{c_{\vec{u}_i}^{(0)}, c_{\vec{u}_i}^{(1)}\}_{i\in[\ell]}, \{c_{\vec{v}_i}^{(0)}, c_{\vec{v}_i}^{(1)}\}_{i\in[\ell]}$. The description of the bilinear map $*$ for a variable y.
Witness: The prover P has witness $\{\vec{u}_i, r_{\vec{u}_i}\}_{i\in[\ell]}$, $\{\vec{v}_i, r_{\vec{v}_i}\}_{i\in[\ell]}$.
Statement: There exist $\{\vec{u}_i, r_{\vec{u}_i}\}_{i\in[\ell]}$ and $\{\vec{v}_i, r_{\vec{v}_i}\}_{i\in[\ell]}$, such that $c_{\vec{u}_i}^{(0)} = g^{r_{\vec{u}_i}}$, $c_{\vec{u}_i}^{(1)} = \vec{g}^{\vec{u}_i} h^{r_{\vec{u}_i}}$, $c_{\vec{v}_i}^{(0)} = g^{r_{\vec{v}_i}}$, $c_{\vec{v}_i}^{(1)} = \vec{g}^{\vec{v}_i} h^{r_{\vec{v}_i}}$ for all $i \in [\ell]$, and $\sum_{i=1}^{\ell} \vec{u}_i * \vec{v}_i = 0$.

1. P picks $\vec{u}_0, \vec{v}_{\ell+1} \leftarrow_\$ \mathbb{Z}_q^M$ and $r_{\vec{u}_0}, r_{\vec{v}_{\ell+1}} \leftarrow_\$ \mathbb{Z}_q$. Then P computes $c_{\vec{u}_0} \leftarrow (c_{\vec{u}_0}^{(0)} = g^{r_{\vec{u}_0}}, c_{\vec{u}_0}^{(1)} = \vec{g}^{\vec{u}_0} h^{r_{\vec{u}_0}})$ and $c_{v_{\ell+1}} \leftarrow (c_{v_{\ell+1}}^{(0)} = g^{r_{\vec{v}_{\ell+1}}}, c_{v_{\ell+1}}^{(1)} = \vec{g}^{\vec{v}_{\ell+1}} h^{r_{\vec{v}_{\ell+1}}})$. P computes for $\phi = 0, \ldots, 2\ell$

$$d_\phi \leftarrow \sum_{\substack{0 \le i \le \ell, 1 \le j \le \ell+1 \\ j = \ell+1-\phi+i}} \vec{u}_i * \vec{v}_j \, .$$

   P picks $r_{d_\phi} \leftarrow_\$ \mathbb{Z}_q$ for $\phi \in \{0, \ldots, 2\ell\} \backslash \{\ell+1\}$ and computes $c_{d_\phi} \leftarrow g^{d_\phi} h^{r_{d_\phi}}$ for $\phi \in \{0, \ldots, 2\ell\} \backslash \{\ell+1\}$. For $\phi = \ell+1$, both parties set $r_{d_{\ell+1}} \leftarrow 0$ and $c_{d_{\ell+1}} \leftarrow 1$, After the computation, P sends $c_{\vec{u}_0}$, $c_{\vec{v}_{\ell+1}}$, and $\{c_{d_\phi}\}_{\phi \in \{0, \ldots, 2\ell\} \backslash \{\ell+1\}}$ to V.
2. V sends $x \leftarrow_\$ \mathbb{Z}_q$ to P.
3. P computes $\vec{u} \leftarrow \sum_{i=0}^{\ell} x^i \vec{u}_i$, $r_{\vec{u}} \leftarrow \sum_{i=0}^{\ell} x^i r_{\vec{u}_i}$, $\vec{v} \leftarrow \sum_{j=1}^{\ell+1} x^{\ell-j+1} \vec{v}_j$, $r_{\vec{v}} \leftarrow \sum_{j=1}^{\ell+1} x^{\ell+1-j} r_{\vec{v}_j}$, and $t \leftarrow \sum_{\phi=0}^{2\ell} x^\phi r_{d_\phi}$, and sends $\vec{u}$, $r_{\vec{u}}$, $\vec{v}$, $r_{\vec{v}}$, t to V.
4. V outputs accept if all equations $\prod_{i=0}^{\ell} (c_{\vec{u}_i}^{(0)})^{x^i} = g^{r_{\vec{u}}}$, $\prod_{i=0}^{\ell} (c_{\vec{u}_i}^{(1)})^{x^i} = \vec{g}^{\vec{u}} h^{r_{\vec{u}}}$, $\prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(0)})^{x^{\ell+1-j}} = g^{r_{\vec{v}}}$, $\prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(1)})^{x^{\ell+1-j}} = \vec{g}^{\vec{v}} h^{r_{\vec{v}}}$, and $\prod_{\phi=0}^{2\ell} c_{d_\phi}^{x^\phi} = g^{\vec{u} * \vec{v}} h^t$ hold. Otherwise, V outputs reject.

Theorem 5. *The protocol $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ is an honest-verifier zero-knowledge argument of knowledge for the relation R_{Zero}.*

The proof of this theorem can be found in Appendix F.

Remark 3. We can further reduce the communication cost of $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$. Notice that in Step 1, P needs to commit to all elements in $\{d_\phi\}_{\phi=0,\ldots,2\ell}$. We could include a list of $2\ell+1$ random elements of $\mathbb{G}$, *e.g.*, $H = \{h_\phi\}_{\phi=0,\ldots,2\ell}$, in the common reference string. P can thus commit to $\{d_\phi\}_{\phi=0,\ldots,2\ell}$ by computing $c_{\vec{d}} \leftarrow (g^{r_{\vec{d}}}, \sum_{\phi=0}^{2\ell} h_\phi^{d_\phi} h^{r_{\vec{d}}})$ for $r_{\vec{d}} \leftarrow_\$ \mathbb{Z}_q$. P now only needs to send $c_{\vec{d}}$ to verifier instead of $\{c_{d_\phi}\}_{\phi \in \{2,\ldots,2\ell\} \backslash \{\ell+1\}}$, and does not need to send t to V in Step 3. Alternatively, P proves to V the following statement for $\vec{y} = [x^0, \ldots, x^\ell, 0, x^{\ell+2}, x^{2\ell}]$ and $D = \vec{u} * \vec{v}$ via the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ in Step 4:

$$\{(\mathbb{G}, q, g, h, H, c_{\vec{d}}, D, \vec{y}) \mid \exists (\vec{d}, r_{\vec{d}}) : c_{\vec{d}}^{(0)} = g^{r_{\vec{d}}} \wedge c_{\vec{d}}^{(1)} = \vec{h}^{\vec{d}} h^{r_{\vec{d}}} \wedge \vec{y}^{\mathrm{T}} \vec{d} = D\} \, .$$

Following this approach, we reduce the linear communication cost of sending d_ϕ's to the logarithmic communication cost of using $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$. Similarly, P can avoid directly sending $\vec{v}$ and $r_{\vec{v}}$, *i.e.*, the opening for $c_{\vec{v}} = (c_{\vec{v}}^{(0)}, c_{\vec{v}}^{(1)}) = (\prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(0)})^{x^{\ell+1-j}}, \prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(1)})^{x^{\ell+1-j}})$. Now P only sends $\vec{u}$ and $r_{\vec{u}}$ in Step 3, and V only verifies the two equations related to $\vec{u}$ and $r_{\vec{u}}$ in Step 4. Then, P sends $D = \vec{u} * \vec{v}$ to V and proves the following statement for $\vec{y} = [y^1 u_1, \ldots, y^M u_M]$ via the protocol $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ in Step 4:

$$\{(\mathbb{G}, q, g, h, G, c_{\vec{v}}, D, \vec{y}) \mid \exists (\vec{v}, r_{\vec{v}}) : c_{\vec{v}}^{(0)} = g^{r_{\vec{v}}} \wedge c_{\vec{v}}^{(1)} = \vec{g}^{\vec{v}} h^{r_{\vec{v}}} \wedge \vec{y}^{\mathrm{T}} \vec{v} = D\} \, .$$

4 PFE Protocol for PVC Security

In this section, we introduce the *first* constant-round PVC-secure PFE protocol with linear complexity in the two-party setting based on the results in Section 3. The corresponding ideal functionality $\mathcal{F}_{\mathsf{covertPFE}}$ is given in the following.

Functionality $\mathcal{F}_{\mathsf{covertPFE}}$

Pre-agreement: The circuit C_f consists of θ gates, m output wires, and $n(= n_A + n_B)$ input wires.
Private inputs: $\mathsf{P_A}$ has a Boolean circuit input C_f and input $x_A \in \{0,1\}^{n_A}$, whereas the other party $\mathsf{P_B}$ has input $x_B \in \{0,1\}^{n_B}$.

1. If an input of the form abort_i from the party P_i for some $i = \{A, B\}$ is received, the ideal functionality sends $\perp$ to both parties and the ideal execution terminates.
2. If a circuit C_f satisfying the pre-agreement from $\mathsf{P_A}$ is received, store C_f.
3. If an input of the form $\mathsf{blatantCheat}$ from $\mathsf{P_B}$ is received, the ideal functionality sends $\mathsf{corrupted}$ to both parties and terminates.
4. If an input of the form cheat from $\mathsf{P_B}$ is received and $\mathsf{P_A}$'s inputs C_f and x_A were received previously:
   - With probability ϵ, the ideal functionality sends $\mathsf{corrupted}$ to both parties and terminates.
   - With probability $1 - \epsilon$, the ideal functionality sends $(\mathsf{undetected}, x_A, C_f)$ to $\mathsf{P_B}$. If $\mathsf{P_A}$ is allowed to receive the output, the ideal functionality waits for $y \in \{0,1\}^m$ from the adversary $\mathcal{A}$, sends y to $\mathsf{P_A}$, and terminates.
5. If input $x_A \in \{0,1\}^{n_A}$ from $\mathsf{P_A}$ and $x_B \in \{0,1\}^{n_B}$ from $\mathsf{P_B}$ are received and an input circuit C_f is stored, the ideal functionality computes $C_f(x_A, x_B)$.
   (a) If $\mathsf{P_A}$ (when she is corrupted by $\mathcal{A}$) is allowed to learn $C_f(x_A, x_B)$, then it sends $C_f(x_A, x_B)$ to $\mathsf{P_A}$.
   (b) Otherwise, the ideal functionality sends $\mathsf{nothing}$ to $\mathsf{P_A}$. Then if $\mathsf{continue}$ from $\mathcal{A}$ is received, the ideal functionality sends $C_f(x_1, x_2)$ to the honest $\mathsf{P_B}$. Otherwise, if abort_A is received from $\mathcal{A}$ on behalf of the corrupted $\mathsf{P_A}$, it sends $\perp$ to the honest $\mathsf{P_B}$.

We give the PVC-security definition for our PFE protocol $\Pi_{\mathsf{covertPFE}}$ as follows.

Definition 4. *A two-party PFE protocol $\Pi_{\mathsf{covertPFE}}$ along with algorithms* Blame *and* Judge *is publicly verifiable covert secure with ϵ-deterrent if the following conditions hold.*

PVC security *The protocol $\Pi_{\mathsf{covertPFE}}$, which might output* cert *if the honest party detects covert cheating, securely realizes $\mathcal{F}_{\mathsf{covertPFE}}$ with ϵ-deterrent.*

Public verifiability *If the honest party outputs* cert *during the protocol execution, then the output of the algorithm* Judge *for* cert *is* 1*, except a negligible probability.*

Defamation freeness *If one party is honest, the probability that the other malicious party generates a certificate* cert *for which* Judge *outputs* 1 *is negligible.*

4.1 Full Description of the Protocol

In the two-party case, active security implies covert security with public verifiability, since we could regard attempts to cheat as abortions. Therefore, techniques for dealing with malicious $\mathsf{P_A}$ are workable for the PVC-secure setting.

Here we briefly introduce the main idea of our PVC-secure protocol $\Pi_{\mathsf{covertPFE}}$. Recall that in Remark 2, we describe how to make the initiation phase non-interactive. This approach can also be adopted here in $\Pi_{\mathsf{covertPFE}}$. Thus, we now do not need to consider malicious $\mathsf{P_B}$ in the initiation phase. We can reuse the initiation phase of $\Pi_{\mathsf{activePFE}}$ for $\Pi_{\mathsf{covertPFE}}$, with the exception that we include G in the common reference string to simplify the proof of security. Note that this small change does not hinder the protocol from achieving global reusability.

In the evaluation phase of $\Pi_{\mathsf{activePFE}}$, $\mathsf{P_A}$ receives the garbled circuit and garbled inputs, evaluates the garbled circuit, and derives the resulting outputs or sends garbled outputs back to $\mathsf{P_B}$. It is easy to see that $\mathsf{P_A}$ has no chance to cheat in the protocol. Even if $\mathsf{P_A}$ sends incorrect garbled outputs to $\mathsf{P_B}$, the incorrect garbled outputs will still be rejected by $\mathsf{P_B}$ due to the authenticity of the garbling. Hence, we only need to focus on the security against covert $\mathsf{P_B}$.

To achieve covert security, we follow the same paradigm of all existing work, *i.e.*, parties generate λ instances of a passively secure protocol, check the correctness of $\lambda-1$ randomly chosen instances, and take the result of the unopened one. In addition, we use a derandomized approach to supporting efficient correctness check in our protocol. More concretely, $\mathsf{P_B}$ needs to pick for each instance a seed to generate random coins during the execution of that instance (including the circuit garbling and OT protocol). $\mathsf{P_A}$ then uses OT protocol to retrieve all but one of the seeds, such that $\mathsf{P_B}$ is unaware of which instances are checked. Now given the seeds, $\mathsf{P_A}$ can easily check the correctness of the corresponding instances. To prevent $\mathsf{P_B}$ leaking inputs, $\mathsf{P_B}$ commits to his pairs of input-wire labels in random order with randomness derived from the seed and send these two commitments to $\mathsf{P_A}$ for each instance. Hence, $\mathsf{P_A}$ can effectively check the correctness of these commitments using the seed for opened instance, while $\mathsf{P_B}$'s inputs are preserved. After the check, $\mathsf{P_A}$ points out the unopened instance, and now one of the two commitments for her input wires needs to be opened by $\mathsf{P_B}$ as his garbled input to enable $\mathsf{P_A}$ to evaluate the unopened garbled circuit.

To add public verifiability to the approach above, we let $\mathsf{P_B}$ sign all transcripts that have been produced before the time when $\mathsf{P_A}$ reveals the index of the unopened instance. In addition, for each instance, let $\mathsf{P_A}$ commit to a random seed at the beginning of the protocol and uses this seed to derived random coins during her execution of the instance. This commitment will be included in $\mathsf{P_B}$'s transcript and signed by $\mathsf{P_B}$, such that it can prevent $\mathsf{P_A}$ from defaming honest $\mathsf{P_B}$. If $\mathsf{P_B}$ deviates from an instance checked by $\mathsf{P_A}$, $\mathsf{P_A}$ can generate a certificate that includes related transcripts and $\mathsf{P_B}$'s signature on them for that instance, such that it allows a third party to verify this proof of misbehavior. Since $\mathsf{P_B}$ cannot realize in time that the instance in which he deviates from the protocol has been checked by $\mathsf{P_A}$, he cannot abort before $\mathsf{P_A}$ has collected enough materials to generate the certificate.

Our protocol $\Pi_{\mathsf{covertPFE}}$ is given in the following. Since parties need to commit to transcripts of the OT executions in the protocol, the description directly uses the protocol Π_{OT} that securely realizes a parallel version of $\mathcal{F}_{\mathsf{OT}}$.

Protocol $\Pi_{\mathsf{covertPFE}}$

Pre-agreement: Both parties agree on a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, where DDH assumption holds. They also have the pre-agreement about C_f: θ gates, m output wires, $n(= n_A + n_B)$ input wires, $N = 2\theta$ incoming wires, and $M = n + \theta - m$ outgoing wires except output wires of the circuit. The common reference string includes a list $G = [g_1, \ldots, g_M] \in \mathbb{G}^M$, where all g_i's are different.
Private inputs: $\mathsf{P_A}$ has a Boolean circuit input C_f and input $x_A \in \{0,1\}^{n_A}$, whereas the other party $\mathsf{P_B}$ has input $x_B \in \{0,1\}^{n_B}$ and keys $(\mathsf{vk}, \mathsf{sigk})$ for a signature scheme. $\mathsf{P_A}$ knows the verification key vk.

Initiation Phase

1. $\mathsf{P_A}$ picks $s \leftarrow\$ \mathbb{Z}_q$ and computes $h \leftarrow g^s$. Denote the public and private keys of the ElGamal encryption by $\mathsf{pk} = (\mathbb{G}, q, g, h)$ and $\mathsf{sk} = s$, respectively.
   $\mathsf{P_A}$ derives an EP π_f from C_f. Then $\mathsf{P_A}$ permutes elements of G according to π_f and encrypts all resulting elements using pk to derive the list $\Phi = [c_1, c_2, \ldots, c_N]$, where c_i is the encryption of $g_{\pi_f(i)}$ for $i \in [N]$.
   $\mathsf{P_A}$ picks $t_i \leftarrow\$ \mathbb{Z}_q$ for $i \in [N]$, such that all t_i's are different, and stores the list $T = [t_1, \ldots, t_N]$ for the evaluation phase. $\mathsf{P_B}$ computes the t_ith power of each plaintext $g_{\pi_f(i)}$ of c_i via the multiplicatively homomorphic property of the ElGamal encryption (using pk) to obtain c'_i. Let the resulting list $\Phi' = [c'_1, \ldots, c'_N]$. $\mathsf{P_A}$ computes the information for decryption of all ciphertexts c'_i (remember that $c'_i = (c'^{(0)}_i, c'^{(1)}_i)$), *i.e.*, $\mathsf{P_A}$ computes $d_i \leftarrow (c'^{(0)}_i)^s$ for $i \in [N]$.
   $\mathsf{P_A}$ sends h, Φ, Φ', and $\{d_i\}_{i \in [N]}$ to $\mathsf{P_B}$. Then $\mathsf{P_A}$ uses the functionality $\mathcal{F}^{\mathsf{EncEP}}_{\mathsf{zk}}$ to prove to $\mathsf{P_B}$ that she has performed a valid EP on G to obtain the list of ciphertexts Φ. Meanwhile, $\mathsf{P_A}$ uses $\mathcal{F}^{\mathsf{DH}}_{\mathsf{zk}}$ to prove to $\mathsf{P_B}$ her knowledge of s, *i.e.*, sk, for $(g, \{c'^{(0)}_i\}_{i \in [N]})$ and $(h, \{d_i\}_{i \in [N]})$, together with her knowledge of t_i for the two-tuple ciphertexts c_i and c'_i for all $i \in [N]$.
2. $\mathsf{P_B}$ decrypts all c'_i's to obtain the plaintexts via $p_i \leftarrow c'^{(1)}_i \cdot d_i^{-1}$. $\mathsf{P_B}$ stores a list $P = [p_1, \ldots, p_N]$ for the evaluation phase.

Evaluation phase

0. $\mathsf{P_A}$ chooses uniform κ-bit strings $\{\mathsf{seed}^A_j\}_{j \in [\lambda]}$, computes $\mathsf{c}^{\mathsf{seed}^A_j} \leftarrow \mathsf{Com}(\mathsf{seed}^A_j)$ and sends $\{\mathsf{c}^{\mathsf{seed}^A_j}\}_{j \in [\lambda]}$ to $\mathsf{P_B}$.
   $\mathsf{P_B}$ chooses uniform κ-bit strings $\{\mathsf{seed}^B_j, \mathsf{witness}_j\}_{j \in [\lambda]}$, while $\mathsf{P_A}$ picks $\hat{\jmath} \leftarrow\$ [\lambda]$ and sets $b_{\hat{\jmath}} = 1$ and $b_j = 0$ for $j \neq \hat{\jmath}$. $\mathsf{P_B}$ and $\mathsf{P_A}$ run λ executions of Π_{OT}. In the jth execution, $\mathsf{P_B}$ uses as input $(\mathsf{seed}^B_j, \mathsf{witness}_j)$ and $\mathsf{P_A}$ uses as input b_j with randomness derived from seed^A_j. At the end, $\mathsf{P_A}$ has $\{\mathsf{seed}^B_j\}_{j \neq \hat{\jmath}}$ and $\mathsf{witness}_{\hat{\jmath}}$. Let us denote the transcript of the jth execution by trans_j.
1. For $j \in [\lambda]$, using the randomness derived from seed^B_j, $\mathsf{P_B}$ picks $w^0_{i,j}, w^1_{i,j} \leftarrow\$ \mathbb{G}$ for $i = M+1, \ldots, M+m$ and $\alpha_{0,j}, \alpha_{1,j} \leftarrow\$ \mathbb{Z}_q$. $\mathsf{P_B}$ also computes wire labels and produces garbled gates as in $\Pi_{\mathsf{activePFE}}$. At the end, $\mathsf{P_B}$ obtains the resulting collection of garbled gates $\mathsf{GC}_j = \{\mathsf{GG}_{i,j}\}_{i \in [\theta]}$, $\mathsf{P_A}$'s input-wire

labels $\{(w^0_{i,j}, w^1_{i,j})\}_{i\in[n_A]}$, $\mathsf{P_B}$'s input-wire labels $\{(w^0_{n_A+i,j}, w^1_{n_A+i,j})\}_{i\in[n_B]}$, and output-wire labels of the garbled circuit $\{(w^0_{M+i,j}, w^1_{M+i,j})\}_{i=1,\dots,m}$.

2. $\mathsf{P_A}$ and $\mathsf{P_B}$ are involved in λ executions of Π_{OT}. In the jth execution, $\mathsf{P_B}$ uses as input $(w^0_{i,j}, w^1_{i,j})_{i\in[n_A]}$, while $\mathsf{P_A}$ uses as input x_A if $j = \hat{\jmath}$ and 0^{n_A} otherwise, and random coins of $\mathsf{P_A}$ and $\mathsf{P_B}$ are derived from seed^A_j and seed^B_j, respectively. At the end, $\mathsf{P_A}$ obtains her garbled input $\{\mathsf{x}_i = w^{x_A[i]}_{i,\hat{\jmath}}\}_{i\in[n_A]}$. Let $\mathtt{h}^{\mathsf{OT}}_j$ denote the hash value of the transcript for the jth execution of Π_{OT}.
3. (a) For all $j \in [\lambda]$, $\mathsf{P_B}$ computes $\mathsf{c}^{x_B}_{i,j,b} \leftarrow \mathsf{Com}(w^b_{n_A+i,j})$ for all $i \in [n_B]$ and $b \in \{0,1\}$. Let $\mathtt{h}^{\mathsf{O}}_j$ be the hash value of $\{(w^0_{M+i,j}, w^1_{M+i,j})\}_{i=1,\dots,m}$. $\mathsf{P_B}$ then computes $\mathsf{c}_j \leftarrow \mathsf{Com}(\mathsf{GC}_j, \{\mathsf{c}^{x_B}_{i,j,b}\}_{i\in[n_B],b\in\{0,1\}}, \mathtt{h}^{\mathsf{O}}_j)$, where two elements in each pair $(\mathsf{c}^{x_B}_{i,j,0}, \mathsf{c}^{x_B}_{i,j,1})$ are permuted in random order. The random coins of commitments and permutations are derived from seed^B_j.

   (b) $\mathsf{P_B}$ generates $\sigma_j \leftarrow \mathsf{Sig}_{\mathsf{sigk}}(G, P, j, \mathsf{c}^{\mathsf{seed}^A_j}, \mathsf{trans}_j, \mathsf{h}^{\mathsf{OT}}_j, \mathsf{c}_j)$ for $j \in [\lambda]$.

   Then $\mathsf{P_B}$ sends $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ to $\mathsf{P_A}$.
4. $\mathsf{P_A}$ verifies that whether all σ_j's are valid. If not, $\mathsf{P_A}$ halts and outputs $\perp$. Then $\mathsf{P_A}$ calls $\mathsf{Blame}(\{\mathsf{h}^{\mathsf{OT}}_j, \mathsf{c}_j\}_{j\in[\lambda]\setminus\{\hat{\jmath}\}})$. If the output is cert, $\mathsf{P_B}$ sends cert to $\mathsf{P_B}$, outputs $\mathsf{corrupted}$, and halts. Otherwise, $\mathsf{P_A}$ sends $(\hat{\jmath}, \{\mathsf{seed}^B_j\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ to $\mathsf{P_B}$. $\mathsf{P_B}$ verifies that these values are all consistent with those he has sent in Step 0 and aborts if not.
5. $\mathsf{P_B}$ assigns $\mathsf{x}_{n_A+i} \leftarrow w^{x_B[i]}_{n_A+i,\hat{\jmath}}$ for $i \in [n_B]$. Then $\mathsf{P_B}$ sends $\mathsf{GC}_{\hat{\jmath}}$, $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$, $\{\mathsf{c}^{x_B}_{i,\hat{\jmath},b}\}_{i\in[n_B],b\in\{0,1\}}$ (in the same order as Step 3a), and $\mathtt{h}^{\mathsf{O}}_{\hat{\jmath}}$, together with $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$ and $\{\mathsf{decom}^{\mathsf{c}^{x_B}_{i,\hat{\jmath},x_B[i]}}\}_{i\in[n_B]}$, to $\mathsf{P_A}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathsf{P_B}$ also sends the garbled output mapping $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$ to $\mathsf{P_A}$.
6. $\mathsf{P_A}$ outputs $\perp$ and aborts if $\mathsf{Com}(\mathsf{GC}_j, \{\mathsf{c}^{x_B}_{i,\hat{\jmath},b}\}_{i\in[n_B],b\in\{0,1\}}, \mathtt{h}^{\mathsf{O}}_{\hat{\jmath}}; \mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}) \neq \mathsf{c}_{\hat{\jmath}}$, for some $i \in [n_B]$, $\mathsf{Com}(\mathsf{x}_{n_A+i}; \mathsf{decom}^{\mathsf{c}^{x_B}_{i,\hat{\jmath},x_B[i]}}) \notin \{\mathsf{c}^{x_B}_{i,\hat{\jmath},0}, \mathsf{c}^{x_B}_{i,\hat{\jmath},1}\}$, or $\mathtt{h}^{\mathsf{O}}_{\hat{\jmath}}$ is not consistent (if it is verifiable).

   $\mathsf{P_A}$ computes $\{\mathsf{y}_i\}_{i\in[m]} \leftarrow \mathsf{Eval}(\mathsf{GC}_{\hat{\jmath}}, \{\mathsf{x}_i\}_{i\in[n]}, T, \pi_f)$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathsf{P_A}$ can thereby derive the output. If $\mathsf{y}_i \notin \{w^0_i, w^1_i\}$ for some $i \in \{M+1, \dots, M+m\}$, $\mathsf{P_A}$ outputs $\perp$. If $\mathsf{P_B}$ is allowed to know the evaluation result, $\mathsf{P_A}$ sends $\{\mathsf{y}_i\}_{i\in[m]}$ to $\mathsf{P_B}$ so that $\mathsf{P_B}$ could derive the result. If the output-wire labels are not consistent with those $\mathsf{P_B}$ generated, $\mathsf{P_B}$ outputs $\perp$.

In the following, we provide the algorithms Blame and Judge used in $\Pi_{\mathsf{covertPFE}}$.

Algorithm Blame

Specified parameters: G, P, $\{\mathsf{trans}_j, \sigma_j, \mathsf{seed}^A_j, \mathsf{decom}^{\mathsf{seed}^A_j}, \mathsf{seed}^B_j\}_{j\in[\lambda]\setminus\{\hat{\jmath}\}}$.
Inputs: $\{\mathsf{h}^{\mathsf{OT}}_j, \mathsf{c}_j\}_{j\in[\lambda]\setminus\{\hat{\jmath}\}}$.

1. For all $j \neq \hat{\jmath}$, simulate $\mathsf{P_B}$'s computation in steps 1, 2, and 3a, and particularly compute $\hat{\mathsf{h}}^{\mathsf{OT}}_j$ and $\hat{\mathsf{c}}_j$. Let J be the set of indices, such that for $j \in J$, $(\hat{\mathsf{h}}^{\mathsf{OT}}_j, \hat{\mathsf{c}}_j) \neq (\mathsf{h}^{\mathsf{OT}}_j, \mathsf{c}_j)$.
2. (a) If $|J| = 0$, the algorithm returns accept.

   (b) If $|J| \geq 1$, the algorithm picks $j \leftarrow_{\$} J$ and outputs a certificate $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}^{\mathsf{OT}}_j, \mathsf{c}_j, \sigma_j, \mathsf{seed}^A_j, \mathsf{decom}^{\mathsf{seed}^A_j})$.

Algorithm Judge

Inputs: A verification key vk for the signature scheme, a certificate $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$, common reference string G.

1. Compute $\mathsf{c}^{\mathsf{seed}_j^A} \leftarrow \mathsf{Com}(\mathsf{seed}_j^A; \mathsf{decom}^{\mathsf{seed}_j^A})$.
2. If $\mathsf{Vf}((G, P, j, \mathsf{c}^{\mathsf{seed}_j^A}, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j), \sigma_j) = 0$, output 0.
3. Simulate the execution of Π_{OT} that involves trans_j (Step 0 of the evaluation phase). In this simulation, the input of $\mathsf{P_A}$ is 0, random coins are derived from seed_j^A, and the incoming messages of $\mathsf{P_B}$ are those included in trans_j. Check whether messages sent by $\mathsf{P_A}$ are consistent with that of trans_j and output 0 if not. Otherwise, denote $\mathsf{P_A}$'s output of this simulation of Π_{OT} by seed_j^B.
4. Use seed_j^A and seed_j^B to simulate the execution of Steps 1, 2, and 3a of the evaluation phase, and particularly compute $\hat{\mathsf{h}}_j^{\mathsf{OT}}$ and $\hat{\mathsf{c}}_j$.
5. (a) If $(\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j) = (\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j)$, output 0.
   (b) If $\hat{\mathsf{c}}_j \neq \mathsf{c}_j$, output 1.
   (c) If the first message for which $\hat{\mathsf{h}}_j^{\mathsf{OT}} \neq \mathsf{h}_j^{\mathsf{OT}}$ corresponds to $\mathsf{P_A}$, output 0. Otherwise, output 1.

We present the theorem for the security of the protocol $\Pi_{\mathsf{covertPFE}}$ as follows.

Theorem 6. *If the commitment algorithm* Com *is computationally binding and hiding, the hash function is modeled as a random oracle, the garbling scheme is secure under the random oracle model, the DDH assumption of* $\mathbb{G}$ *holds, perfectly correct protocol* Π_{OT} *UC-realizes* $\mathcal{F}_{\mathsf{OT}}$*, and the signature scheme* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *is* EUF-CMA*, then the protocol* $\Pi_{\mathsf{CovertPFE}}$ *along with* Blame *and* Judge *is publicly verifiable covert secure with deterrence factor* $\epsilon = 1 - \frac{1}{\lambda}$ *in the* $(\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}})$*-hybrid world.*

The proof of this theorem can be found in Appendix G. Following the same discussion as $\Pi_{\mathsf{activePFE}}$, it is easy to see that $\Pi_{\mathsf{covertPFE}}$ could be instantiated as a constant-round PVC-secure PFE protocol with linear complexity. Similarly, it is straightforward that we have the theorem below, and Remark 2 is also applicable to $\Pi_{\mathsf{covertPFE}}$ to achieve global reusability.

Theorem 7. *Once the initiation phase for a private circuit* C_f *is executed, every subsequent execution of the evaluation phase of* $\Pi_{\mathsf{covertPFE}}$ *does not degenerate the security of* $\Pi_{\mathsf{covertPFE}}$*.*

5 Analysis

5.1 Performance of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$

In Table 3, we provide from two directions the communication cost of each part of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ for one execution of $\Pi_{\mathsf{zk}}^{\mathsf{EncEP}}$ with parameters M and N in the honest-verifier zero-knowledge setting. Note that $\Pi_{\mathsf{zk}}^{\mathsf{Zero+}}$ is the optimized protocol of

$\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero}}$ according to the idea introduced in Remark 3. The row of remaining is for the communication cost of $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ excluding the cost of sub-protocols. Since messages sent from V to P are random messages in all protocols, we can leverage the random oracle and compile these protocols to be non-interactive via the Fiat-Shamir heuristic. Using this approach, the communication cost now only takes into account the cost from P to V.

Table 3: Communication cost of each part of $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ with parameters M and N.

Protocols	Bits from P to V	Bits from V to P
$\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Sum}}$	$(4\lceil\log_2 M\rceil + 3)\|\mathbb{G}\| + 4\|\mathbb{Z}_q\|$	$\|\mathbb{G}\| + (\lceil\log_2 M\rceil + 1)\|\mathbb{Z}_q\|$
$\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero}}$	$(2N + 4)\|\mathbb{G}\| + (2M + 3)\|\mathbb{Z}_q\|$	$\|\mathbb{Z}_q\|$
$\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero+}}$	$(4\lceil\log_2(2N + 3)\rceil + 4\lceil\log_2 M\rceil + 12)\|\mathbb{G}\| + (M + 10)\|\mathbb{Z}_q\|$	$2\|\mathbb{G}\| + (\lceil\log_2(2N+3) + \lceil\log_2 M\rceil + 3)\|\mathbb{Z}_q\|$
Remaining	0	$3\|\mathbb{Z}_q\|$

We give comparisons between the previous generic work [34] and our protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ (using the optimized protocol $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{Zero+}}$) in Tables 4 and 5. From Table 4,

Table 4: Communication cost comparison between the previous generic work [34] and $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ in this paper with parameters M and N.

Protocols	Bits from P to V	Bits from V to P
[34]	$\sim (32N\|\mathbb{G}\| + 12N\|\mathbb{Z}_q\|)$	$\sim (2N\|\mathbb{G}\| + 10N\|\mathbb{Z}_q\|)$
This paper	$\sim (4\lceil\log_2(N)\rceil + 8\lceil\log_2 M\rceil)\|\mathbb{G}\| + M\|\mathbb{Z}_q\|$	$\sim (\lceil\log_2 N\rceil + 2\lceil\log_2 M\rceil)\|\mathbb{Z}_q\|$

we can see that the (non-interactive) communication cost of our protocol is around $M\|\mathbb{Z}_q\|$. In comparison, the protocol in [34] cannot be compiled to be non-interactive. Its total communication cost is around $(34N\|\mathbb{G}\| + 22N\|\mathbb{Z}_q\|)$ bits. For a regular circuit, we always have $M < N$. Meanwhile, we have $\|\mathbb{G}\| > \|\mathbb{Z}_q\|$. Hence, the number of the transmitted bits of the previous generic protocol is at least $56\times$ larger than ours.

Table 5: Computation cost comparison between the previous generic work [34] and $\mathit{\Pi}_{\mathsf{zk}}^{\mathsf{EncEP}}$ in this paper with parameters M and N.

Protocols	Time P Expos	Time V Expos
[34]	$\sim 59N$	$\sim 52N$
This paper	$\sim (16N + 11M)$	$\sim (10N + 3M)$

In Table 5, we count the total number of exponentiations that P and V need to perform in these two protocols. It is easy to see that the execution of our protocol should be much faster than the protocol in [34].

5.2 Performance of Our PFE Protocols

In this paper, we provide the *first* constant-round actively secure PFE protocol with linear complexity and the *first* constant-round PVC-secure PFE protocol with linear complexity in the two-party setting. Furthermore, our constructions have comparably good performance with existing passively secure PFE protocols.

The same initiation phase of the two protocols can be compiled to be non-interactive, and the resulting non-interactive information for the initiation phase is around $(8N\|\mathbb{G}\| + 2M\|\mathbb{Z}_q\|)$ bits. The linear constant-round passively secure PFE protocols in [23] and [33] do not achieve reusability, but we can still divide them into the same two phases, such that the phase for preprocessing the circuit C_f is the initiation phase, and the phase for generating, sending the garbled circuit, and evaluating that circuit is the evaluation phase. The communication cost of the initiation phase of the optimized protocol in [23], the protocol in [33], and the protocol in [8] are $(2M + 6N)\|\mathbb{G}\|$ bits, $(2M + 4N)\|\mathbb{G}\|$ bits, and $(M + N)\|\mathbb{G}\|$ bits, respectively. We can see that our protocol is competitive, even if it is actively secure. We also remark that since the protocols in [23] and [33] do not achieve reusability. Their initiation phases require to be executed every time when the same circuit C_f is involved, while the cost of the initiation phase can be amortized to multiple executions if a protocol achieves reusability. Meanwhile, the initiation phase of the protocol in [8] is interactive, and it does not achieve global reusability. In comparison, the initiation phase of our protocol could be non-interactive, and it achieves global reusability.

It is shown that the linear passively secure PFE protocol in [8] outperforms the protocols in [23] and [33] when it is executed any number (more than one) of time for a fixed private circuit. Here, we reason that our PVC-secure protocol does not have too much overhead compared with the passively secure protocol in [8] in the evaluation phase. The additional communication cost of $\Pi_{\mathsf{covertPFE}}$ compared with the passively secure protocol in [8] mainly includes the following.

1. The λ executions of Π_{OT} in Step 0 for seed transmission.
2. The extra $\lambda - 1$ executions of Π_{OT} for input-wire labels retrieval in Step 2.
3. The λ tuples of $\{\mathsf{c}_j, \sigma_j\}$ sent in Step 3.
4. The messages $\{\mathsf{c}_{i,j,b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}$, $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$, $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$, and $\{\mathsf{decom}^{\mathsf{c}_{i,\hat{\jmath},x_B[i]}^{x_B}}\}_{i\in[n_B]}$ sent in Step 5.

Let us analyze the cost of $\Pi_{\mathsf{covertPFE}}$ for the deterrence factor $\epsilon = 1/2$, *i.e.*, $\lambda = 2$. The additional communication cost of Step 1 and Step 3 is constant now. Meanwhile, the additional communication cost of Step 2 and Step 4 now only depends on the input length n of the circuit. For most regular circuits, this cost is significantly smaller than the dominant communication cost of transmitting the garbled gates, which is bounded by $\mathcal{O}(\theta)$ for circuit size θ. The additional computation cost for both parties is mainly from the cost of generating the corresponding GC_j's to compute the commitments c_j's for checked instances. Therefore, for the evaluation phase, the computation cost of both parties in our PVC-secure PFE protocol with $\epsilon = 1/2$ is only around $2.6\times$ that of the passively secure PFE protocol [8], and thus it is acceptable.

Finally, let us see the size of the certificate in our PVC-secure PFE protocol. Note that all elements other than the list P inside a certificate do not depend on the size of the private circuit C_f. If the initiation phase is compiled to be non-interactive, we can assume that all parties have already held the messages generated in the initiation phase, including P. Now we do not need to include P in the certificate, and the size of the certificate is constant.

Acknowledgments. We thank the reviewers for their detailed and helpful comments. Y. Liu and Q. Wang were partially supported by the Shenzhen fundamental research programs under Grant no. 20200925154814002 and Guangdong Provincial Key Laboratory (Grant No. 2020B121201001).

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. Asharov, G., Orlandi, C.: Calling out cheaters: Covert security with public verifiability. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 681–698. Springer (2012)
4. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. J. Cryptol. **23**(2), 281–343 (2010)
5. 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)
6. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: Ortiz, H. (ed.) Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA. pp. 503–513. ACM (1990)
7. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012. pp. 784–796. ACM (2012)
8. 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)
9. 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)
10. 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)
11. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 315–334. IEEE Computer Society (2018)

12. Canetti, R., Sarkar, P., Wang, X.: Blazing fast OT for three-round UC OT extension. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12111, pp. 299–327. Springer (2020)
13. Chang, Y., Lu, C.: Oblivious polynomial evaluation and oblivious neural learning. Theor. Comput. Sci. **341**(1-3), 39–54 (2005)
14. 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)
15. Damgård, I., Orlandi, C., Simkin, M.: Black-box transformations from passive to covert security with public verifiability. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 647–676. Springer (2020)
16. 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)
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 169–178. ACM (2009)
18. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–190 (1996)
19. 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)
20. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010)
21. 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)
22. Hong, C., Katz, J., Kolesnikov, V., Lu, W., Wang, X.: Covert security with public verifiability: Faster, leaner, and simpler. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 97–121. Springer (2019)
23. 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)
24. Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 724–741. Springer (2015)
25. 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)
26. Kolesnikov, V., Malozemoff, A.J.: Public verifiability in the covert model (almost) for free. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 210–235. Springer (2015)
27. 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)
28. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4515, pp. 52–78. Springer (2007)
29. Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009)
30. 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
31. Liu, H., Yu, Y., Zhao, S., Zhang, J., Liu, W., Hu, Z.: Pushing the limits of valiant's universal circuits: Simpler, tighter and more compact. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12826, pp. 365–394. Springer (2021)
32. Liu, Y., Wang, Q., Yiu, S.M.: Blind polynomial evaluation and data trading. In: Applied Cryptography and Network Security - 19th International Conference, ACNS 2021, Kamakura, Japan, June 21-24, 2021. Lecture Notes in Computer Science, Springer (2021)
33. 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)
34. 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)
35. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. **35**(5), 1254–1281 (2006)
36. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM **51**(2), 231–262 (2004)
37. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer (1991)
38. Sander, T., Young, A.L., Yung, M.: Non-interactive cryptocomputing for nc^1. In: 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA. pp. 554–567. IEEE Computer Society (1999)
39. Thaler, J.: Proofs, arguments, and zero-knowledge (2021), http://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf
40. 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)
41. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986. pp. 162–167. IEEE Computer Society (1986)
42. 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)
43. Zhu, R., Ding, C., Huang, Y.: Efficient publicly verifiable 2pc over a blockchain with applications to financially-secure computations. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 633–650. ACM (2019)

A Garbling Scheme

In this section, we present a garbling scheme for the standard SFE (rather than PFE) setting. This scheme is related to the garbling scheme used in our protocols. It aims to help readers understand our protocols in a more comprehensive way, and this also assists our security proof of Theorem 1, Theorem 6, etc. We stress that this garbling scheme is conceptually different from the one we use in our PFE protocols. The main difference is due to the fact that the garbling scheme in our PFE protocols should be coupled with other parts of the protocols, whereas the garbling scheme here is independent, in the sense that it can *only* be used in traditional garbled circuits approach when two parties *commonly* agree with the same list T and EP π_f (and thus the circuit C_f). For instance, wire labels are generated by the two parties together in our PFE protocols, while these labels are simply generated by an algorithm Init in the present scheme. In addition, the generated labels are not the same in these two scenarios.

This garbling scheme consists of algorithms $(\mathsf{Init}, \mathsf{Gb}, \mathsf{Eval})$ based on a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. It is used for a circuit C_f that consists solely of θ NAND gates, with m output wires and n input wires. We denote the number of incoming wires by $N = 2\theta$ and the number of outgoing wires except those that are output wires of the circuit by $M = n + \theta - m$. These parameters are implicitly taken as input by the three algorithms of the scheme. These three algorithms are presented below.

- Init takes as input an EP π_f derived from a circuit C_f (see Section 2.1) and a list $T = [t_1, \dots, t_N]$, where $t_i \in \mathbb{Z}_q$. For each outgoing wire OW_i with index $i \in [M+m]$, the algorithm picks the label $w_i^b \leftarrow_\$ \mathbb{Z}_q$ for $b \in \{0,1\}$. For each incoming wire IW_i with index $i \in [N]$ (connecting with the outgoing wire $\mathsf{OW}_{\pi_f(i)}$), it picks $t_i \leftarrow_\$ \mathbb{Z}_q$ and computes the label $v_i^b \leftarrow (w^b_{\pi_f(i)})^{t_i}$ for $b \in \{0,1\}$. Finally, the algorithm outputs $(\{(w_i^0, w_i^1)\}_{i\in[M+m]}, \{(v_i^0, v_i^1)\}_{i\in[N]})$.
- Gb is invoked to generate garbled gates. According to the circuit representation approach in Section 2.1, a gate G_i consists of two input wires, *i.e.*, incoming wires, with indices $2i-1$ and $2i$, and one output wire, *i.e.*, an outgoing wire, with index $n+i$. For such a gate, Gb takes as input the gate index i, the two pairs of input-wire labels (v^0_{2i-1}, v^1_{2i-1}) and (v^0_{2i}, v^1_{2i}), together with the pair of output-wire labels (w^0_{n+i}, w^1_{n+i}), and prepares four ciphertexts: $c_i^{a,b} \leftarrow \mathsf{Enc}^i_{v^a_{2i-1}, v^b_{2i}}(w^{\overline{a\cdot b}}_{n+i})$ for $a, b \in \{0,1\}$ for a dual-key cipher Enc. Gb outputs the set of garbled gates $\{\mathsf{GG}_i\}_{i\in[\theta]}$. Here $\mathsf{GG}_i = \{c_i^{a,b}\}_{a,b\in\{0,1\}}$, where $c_i^{a,b}$ are randomly permuted.
- Eval takes as input a set of garbled gates $\{\mathsf{GG}_i\}_{i\in[\theta]}$, a set of input-wire labels $\{\mathsf{x}_i\}_{i\in[n]}$, a list $T = \{t_i\}_{i\in[N]}$, and an EP π_f. This algorithm first derives the description of the corresponding circuit C_f from π_f. Now starting from (outgoing-wire) labels $\{\mathsf{x}_i\}_{i\in[n]}$, Eval computes incoming-wire labels from outgoing-wire labels and evaluates garbled gates one by one following the topographical order of the circuit to obtain the final output-wire labels.
  Without loss of generality, for an outgoing wire OW_i, we denote its label in hand by w_i^b, where $b \in \{0,1\}$. Note that each outgoing wire may connect with

some incoming wires that are the input wires of some gates. Assume that an incoming wire IW_j is connected with OW_i. $\mathsf{P_A}$ can compute the corresponding wire label of IW_j by computing the t_jth power of w_i^b, *i.e.*, $(w_i^b)^{t_j}$. Since we have $(w_i^b)^{t_j} = v_j^b$, the result is the input-wire (incoming-wire) label as we want. When having two input-wire (incoming-wire) labels $(v_{2i-1}^b, v_{2i}^{b'})$, where $b, b' \in \{0,1\}$, for a garbled gate GG_i, the algorithm can decrypt GG_i using these two labels as keys (via a simple reverse approach of Enc) and obtain the non-$\perp$ resulting output-wire (outgoing-wire) label $w_{n+i}^{\overline{b \cdot b'}}$. It is easy to see that the values of the wire b and b' are hidden from $\mathsf{P_A}$ during this procedure. Since Eval follows the topology of the circuit, input-wire labels of a gate are always ready when we proceed to evaluate that gate. Finally, Eval returns the decrypted output-wire labels of the output gates.

To simplify our description, we could use a standard dual-key cipher Enc in the random oracle model. Let the random oracle be $\mathsf{H}\colon \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \to \{0,1\}^{\|\mathbb{G}\| \times \tau}$, where τ is an integer specifying length of redundant bits that ensures correct decryption. We define the dual-key cipher as

$$c_i \leftarrow \mathsf{Enc}^i_{v_j^a, v_k^b}(w_\ell^{\overline{a \cdot b}}) = \mathsf{H}(v_j^a, v_k^b, i) \oplus ([w_\ell^{\overline{a \cdot b}}] || 0^\tau)\,,$$

where $[w_\ell^{\overline{a \cdot b}}]||0^\tau$ denote the bit string that is the concatenation of the bit-representation of $w_\ell^{\overline{a \cdot b}}$ and the string of τ zeros. The decryption algorithm Dec for this dual-key cipher takes as input the two keys (v_j, v_k), the index i and ciphertext c_i corresponding to the dual-key cipher Enc, and computes $\mathsf{H}(v_i^a, v_j^b, i) \oplus c_i$. If the last τ bits of the result are all 0, it outputs the group element of $\mathbb{G}$ that represented by the first $\|\mathbb{G}\|$ bits of $\mathsf{H}(v_i^a, v_j^b, i) \oplus c_i$. Otherwise, it outputs $\perp$. Note that this scheme could be further optimized, *e.g.*, via using a variant of the point-and-permute optimization [6] (see [8] for more information).

In the following, we briefly present definitions for the garbling scheme that follows the approach of [7].

Correctness For any EP π_f (for the circuit C_f) and list T as above, and any input $x \in \{0,1\}^n$, we follow the steps below.

1. Run $(\{(w_i^0, w_i^1)\}_{i \in [M+m]}, \{(v_i^0, v_i^1)\}_{i \in [N]}) \leftarrow \mathsf{Init}(\pi_f, T)$.
2. Compute $\{\mathsf{GG}_i\}_{i \in [\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}^0, v_{2i-1}^1), (v_{2i}^0, v_{2i}^1), (w_{n+i}^0, w_{n+i}^1)\}_{i \in [\theta]})$.
3. Let $\mathsf{x}_i = w_i^{x[i]}$ for $i \in [n]$.
4. Execute $\{\mathsf{y}_i\}_{i \in [m]} \leftarrow \mathsf{Eval}(\{\mathsf{GG}\}_{i \in [\theta]}, \{\mathsf{x}_i\}_{i \in n}, T, \pi_f)$.

Then this garbling scheme is correct if for all $y \in [m]$, it holds that $\mathsf{y}_i = w_{M+i}^{y[i]}$, where $y = C_f(x)$.

Privacy We say that the garbling scheme achieves privacy if for any EP π_f, list T, and input x, where the format/length of T and x satisfy the circuit C_f corresponding to π_f, there exists a PPT simulator $\mathcal{S}$, such that the output distributions of the following two procedures are computationally indistinguishable.

– 1. $(\{(w_i^0, w_i^1)\}_{i\in[M+m]}, \{(v_i^0, v_i^1)\}_{i\in[N]}) \leftarrow \mathsf{Init}(\pi_f, T)$.
  2. $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}^0, v_{2i-1}^1), (v_{2i}^0, v_{2i}^1), (w_{n+i}^0, w_{n+i}^1)\}_{i\in[\theta]})$.
  3. $\mathsf{x}_i \leftarrow w_i^{x[i]}$ for $i \in [n]$.
  4. Output $(\{\mathsf{x}_i\}_{i\in[n]}, \{\mathsf{GG}_i\}_{i\in[\theta]}, \{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]})$.

– 1. $(\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}, \{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]}, \{(\widetilde{w}_{M+i}^0, \widetilde{w}_{M+i}^1)\}_{i\in[m]}) \leftarrow \mathcal{S}(C_f(x), \pi_f, T)$.
  2. Output $(\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}, \{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]}, \{(\widetilde{w}_{M+i}^0, \widetilde{w}_{M+i}^1)\}_{i\in[m]})$.

Obliviousness We say that the garbling scheme achieves obliviousness if for any EP π_f, list T, and input x, where the format/length of T and x satisfy the circuit C_f corresponding to π_f, there exists a PPT simulator $\mathcal{S}$, such that the output distributions of the following two procedures are computationally indistinguishable.

– 1. $(\{(w_i^0, w_i^1)\}_{i\in[M+m]}, \{(v_i^0, v_i^1)\}_{i\in[N]}) \leftarrow \mathsf{Init}(\pi_f, T)$.
  2. $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}^0, v_{2i-1}^1), (v_{2i}^0, v_{2i}^1), (w_{n+i}^0, w_{n+i}^1)\}_{i\in[\theta]})$.
  3. $\mathsf{x}_i \leftarrow w_i^{x[i]}$ for $i \in [n]$.
  4. Output $(\{\mathsf{x}_i\}_{i\in[n]}, \{\mathsf{GG}_i\}_{i\in[\theta]})$.

– 1. $(\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}, \{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]}) \leftarrow \mathcal{S}(\pi_f, T)$.
  2. Output $(\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}, \{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]})$.

Authenticity We say that the garbling scheme achieves authenticity if for all PPT adversaries $\mathcal{A}$, the following procedure outputs true with a negligible probability.

1. $(\pi_f, T, x) \leftarrow \mathcal{A}(1^\kappa)$.
2. If the EP π_f, list T, and input x satisfy the pre-agreement of the circuit C_f, continue the procedure. Otherwise, return $\perp$.
3. $(\{(w_i^0, w_i^1)\}_{i\in[M+m]}, \{(v_i^0, v_i^1)\}_{i\in[N]}) \leftarrow \mathsf{Init}(\pi_f, T)$.
4. $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}^0, v_{2i-1}^1), (v_{2i}^0, v_{2i}^1), (w_{n+i}^0, w_{n+i}^1)\}_{i\in[\theta]})$.
5. $\mathsf{x}_i \leftarrow w_i^{x[i]}$ for $i \in [n]$.
6. $\{\mathsf{y}_i\}_{i\in[m]} \leftarrow \mathcal{A}(\{\mathsf{x}_i\}_{i\in[n]}, \{\mathsf{GG}_i\}_{i\in[\theta]})$.
7. $y \leftarrow C_f(x)$.
8. Return $(\forall i \in [m], \mathsf{y}_i \in \{w_{M+i}^0, w_{M+i}^1\}) \wedge (\exists i \in [m], \mathsf{y}_i \neq w_{M+i}^{y[i]})$.

It is straightforward to see that the garbling scheme above is correct. For its security, we present the following theorem. The proof of the theorem simply follows the approach used in [29]. We refer readers to [29] and [8] for more information and details.

Theorem 8. *The garbling scheme* $(\mathsf{Init}, \mathsf{Gb}, \mathsf{Eval})$ *associated with the dual-key cipher* Enc *in the random oracle model above achieves privacy, obliviousness, and authenticity.*

Proof (Sketch). We first prove the privacy of the scheme. Let us define the simulator $\mathcal{S}$ as follows. $\mathcal{S}$ takes as input (y, π_f, T), where $y = C_f(x)$, and goes through the following steps.

1. $\mathcal{S}$ picks labels $\widetilde{w}_i \leftarrow_\$ \mathbb{G}$ for outgoing wires OW_i with indices $i \in [M+m]$. Let $\widetilde{\mathsf{x}}_i \leftarrow \widetilde{w}_i$ for $i \in [n]$. $\mathcal{S}$ also sets $\widetilde{w}_{M+i}^{y[i]} \leftarrow w_{M+i}$ and $\widetilde{w}_{M+i}^{1-y[i]} \leftarrow_\$ \mathbb{G}$ for $i \in [m]$.
2. $\mathcal{S}$ computes $\widetilde{v}_i \leftarrow (\widetilde{w}_{\pi_f(i)})^{t_i}$ for $i \in [N]$. $\mathcal{S}$ picks $\widetilde{v}_i' \leftarrow_\$ \mathbb{G}$ for $i \in [N]$.
3. $\mathcal{S}$ produces $\{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (\widetilde{v}_{2i-1}, \widetilde{v}_{2i-1}'), (\widetilde{v}_{2i}, \widetilde{v}_{2i}'), (\widetilde{w}_{n+i}, \widetilde{w}_{n+i})\})$.
4. $\mathcal{S}$ outputs $(\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}, \{\widetilde{\mathsf{GG}}_i\}_{i\in[\theta]}, \{\widetilde{w}_{M+i}^0, \widetilde{w}_{M+i}^1\}_{i\in[m]})$.

We can see that $\{\mathsf{x}_i\}_{i\in[n]}$ in the real execution and $\{\widetilde{\mathsf{x}}_i\}_{i\in[n]}$ in the simulation are randomly generated from $\mathbb{G}$ and have identical distributions. Then we could simply follow the proof in [29] to design a sequence of hybrid games for the remain proof.

For the garbled gates, starting from the real execution, we could design a sequence of hybrid games. According to the topological order of gates, in each subsequent game, a garbled gate generated as in the real execution is replaced by a garbled gate generated as in the simulation. For two adjacent games, the four ciphertexts (ignoring their order) are computed from the four values generated by the random oracle:

$$\mathsf{H}(v_{2i-1}^0, v_{2i}^0, i) \quad \mathsf{H}(v_{2i-1}^0, v_{2i}^1, i) \quad \mathsf{H}(v_{2i-1}^1, v_{2i}^0, i) \quad \mathsf{H}(v_{2i-1}^1, v_{2i}^1, i)$$

in the real execution and

$$\mathsf{H}(\widetilde{v}_{2i-1}, \widetilde{v}_{2i}, i) \quad \mathsf{H}(\widetilde{v}_{2i-1}, \widetilde{v}_{2i}', i) \quad \mathsf{H}(\widetilde{v}_{2i-1}', \widetilde{v}_{2i}, i) \quad \mathsf{H}(\widetilde{v}_{2i-1}', \widetilde{v}_{2i}', i)$$

in the simulation. Since the *inactive keys*, *i.e.*, the incoming-wire labels of the gate that the circuit evaluator does not obtain (see more in [29]), is derived from the corresponding randomly generated outgoing-wire labels and a fixed t_i's, the keys themselves are totally random. Hence, the four ciphertexts are computationally indistinguishable with respect to the random oracle.

To show that the output mapping in the real execution is computationally indistinguishable from that in the simulation, we could also construct a sequence of hybrid games similar to the approach above to prove the result. Therefore, we can see that the garbling scheme achieves privacy. Following a very similar approach, we can also prove that the garbling scheme achieves obliviousness.

Finally, we prove that the garbling scheme achieves authenticity. Assume that there exists a garbled output y_i from $\mathcal{A}$ that satisfies $\mathsf{y}_i \in \{w_{M+i}^0, w_{M+i}^1\}$ and $\mathsf{y}_i \neq w_{M+i}^{y[i]}$. According to the analysis of the privacy for the garbling scheme, the (inactive) keys to encrypt $w_{M+i}^{1-y[i]}$ are randomly generated and hidden from $\mathcal{A}$. Hence, the usage of the random oracle ensures that $\mathcal{A}$ can only successfully derive $w_{M+i}^{1-y[i]}$ with a negligible probability. □

Based on the definition of the garbling scheme $(\mathsf{Init}, \mathsf{Gb}, \mathsf{Eval})$ introduced above, we can restate Theorem 1 as follows.

Theorem 9. *If the garbling scheme* $(\mathsf{Init}, \mathsf{Gb}, \mathsf{Eval})$ *with respect to the dual-key cipher* Enc *achieves privacy, obliviousness, and authenticity, and the DDH assumption of* $\mathbb{G}$ *holds, the protocol* $\Pi_{\mathsf{activePFE}}$ *securely realizes* $\mathcal{F}_{\mathsf{activePFE}}$ *in the presence of malicious* $\mathsf{P_A}$ *and semi-honest* $\mathsf{P_B}$ *in the* $(\mathcal{F}_{\mathsf{OT}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}})$*-hybrid world.*

B Proof of Theorem 1 and Theorem 9

Proof. Firstly, we focus on the case that $\mathsf{P_A}$ is malicious. For an adversary $\mathcal{A}$ corrupting $\mathsf{P_A}$ in the $(\mathcal{F}_{\mathsf{OT}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}, \mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}})$-hybrid world, we construct a simulator $\mathcal{S}$ that runs $\mathcal{A}$ as a subroutine and plays the role of $\mathsf{P_B}$ in the ideal world. We present the simulation procedures for both the initiation and evaluation phases (denoted by $\mathbf{Game}_0$). The simulator $\mathcal{S}$ simulates the initiation phase as follows.

1. $\mathcal{S}$ picks $G = [g_1, \ldots, g_M]$ as in the protocol. Then $\mathcal{S}$ sends G to $\mathcal{A}$.
2. $\mathcal{S}$ receives h, Φ, Φ', and $\{d_i\}_{i\in[N]}$ from $\mathcal{A}$. Then $\mathcal{S}$ receives the EP π_f (and corresponding random coins) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$. $\mathcal{S}$ verifies whether π_f and the corresponding random coins are correct. If not, $\mathcal{S}$ sends abort_A to $\mathcal{F}_{\mathsf{activePFE}}$ and simulates the termination of $\mathsf{P_B}$. $\mathcal{S}$ also receives s and $\{t_i\}_{i\in[N]}$ from $\mathcal{A}$ for $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ and verifies them following a similar procedure as for $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$.
3. $\mathcal{S}$ computes $P = [p_1, \ldots, p_N]$ as in the protocol.

$\mathcal{S}$ derives the evaluated circuit C_f from the EP π_f. Then $\mathcal{S}$ sends C_f to $\mathcal{F}_{\mathsf{activePFE}}$ and proceeds to simulate the evaluation phase.

1. First, $\mathcal{S}$ chooses $\alpha_i \leftarrow_\$ \mathbb{Z}_q$ for $i \in [M]$ and computes $w_i \leftarrow g_i^{\alpha_i}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w_i \leftarrow_\$ \mathbb{G}$ for $i = M+1, \ldots, M+m$ for output-wire labels of output gates. $\mathcal{S}$ also computes $v_i \leftarrow (w_{\pi_f(i)})^{t_i}$ for $i \in [N]$ and picks $v'_i \leftarrow_\$ \mathbb{G}$ for $i \in [N]$.
   $\mathcal{S}$ computes $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}, v'_{2i-1}), (v_{2i}, v'_{2i}), (w_{n+i}, w_{n+i})\}_{i\in[\theta]})$.
2. $\mathcal{S}$ obtains x_A from $\mathcal{A}$'s input to $\mathcal{F}_{\mathsf{OT}}$ and sends $\{w_i\}_{i\in[n_A]}$ as output of $\mathcal{F}_{\mathsf{OT}}$ to $\mathcal{A}$. $\mathcal{S}$ sends x_A to the ideal functionality $\mathcal{F}_{\mathsf{activePFE}}$ and receives the evaluation result $y \in \{0,1\}^m$ or $\mathsf{nothing}$ depending on whether $\mathsf{P_A}$ is allowed to receive the evaluation result.
3. $\mathcal{S}$ sends $\mathsf{GC} = \{\mathsf{GG}_i\}_{i\in[\theta]}$ and $\{w_{n_A+i}\}_{i\in[n_B]}$ to $\mathcal{A}$.
   - If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also sends the mapping $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$ to $\mathcal{A}$, where $w^{y[i]}_{M+i} = w_{M+i}$ and $w^{1-y[i]}_{M+i} \leftarrow_\$ \mathbb{G}$. Then $\mathcal{S}$ outputs what $\mathcal{A}$ outputs to conclude the simulation.
   - If $\mathsf{P_A}$ is not allowed to know the evaluation result, $\mathcal{S}$ continues to the next step.
4. $\mathcal{S}$ receives from $\mathcal{A}$ the output-wire labels $\{\tilde{w}_{M+i}\}_{i\in[m]}$. If all elements of $\{\tilde{w}_{M+i}\}_{i\in[m]}$ are consistent with those of $\{w_{M+i}\}_{i\in[m]}$, $\mathcal{S}$ sends $\mathsf{continue}$ to $\mathcal{F}_{\mathsf{activePFE}}$. Otherwise, $\mathcal{S}$ sends abort_A to $\mathcal{F}_{\mathsf{activePFE}}$.

Note that the messages that $\mathcal{A}$ receives include G, $\{w_i\}_{i\in[n]}$, GC, and (possibly) $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$. It remains to show that the joint distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ and the output of $\mathsf{P_B}$ in the ideal world is indistinguishable from the joint distribution of the view of $\mathcal{A}$ and the output of $\mathsf{P_B}$ in the real world. We define the following games and let the output of each game be the view of $\mathcal{A}$ and output of $\mathsf{P_B}$.

$\mathbf{Game}_1$ We modify the evaluation phase of $\mathbf{Game}_0$ as follows.

1. $\mathcal{S}$ chooses $\alpha_{i,0}, \alpha_{i,1} \leftarrow\!\!\$\, \mathbb{Z}_q$ for $i \in [M]$. $\mathcal{S}$ computes $w_i^0 \leftarrow g_i^{\alpha_{i,0}}$ and $w_i^1 \leftarrow g_i^{\alpha_{i,1}}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w_i^0, w_i^1 \leftarrow\!\!\$\, \mathbb{G}$ for $i = M+1, \ldots, M+m$ for output-wire labels of output gates as in the protocol. $\mathcal{S}$ also computes $v_i^0 \leftarrow (w_{\pi_f(i)}^0)^{t_i}$ and $v_i^1 \leftarrow (w_{\pi_f(i)}^1)^{t_i}$ for $i \in [N]$.
   $\mathcal{S}$ produces $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}^0, v_{2i-1}^1), (v_{2i}^0, v_{2i}^1), (w_{n+i}^0, w_{n+i}^1)\}_{i\in[\theta]})$.
2. $\mathcal{S}$ returns $\{w_i^{x_A[i]}\}_{i\in[n_A]}$ as the output of $\mathcal{F}_{\mathsf{OT}}$ to $\mathcal{A}$. The rest of the procedure in this step is the same as $\mathbf{Game}_0$.
3. $\mathcal{S}$ sends $\mathsf{GC} = \{\mathsf{GG}_i\}_{i\in[\theta]}$ and $\{w_{n_A+i}^{x_B[i]}\}_{i\in[n_B]}$ to $\mathcal{A}$.
   - If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also sends the mapping $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]}$ to $\mathcal{A}$, and then $\mathcal{S}$ outputs what $\mathcal{A}$ outputs to conclude the simulation.
   - If $\mathsf{P_A}$ is not allowed to know the evaluation result, $\mathcal{S}$ continues to the next step.
4. $\mathcal{S}$ receives from $\mathcal{A}$ the output-wire labels $\{\tilde{w}_{M+i}\}_{i\in[m]}$. Now $\{w_{M+i}^{y[i]}\}_{i\in[m]}$ is used for the consistency check. The rest of the procedure in this step is the same as $\mathbf{Game}_0$.

The security (privacy or obliviousness depending on whether $\mathsf{P_A}$ obtains the evaluation result) of the garbling scheme presented in Appendix A guarantees that the output of $\mathbf{Game}_1$ is computationally indistinguishable from the output of $\mathbf{Game}_0$.

$\mathbf{Game}_2$ We now modify Step 4 of the evaluation phase for the consistency check.

4. $\mathcal{S}$ receives from $\mathcal{A}$ the output-wire labels $\{\tilde{w}_{M+i}\}_{i\in[m]}$. $\mathcal{S}$ checks whether $\tilde{w}_{M+i} \in \{(w_{M+i}^0, w_{M+i}^1)\}$ for all $i \in [m]$ as in the protocol. The rest of the procedure in this step is the same as the previous game.

Due to the security (authenticity) of the garbling scheme presented in Appendix A, $\mathcal{A}$ can only derive $w_{M+i}^{y[i]}$ from GC. Hence, $\mathcal{A}$ cannot deduce information about $w_{M+i}^{1-y[i]}$, and the output of $\mathbf{Game}_2$ is computationally indistinguishable from the output of $\mathbf{Game}_1$.

$\mathbf{Game}_3$ We modify the first step in the evaluation phase of the previous game as follows.

1. $\mathcal{S}$ chooses $\alpha_0 \leftarrow\!\!\$\, \mathbb{Z}_q$ and $\alpha_{i,1} \leftarrow\!\!\$\, \mathbb{Z}_q$ for $i \in [M]$. $\mathcal{S}$ computes $w_i^0 \leftarrow g_i^{\alpha_0}$ and $w_i^1 \leftarrow g_i^{\alpha_{i,1}}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w_i^0, w_i^1 \leftarrow\!\!\$\, \mathbb{G}$ for $i = M + 1, \ldots, M+m$ for output-wire labels of output gates as in the protocol.
   The rest of the procedure in this step is the same as the previous game.

The difference between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ is that $\mathcal{S}$ fixes one α_0 instead of generating a set of $\alpha_{i,0}$'s. In this setting, a subset of elements in $\{w_i^0\}_{i\in[M]}$ that received or derived by $\mathcal{A}$ are involved. Let us denote the index set for this subset by S, and thus the subset in $\mathbf{Game}_2$ by $\hat{W} = \{\hat{w}_i^0\}_{i\in S}$ and in $\mathbf{Game}_3$ by $\hat{W}' = \{\hat{w'}_i^0\}_{i\in S}$. The difference between $\hat{W}$ and $\hat{W}'$ is that the elements in $\hat{W}$ is of the form $\hat{w}_i^0 = g_i^{\alpha_{0,i}}$ for random $\alpha_{0,i}$ while the elements in $\hat{W}'$ is of the form $\hat{w'}_i^0 = g_i^{\alpha_0}$ for random but fixed α_0. According to Lemma 1, $\hat{W}$ is computationally indistinguishable from $\hat{W}'$, and thus the output of $\mathbf{Game}_3$ is computationally indistinguishable from the output of $\mathbf{Game}_2$.

$\mathbf{Game}_4$ The first step in the evaluation phase of the previous game is modified in the following.

1. $\mathcal{S}$ chooses $\alpha_0, \alpha_1 \leftarrow_{\$} \mathbb{Z}_q$ for $i \in [M]$. $\mathcal{S}$ computes $w_i^0 \leftarrow g_i^{\alpha_0}$ and $w_i^1 \leftarrow g_i^{\alpha_1}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w_i^0, w_i^1 \leftarrow_{\$} \mathbb{G}$ for $i = M+1, \ldots, M+m$ for output-wire labels of output gates as in the protocol. The rest of the procedure in this step is the same as the previous game.

Following the same argument as $\mathbf{Game}_3$, the output of $\mathbf{Game}_4$ is computationally indistinguishable from the output of $\mathbf{Game}_3$.

$\mathbf{Game}_5$ We continue to modify the first step in the evaluation phase of the previous game. $\mathcal{S}$ now computes v_i^0 and v_i^1 via $v_i^0 \leftarrow (p_i)^{\alpha_0}$ and $v_i^1 \leftarrow (p_i)^{\alpha_1}$ for $i \in [N]$.

Since we have $p_i = g_{\pi(i)}^{t_i}$ in the initiation phase, we know $v_i^b = (p_i)^{\alpha_b} = g_{\pi(i)}^{t_i \alpha_b} = (w_{\pi_f(i)}^b)^{t_i}$ for $b \in \{0,1\}$. Therefore, the output of $\mathbf{Game}_5$ is perfectly indistinguishable from the output of $\mathbf{Game}_4$.

Note that $\mathbf{Game}_5$ corresponds to a real execution of the protocol, and the output in $\mathbf{Game}_5$ is computationally indistinguishable from the output of $\mathbf{Game}_0$. Thus, we complete the proof for malicious $\mathsf{P_A}$.

Now we focus on the case that $\mathsf{P_B}$ is semi-honest. For an adversary $\mathcal{A}$ corrupting $\mathsf{P_B}$ in the real world, we construct a simulator $\mathcal{S}$ that simulates $\mathsf{P_B}$'s view. Now we present the simulation procedures for the initiation phase and evaluation phase (denoted by $\mathbf{Game}_0$). The simulator $\mathcal{S}$ simulates the initiation phase as follows.

1. $\mathcal{S}$ picks $g_i \leftarrow_{\$} \mathbb{G}$ to generate the list G as in the protocol.
2. $\mathcal{S}$ picks $h \leftarrow_{\$} \mathbb{G}$. Then $\mathcal{S}$ generates $c_i \leftarrow_{\$} \mathbb{G}^2$, $c_i' \leftarrow_{\$} \mathbb{G}^2$, and $d_i \leftarrow_{\$} \mathbb{G}$ for $i \in [N]$. $\mathcal{S}$ generates accept's as the outputs of $\mathcal{A}$ from $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$.
3. $\mathcal{S}$ computes the list P as in the protocol.

Then in the evaluation phase, $\mathcal{S}$ follows the simulation procedure below.

1. $\mathcal{S}$ randomly picks α_0, α_1, and derives $\{w_i^0, w_i^1\}_{i \in [M+m]}$ as in the protocol. $\mathcal{S}$ then generates garbled gates $\{\mathsf{GG}_i\}_{i \in [\theta]}$ as in the protocol.
2. $\mathcal{S}$ simulates the executions of $\mathcal{F}_{\mathsf{OT}}$ as specified in the protocol.
3. $\mathcal{S}$ follows the instructions of $\mathsf{P_B}$ in this step as in the protocol.
4. If $\mathsf{P_B}$ is allowed to know the evaluation result $y \in \{0,1\}^m$, $\mathcal{S}$ sets $\{w_{M+i}^{y[i]}\}_{i \in [m]}$ as the messages sent from $\mathsf{P_A}$.

It remains to show that the distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ in the ideal world is indistinguishable from the distribution of the view of $\mathcal{A}$ in the real world. We first prove the following lemma.

Lemma 2. *If the ElGamal encryption scheme with* $\mathsf{pk} = (\mathbb{G}, q, g, h)$ *is* IND-CPA *secure for security parameter* κ *under the DDH assumption for* $\mathbb{G}$*, given* $\hat{g} \leftarrow_{\$} \mathbb{G}$*,* $(\mathsf{pk}, \hat{g}, c_0, c_0', d_0)$ *is computationally indistinguishable from* $(\mathsf{pk}, \hat{g}, c_1, c_1', d_1)$*, where* $c_0 \leftarrow_{\$} \mathbb{G}^2$*,* $c_0' \leftarrow_{\$} \mathbb{G}^2$*,* $d_0 \leftarrow_{\$} \mathbb{G}$*,* $c_1 \leftarrow (g^r, \hat{g}h^r)$*,* $c_1' \leftarrow (g^{rt}, \hat{g}^t h^{rt})$ *and* $d_1 \leftarrow \hat{g}^t$ *for* $r \leftarrow_{\$} \mathbb{Z}_q$ *and* $t \leftarrow_{\$} \mathbb{Z}_q$*.*

Proof. We consider a hybrid distribution $(\mathsf{pk}, \hat{g}, c_2, c_2', d_2)$ generated as follows:

$$\hat{g} \leftarrow_{\$} \mathbb{G}; \quad t \leftarrow_{\$} \mathbb{Z}_q; \quad c_2 \leftarrow \mathbb{G}^2; \quad c_2' \leftarrow ((c_2^{(0)})^t, (c_2^{(1)})^t); \quad d_2 \leftarrow \hat{g}^t\,.$$

It is straightforward to see that $(\mathsf{pk}, \hat{g}, c_2, c_2', d_2)$ and $(\mathsf{pk}, \hat{g}, c_0, c_0', d_0)$ are computationally indistinguishable according to Lemma 1. We can also see that $(\mathsf{pk}, \hat{g}, c_2, c_2', d_2)$ and $(\mathsf{pk}, \hat{g}, c_1, c_1', d_1)$ are computationally indistinguishable. For the IND-CPA experiment of the ElGamal encryption scheme, we pick $\hat{g} \leftarrow_{\$} \mathbb{G}$ and $t \leftarrow_{\$} \mathbb{Z}_q$. Then we send $\hat{g}$ and a random element $s \in \mathbb{G}$ to the encryption oracle. We receive a ciphertext c from the oracle. Let $c' \leftarrow ((c^{(0)})^t, (c^{(1)})^t)$ and $a = (\mathsf{pk}, \hat{g}, c, c', \hat{g}^t)$. If c encrypts $\hat{g}$, the distribution of a is identical to that of $(\mathsf{pk}, \hat{g}, c_1, c_1', d_1)$. If c encrypts s, the distribution of a is identical to that of $(\mathsf{pk}, \hat{g}, c_2, c_2', d_2)$. If $(\mathsf{pk}, \hat{g}, c_2, c_2', d_2)$ and $(\mathsf{pk}, \hat{g}, c_1, c_1', d_1)$ are not computationally indistinguishable, the IND-CPA security of the ElGamal encryption scheme is broken. Therefore, we complete our proof. ■

For convenience of presentation, let the list $D = [d_1, \ldots, d_N]$. We define the following game and let the output of the game be the view of $\mathcal{A}$.

Game$_1$ The list Φ is generated as in the protocol according to π_f, where π_f is the EP derived from C_f. Then Φ' and d_i are computed as in the protocol. According to Lemma 2, the output of **Game**$_1$ is computationally indistinguishable from the output of **Game**$_0$. More concretely, we define a sequence of hybrids. In the kth hybrid, the first k elements of Φ, Φ', and D are computed as in **Game**$_1$, while other elements of these two lists are generated as in **Game**$_0$. Then we can use the hybrid tuple from the experiment of Lemma 2 in the place of c_k, c_k', and d_k. Now we can easily simulate a hybrid which may be either the $(k-1)$th hybrid or the kth hybrid. Therefore, it is straightforward to see that the output of **Game**$_1$ and the output of **Game**$_0$ should be computationally indistinguishable.

It is easy to see that **Game**$_1$ is corresponding to the real execution, and thus the distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ in the ideal world is indistinguishable from the distribution of the view of $\mathcal{A}$ in the real world. The proof is thus completed. □

C Proof of Theorem 2

Proof. We prove the scenario where one initiation phase and two evaluation phases are executed. The case that more than two evaluation phases are involved can be proved following a similar flow. Here $\mathsf{P_A}$ in the ideal world submits the circuit C_f to the ideal functionality $\mathcal{F}_{\mathsf{activePFE}}$, and then both parties can submit their private inputs to $\mathcal{F}_{\mathsf{activePFE}}$ and get the output (the evaluation results or $\mathsf{nothing}$) twice.

The simulator $\mathcal{S}$ follows the strategy used in the proof of Theorem 1 to simulate the view of the adversary $\mathcal{A}$ controlling $\mathsf{P_A}$ in the initiation phase and two evaluation phases sequentially. We use a sequence of games as in the proof of Theorem 1 to show that the view of $\mathcal{A}$ and the output of $\mathsf{P_B}$ in the ideal world is computationally indistinguishable from those in the real world. We can follow the same argument as in the proof of Theorem 1 except for the following difference.

The difference between this proof and the proof of Theorem 1 are the comparisons between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ and between $\mathbf{Game}_3$ and $\mathbf{Game}_4$. Let us focus on the comparison between $\mathbf{Game}_2$ and $\mathbf{Game}_3$, and the same approach can be used for the comparison between $\mathbf{Game}_3$ and $\mathbf{Game}_4$. Different from the proof of Theorem 1, elements of $\{w^0_{b,i}\}_{i\in[M]}$ in the two executions, *i.e.*, $b = 1, 2$, are derived from the same list $G = [g_1, \dots, g_M]$, and thus we need to consider their joint distributions. More concretely, there exists a set S, such that in the two executions 1 and 2, elements of $\{w^0_{1,i}\}_{i\in S}$ and $\{w^0_{2,i}\}_{i\in S}$ are both received or derived by $\mathcal{A}$. We denote the set of $\{w^0_{1,i}\}_{i\in S}$ and $\{w^0_{2,i}\}_{i\in S}$ in $\mathbf{Game}_2$ by $\hat{W}_1 = \{\hat{w}^0_{1,i}\}_{i\in S}$ and $\hat{W}_2 = \{\hat{w}^0_{2,i}\}_{i\in S}$, and in $\mathbf{Game}_3$ by $\hat{W}'_1 = \{\hat{w'}^0_{1,i}\}_{i\in S}$ and $\hat{W}'_2 = \{\hat{w'}^0_{2,i}\}_{i\in S}$. The difference between $(\hat{W}_1, \hat{W}_2)$ and $(\hat{W}'_1, \hat{W}'_2)$ is that elements in $(\hat{W}_1, \hat{W}_2)$ are of the form $(\{g_i^{\alpha_{1,i,0}}\}_{i\in S}, \{g_i^{\alpha_{2,i,0}}\}_{i\in S})$ for random $\alpha_{1,i,0}$'s and $\alpha_{2,i,0}$'s, while elements in $(\hat{W}'_1, \hat{W}'_2)$ are of the form $(\{g_i^{\alpha_{1,0}}\}_{i\in S}, \{g_i^{\alpha_{2,0}}\}_{i\in S})$ for random but fixed $\alpha_{1,0}$ and $\alpha_{2,0}$. Let $m \leftarrow |S|$, $\hat{g}_i \leftarrow g_i$ for $i \in S$, $\beta \leftarrow \alpha_{1,0}$, $\beta' \leftarrow \alpha_{2,0}$, and $\beta_i \leftarrow \alpha_{1,i,0}$, $\beta_{m+i} \leftarrow \alpha_{2,i,0}$ for $i \in [m]$. Our goal now is to prove that $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta_i}\}_{i\in[m]}, \{\hat{g}_i^{\beta_{m+i}}\}_{i\in[m]})$ is computationally indistinguishable from $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta}\}_{i\in[m]}, \{\hat{g}_i^{\beta'}\}_{i\in[m]})$. We first prove the following lemma.

Lemma 3. *Under the DDH assumption for the cyclic group $\mathbb{G}$ of prime order $q \in \Theta(2^\kappa)$, for any positive integer $m = \mathsf{poly}(\kappa)$, given elements $\hat{g}_1, \dots, \hat{g}_m \leftarrow_\$ \mathbb{G}$, $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta_i}\}_{i\in[m]}, \{\hat{g}_i^{\beta_{m+i}}\}_{i\in[m]})$, where $\beta_1, \dots, \beta_{2m} \leftarrow_\$ \mathbb{Z}_q$, is computationally indistinguishable from $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta}\}_{i\in[m]}, \{\hat{g}_i^{\beta'}\}_{i\in[m]})$, where $\beta, \beta' \leftarrow_\$ \mathbb{Z}_q$.*

Proof. We define the following games.

$\mathbf{Game}_0$ This game is for the distribution of $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta_i}\}_{i\in[m]}, \{\hat{g}_i^{\beta_{m+i}}\}_{i\in[m]})$, where $\hat{g}_1, \dots, \hat{g}_m \leftarrow_\$ \mathbb{G}$ and $\beta_1, \dots, \beta_{2m} \leftarrow_\$ \mathbb{Z}_q$.

$\mathbf{Game}_1$ In this game, we let $\beta \leftarrow_\$ \mathbb{Z}_q$. The distribution of in the previous game is modified to be $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta}\}_{i\in[m]}, \{\hat{g}_i^{\beta_{m+i}}\}_{i\in[m]})$. We now show that the distribution of $\mathbf{Game}_1$ and $\mathbf{Game}_0$ is computationally indistinguishable by constructing a distinguisher $\mathcal{D}$ attacking the corresponding experiment of Lemma 1 using an adversary $\mathcal{A}$ that can distinguish the distribution in $\mathbf{Game}_1$ and the distribution in $\mathbf{Game}_0$.
The distinguisher $\mathcal{D}$ runs $\mathcal{A}$ internally. When receiving $(\{g_i\}_{i\in[m]}, \{h_i\}_{i\in[m]})$, $\mathcal{D}$ picks $\gamma_i \leftarrow_\$ \mathbb{Z}_q$ and computes $h'_i = g_i^{\gamma_i}$ for $i \in [m]$. Then $\mathcal{D}$ sends to $\mathcal{A}$ the message $(\{g_i\}_{i\in[m]}, \{h_i\}_{i\in[m]}, \{h'_i\}_{i\in[m]})$ and outputs what $\mathcal{A}$ outputs. We note that if $\{h_i\}_{i\in[m]}$ is of the form $\{g_i^{\alpha_i}\}_{i\in[n]}$, the message sent to $\mathcal{A}$ is identical to the distribution in $\mathbf{Game}_0$, otherwise if it is of the form $\{g_i^{\alpha}\}_{i\in[n]}$, the message sent to $\mathcal{A}$ is identical to the distribution in $\mathbf{Game}_1$. If $\mathcal{A}$ can distinguish $\mathbf{Game}_0$ and $\mathbf{Game}_1$ with a non-negligible probability, the distinguisher $\mathcal{D}$ can successfully attack the corresponding experiment of Lemma 1, which contradict to the DDH assumption. Hence, the distribution in $\mathbf{Game}_1$ is computationally indistinguishable from the distribution in $\mathbf{Game}_0$.

$\mathbf{Game}_2$ In this game, we let $\beta' \leftarrow_\$ \mathbb{Z}_q$. We modify the distribution in the previous game to be $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta}\}_{i\in[m]}, \{\hat{g}_i^{\beta'}\}_{i\in[m]})$. Using the same argument, we can easily prove that the distribution in $\mathbf{Game}_2$ is computationally indistinguishable from the distribution in $\mathbf{Game}_1$.

Therefore, the distribution in $\mathbf{Game}_2$ is computationally indistinguishable from the distribution in $\mathbf{Game}_0$, and the proof is completed. ■

From the lemma, we have that $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta_i}\}_{i\in[m]}, \{\hat{g}_i^{\beta_{m+i}}\}_{i\in[m]})$ is computationally indistinguishable from $(\{\hat{g}_i\}_{i\in[m]}, \{\hat{g}_i^{\beta}\}_{i\in[m]}, \{\hat{g}_i^{\beta'}\}_{i\in[m]})$. For elements not in the set S, we can simply follow the same argument in the proof of Theorem 1. Thus, we prove that the output of $\mathbf{Game}_2$ and the output of $\mathbf{Game}_3$ (and also the output of $\mathbf{Game}_3$ and the output of $\mathbf{Game}_4$) are computationally indistinguishable.

Then following the same procedure as in the proof of Theorem 1, we complete the proof for $\mathsf{P_A}$.

Now we focus on the case that $\mathsf{P_B}$ is semi-honest. For an adversary $\mathcal{A}$ corrupting $\mathsf{P_B}$ in the real world, we construct a simulator $\mathcal{S}$ that simulates $\mathsf{P_B}$'s view. Now we present the simulation procedures for the initiation phase and evaluation phase (denoted by $\mathbf{Game}_0$). The simulator $\mathcal{S}$ simulates the initiation phase as follows.

1. $\mathcal{S}$ picks $g_i \leftarrow_\$ \mathbb{G}$ to generate the list G as in the protocol.
2. $\mathcal{S}$ picks $h \leftarrow_\$ \mathbb{G}$ and generates $c_i \leftarrow_\$ \mathbb{G}^2$, $c_i' \leftarrow_\$ \mathbb{G}^2$, and $d_i \leftarrow_\$ \mathbb{G}$ for $i \in [N]$. Then $\mathcal{S}$ generates accept's as the outputs from $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$.
3. $\mathcal{S}$ computes the list P as in the protocol.

Then in the two evaluation phases, $\mathcal{S}$ follows the simulation procedure below for both of them.

1. $\mathcal{S}$ randomly picks α_0, α_1, and derives $\{w_i^0, w_i^1\}_{i\in[M+m]}$ as in the protocol. $\mathcal{S}$ then generates garbled gates $\{\mathsf{GG}_i\}_{i\in[\theta]}$ as in the protocol.
2. $\mathcal{S}$ simulates the executions of $\mathcal{F}_{\mathsf{OT}}$ as specified in the protocol.
3. $\mathcal{S}$ follows the instructions of $\mathsf{P_B}$ in this step as in the protocol.
4. If $\mathsf{P_B}$ is allowed to know the evaluation result $y \in \{0,1\}^m$, $\mathcal{S}$ sets $\{w_{M+i}^{y[i]}\}_{i\in[m]}$ as the messages sent from $\mathsf{P_A}$.

It remains to show that the distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ in the ideal world is indistinguishable from the distribution of the view of $\mathcal{A}$ in the real world. We can follow the same procedure in the proof of Theorem 1 for the initiation phase. For the two evaluation phases, it is easy to see that they are independent, and thus we follow again the proof of Theorem 1 for the evaluation phase. The proof is thus completed. □

D Proof of Theorem 3

Proof. We first focus on the completeness of the protocol. Note that for $i \in [N]$, $\vec{e}_i$ is of the form that exact one entry is 1 and other entries are all 0 if and only if $\vec{e}_i\vec{e}_i = \vec{e}_i$ and $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$. Then for $x \leftarrow_{\$} \mathbb{Z}_q$, we have

$$\sum_{i=1}^{N} x^i \vec{e}_i\vec{e}_i = \sum_{i=1}^{N} x^i \vec{e}_i\,.$$

Since $\vec{d}_i = x^i\vec{e}_i$ and $\vec{d} = \sum_i^N \vec{d}_i$, we can rewrite the above equation as $\sum_{i=1}^N \vec{d}_i\vec{e}_i - \vec{d} = 0$. Moreover, given y for the bilinear mapping $*$, we have

$$\sum_{i=1}^{N} \vec{d}_i * \vec{e}_i - \vec{1} * \vec{d} = 0\,.$$

For $\omega \leftarrow_{\$} \mathbb{Z}_q$, we have

$$\sum_{i=1}^{N} \omega^i \vec{1}^{\mathrm{T}}\vec{e}_i = \sum_{i=1}^{N} \omega^i\,.$$

Let $\Omega = \sum_{i=1}^N \omega^i$ and $\vec{e} = \sum_{i=1}^N \omega^i\vec{e}_i$. We can also rewrite the above equation as $\sum_{i=1}^N \omega^i\vec{1}^{\mathrm{T}}\vec{e}_i = \vec{1}^{\mathrm{T}}(\sum_{i=1}^N \omega^i\vec{e}_i) = \vec{1}^{\mathrm{T}}\vec{e} = \Omega$. Now it is easy to see that the protocol is complete when sub-protocols $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ are complete.

For the honest-verifier zero-knowledge property, we construct a simulator $\mathcal{S}$ as follows. $\mathcal{S}$ picks the challenges $x, y \leftarrow_{\$} \mathbb{Z}_q$, computes $c_{\vec{d}_i}$ for $i \in [N]$, $c_{\vec{d}}$, $c_{-\vec{1}}$, C, and Ω as in the protocol, and sets $\vec{y} = \vec{1}$. Then $\mathcal{S}$ runs the honest-verifier zero-knowledge simulators for both of the underlying protocols $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$. Since the underlying protocols $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ and $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ are both honest-verifier zero-knowledge, it is obvious that this simulation is indistinguishable from the transcript of real executions.

Finally, we focus on the soundness of the protocol. It remains to prove that the protocol has witness-extended emulation. The emulator $\mathcal{E}$ runs the protocol and if the transcript is accepted, $\mathcal{E}$ has to extract a witness.

$\mathcal{E}$ runs the witness-extended emulator for $\Pi_{\mathsf{zk}}^{\mathsf{Zero}}$ to get the extracted witness $(\{\vec{e}_i, r_i, \vec{d}_i, r_{\vec{d}_i}\}_{i\in[N]}, \vec{d}, r_{\vec{d}}, \vec{\tau})$, where $\vec{\tau}$ satisfies $\vec{g}^{\vec{\tau}} = \prod_{i=1}^N g_i^{-1}$. We claim that we have $\vec{d}_i = x^i\vec{e}_i$ and $r_{\vec{d}_i} = x^i r_i$. Otherwise, we have two opening for $c_{\vec{d}_i}$, which allows us to derive a nontrivial discrete logarithm relation, and this contradicts to the discrete logarithm relation assumption. Using the same argument, we have $\vec{d} = \sum_i^N \vec{d}_i$, $r_{\vec{d}} = \sum_{i=1}^N r_{\vec{d}_i}$, and $\vec{\tau} = -\vec{1}$. Hence, we have

$$\begin{aligned}
\sum_{i=1}^{N} x^i\vec{e}_i * \vec{e}_i - \vec{1} * (\sum_{i=1}^{N} x^i\vec{e}_i) = 0 &\iff \sum_{i=1}^{N} x^i(\vec{e}_i * \vec{e}_i - \vec{1} * \vec{e}_i) = 0\\
&\iff \sum_{i=1}^{N} x^i(\sum_{j=1}^{M} e_{ij}e_{ij}y^j - \sum_{k=1}^{M} e_{ik}y^k) = 0\\
&\iff \sum_{i=1}^{N} x^i(\sum_{j=1}^{M} y^j(e_{ij}e_{ij} - e_{ij})) = 0\,.
\end{aligned}$$

Since $x, y \leftarrow_{\$} \mathbb{Z}_q$, with an overwhelming probability, we have $e_{ij}e_{ij} = e_{ij}$ for $i \in [N], j \in [M]$, *i.e.*, $\vec{e}_i\vec{e}_i = \vec{e}_i$ for $i \in [N]$.

$\mathcal{E}$ also runs the witness-extended emulator for $\Pi_{\mathsf{zk}}^{\mathsf{Sum}}$ to get $(\vec{e}, r_{\vec{e}})$. Similarly, we can claim that $\vec{e} = \sum_{i=1}^{N} \omega^i \vec{e}_i$. Thus, we have

$$\vec{1}^{\mathrm{T}}\vec{e} = \Omega \iff \vec{1}^{\mathrm{T}}(\sum_{i=1}^{N} \omega^i \vec{e}_i) = \sum_{i=1}^{N} \omega^i \iff \sum_{i=1}^{N} \omega^i (\vec{1}^{\mathrm{T}}\vec{e}_i) = \sum_{i=1}^{N} \omega^i \, .$$

Since $\omega \leftarrow_{\$} \mathbb{Z}_q$, with an overwhelming probability, we have $\vec{1}^{\mathrm{T}}\vec{e}_i = 1$ for $i \in [N]$.

From $\{\vec{e}_i\}_{i\in[N]}$, we can derive the EP π. Therefore, we obtain the extracted witness $\{r_i\}_{i\in[N]}$ and π for the protocol, and the protocol has witness-extended emulation. □

E Proof of Theorem 4

Proof. The completeness of the protocol is clear. For the round that $\ell = 1$, the following equations are satisfied:

$$\begin{aligned}
\bar{g}^{z_1} u^{z_2} h^{z_3} &= \bar{g}^{x_1+\alpha\bar{e}} u^{x_2+\alpha\rho_{\vec{e}}} h^{x_3+\alpha r_{\vec{e}}} \\
&= (\bar{g}^{x_1} u^{x_2} h^{x_3})(\bar{g}^{\alpha\bar{e}} u^{\alpha\rho_{\vec{e}}} h^{\alpha r_{\vec{e}}}) \\
&= a_1 c_{\vec{e}}^{\alpha} \, ,
\end{aligned}$$

$$\begin{aligned}
\gamma^{z_1} u^{z_4} &= \gamma^{x_1+\alpha\bar{e}} u^{x_4+\alpha\rho'_{\vec{e}}} \\
&= (\gamma^{x_1} u^{x_4})(\gamma^{\alpha\bar{e}} u^{\alpha\rho'_{\vec{e}}}) \\
&= a_2 (c'_{\vec{e}})^{\alpha} \, ,
\end{aligned}$$

and

$$g^{z_3} = g^{x_3+\alpha r_{\vec{e}}} = g^{x_3} g^{\alpha r_{\vec{e}}} = a_3 (C^{(0)})^{\alpha} \, .$$

For the round that $\ell \neq 1$, the computed $c_{\vec{e}'}$, $\vec{e}'$, $\vec{g}'$, $\rho_{\vec{e}'}$, and $r_{\vec{e}}$ satisfy the following relation:

$$\begin{aligned}
(\vec{g}')^{\vec{e}'} u^{\rho_{\vec{e}'}} h^{r_{\vec{e}}} &= (\vec{g}_L^{\alpha^{-1}} \vec{g}_R^{\alpha})^{(\alpha\vec{e}_L+\alpha^{-1}\vec{e}_R)} u^{(\rho_{\vec{e}}+\alpha^2\rho_L+\alpha^{-2}\rho_R)} h^{r_{\vec{e}}} \\
&= (\vec{g}_L^{\alpha^{-1}} \vec{g}_R^{\alpha})^{(\alpha\vec{e}_L)} (\vec{g}_L^{\alpha^{-1}} \vec{g}_R^{\alpha})^{(\alpha^{-1}\vec{e}_R)} u^{(\rho_{\vec{e}}+\alpha^2\rho_L+\alpha^{-2}\rho_R)} h^{r_{\vec{e}}} \\
&= (\vec{g}_L^{\vec{e}_L} \vec{g}_R^{\alpha^2\vec{e}_L})(\vec{g}_L^{\alpha^{-2}\vec{e}_R} \vec{g}_R^{\vec{e}_R}) u^{(\rho_{\vec{e}}+\alpha^2\rho_L+\alpha^{-2}\rho_R)} h^{r_{\vec{e}}} \\
&= \vec{g}^{\vec{e}} \vec{g}_R^{\alpha^2\vec{e}_L} \vec{g}_L^{\alpha^{-2}\vec{e}_R} u^{(\rho_{\vec{e}}+\alpha^2\rho_L+\alpha^{-2}\rho_R)} h^{r_{\vec{e}}} \\
&= (\vec{g}^{\vec{e}} u^{\rho_{\vec{e}}} h^{r_{\vec{e}}})(\vec{g}_R^{\alpha^2\vec{e}_L} u^{\alpha^2\rho_L})(\vec{g}_L^{\alpha^{-2}\vec{e}_R} u^{\alpha^{-2}\rho_R}) \\
&= c_{\vec{e}} v_L^{\alpha^2} v_R^{\alpha^{-2}} \\
&= c_{\vec{e}'} \, .
\end{aligned}$$

Meanwhile, $c'_{\vec{e}}$, $\vec{e}'$, and $\vec{y}'$ satisfy the following equation:

$$\begin{aligned}
g^{(\vec{y}')^{\mathrm{T}}\vec{e}'}u^{\rho'_{\vec{e}'}} &= g^{(\alpha^{-1}\vec{y}_L+\alpha\vec{y}_R)^{\mathrm{T}}(\alpha\vec{e}_L+\alpha^{-1}\vec{e}_R)}u^{(\rho'_{\vec{e}}+\alpha^2\rho'_L+\alpha^{-2}\rho'_R)} \\
&= g^{\vec{y}_L^{\mathrm{T}}\vec{e}_L}g^{\alpha^2\vec{y}_R^{\mathrm{T}}\vec{e}_L}g^{\alpha^{-2}\vec{y}_L^{\mathrm{T}}\vec{e}_R}g^{\vec{y}_R^{\mathrm{T}}\vec{e}_R}u^{(\rho'_{\vec{e}}+\alpha^2\rho'_L+\alpha^{-2}\rho'_R)} \\
&= g^{\vec{y}^{\mathrm{T}}\vec{e}}g^{\alpha^2\vec{y}_R^{\mathrm{T}}\vec{e}_L}g^{\alpha^{-2}\vec{y}_L^{\mathrm{T}}\vec{e}_R}u^{(\rho'_{\vec{e}}+\alpha^2\rho'_L+\alpha^{-2}\rho'_R)} \\
&= (g^{\vec{y}^{\mathrm{T}}\vec{e}}u^{\rho'_{\vec{e}}})(g^{\alpha^2\vec{y}_R^{\mathrm{T}}\vec{e}_L}u^{\alpha^2\rho'_L})(g^{\alpha^{-2}\vec{y}_L^{\mathrm{T}}\vec{e}_R}u^{\alpha^{-2}\rho'_R}) \\
&= c'_{\vec{e}}(v'_L)^{\alpha^2}(v'_R)^{\alpha^{-2}} \\
&= c'_{\vec{e}'}\,.
\end{aligned}$$

Therefore, an honest prover P following the protocol could generate all the messages that pass the verification conducted by the verifier V.

For the honest-verifier zero-knowledge property, we construct a simulator $\mathcal{S}$. $\mathcal{S}$ firstly pick $u \leftarrow\!\!\$\, \mathbb{G}$.

When $\ell = 1$, the simulator $\mathcal{S}$ picks $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$ as the challenge. $\mathcal{S}$ also generates $z_1, z_2, z_3, z_4 \leftarrow\!\!\$\, \mathbb{Z}_q$. Then $\mathcal{S}$ computes $a_1 \leftarrow \bar{g}^{z_1}u^{z_2}h^{z_3}c_{\vec{e}}^{-\alpha}$, $a_2 \leftarrow \gamma^{z_1}u^{z_4}(c'_{\vec{e}})^{-\alpha}$, and $a_3 \leftarrow g^{z_3}(C^{(0)})^{-\alpha}$. It is obvious that the generated $(a_1, a_2, a_3, \alpha, z_1, z_2, z_3, z_4)$ is perfectly indistinguishable from the distribution of the real execution.

For the cases that $\ell \neq 1$, the simulator $\mathcal{S}$ firstly picks $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$ as the challenge. $\mathcal{S}$ then chooses $v_L, v_R, v'_L, v'_R \leftarrow\!\!\$\, \mathbb{G}$, and computes $c_{\vec{e}'}$, $c'_{\vec{e}'}$, $\vec{g}'$, $\vec{y}'$ as in the real execution. We note that since v_L, v_R, v'_L, v'_R can be regarded as Pedersen commitments that are perfectly hiding, the generated transcript is perfectly indistinguishable from the transcripts of real executions.

Hence, the simulator $\mathcal{S}$ produces a simulated proof that is indistinguishable from valid proofs generated by an honest prover interacting with an honest verifier.

Finally, we focus on the soundness of the protocol. It remains to prove that the protocol has witness-extended emulation. The emulator $\mathcal{E}$ runs the protocol, and if the transcript is accepted, $\mathcal{E}$ has to extract a witness. We will use an inductive argument to show that in each step, $\mathcal{E}$ can efficiently extract a witness. $\mathcal{E}$ first forks the execution with challenges u and u', such that $u \neq u'$. We then focus on the case with u.

When $\ell = 1$, after receiving a_1, a_2, and a_3, the emulator $\mathcal{E}$ obtains two accepting transcripts with two challenge α and α' such that $\alpha' \neq \alpha$ by rewinding the prover. From the two transcripts, $\mathcal{E}$ derives two pairs (z_1, z_2, z_3, z_4) and (z'_1, z'_2, z'_3, z'_4), such that

$$\bar{g}^{z_1}u^{z_2}h^{z_3} = a_1c_{\vec{e}}^{\alpha}\,, \quad \gamma^{z_1}u^{z_4} = a_2(c'_{\vec{e}})^{\alpha}\,, \quad g^{z_3} = a_3(C^{(0)})^{\alpha}\,,$$

and

$$\bar{g}^{z'_1}u^{z'_2}h^{z'_3} = a_1c_{\vec{e}}^{\alpha'}\,, \quad \gamma^{z'_1}u^{z'_4} = a_2(c'_{\vec{e}})^{\alpha'}\,, \quad g^{z'_3} = a_3(C^{(0)})^{\alpha'}\,.$$

Therefore, we can derive

$$g^{z_3-z'_3} = (C^0)^{\alpha-\alpha'}\,.$$

Let $r_{\vec{e}} = (z_3 - z'_3)/(\alpha' - \alpha)$, and then we have

$$g^{r_{\vec{e}}} = g^{(z_3-z'_3)/(\alpha'-\alpha)} = C^{(0)}\,.$$

Hence, we extract the discrete logarithm of $C^{(0)}$. Similarly, the emulator can compute

$$\bar{e} \leftarrow (z_1 - z_1')/(\alpha' - \alpha)\,, \quad \rho_{\vec{e}} \leftarrow (z_2 - z_2')/(\alpha' - \alpha)\,, \quad \rho'_{\vec{e}} \leftarrow (z_4 - z_4')/(\alpha' - \alpha)\,,$$

These extracted values are the corresponding discrete logarithms of $c_{\vec{e}}$, $c'_{\vec{e}}$, and $C^{(0)}$.

For the case that $\ell \neq 1$, $\mathcal{E}$ runs the prover and receives v_L, v_R, v'_L, and v'_R. Then $\mathcal{E}$ obtains three accepting transcripts with challenge α_i, such that $\alpha_i \neq \alpha_j$ for $1 \leq i < j \leq 3$ by rewinding the prover. From the three transcripts, $\mathcal{E}$ derives pairs $(\vec{e}'_i, \rho_{\vec{e}'_i}, \rho'_{\vec{e}''_i})$ for $i = 1, 2, 3$, such that

$$c_{\vec{e}} v_L^{\alpha_i^2} v_R^{\alpha_i^{-2}} = (\vec{g}_L^{\alpha_i^{-1}} \vec{g}_R^{\alpha_i})^{\vec{e}'_i} u^{\rho_{\vec{e}'_i}} h^{r_{\vec{e}}} \tag{1}$$

$$c'_{\vec{e}} (v'_L)^{\alpha_i^2} (v'_R)^{\alpha_i^{-2}} = g^{(\alpha_i^{-1} \vec{y}_L + \alpha_i \vec{y}_R)^{\mathrm{T}} \vec{e}'_i} u^{\rho'_{\vec{e}_i}} \tag{2}$$

We can easily find ν_1, ν_2, ν_3, such that

$$\sum_{i=1}^{3} \nu_i \alpha_i^2 = 0\,, \quad \sum_{i=1}^{3} \nu_i = 1\,, \quad \sum_{i=1}^{3} \nu_i \alpha_i^{-2} = 0\,.$$

This follows from the fact that the matrix below is full rank:

$$\begin{bmatrix} 1 & \alpha_1^2 & \alpha_1^{-2} \\ 1 & \alpha_2^2 & \alpha_2^{-2} \\ 1 & \alpha_3^2 & \alpha_3^{-2} \end{bmatrix}.$$

Then we take the linear combination (to the power) of the three equalities (for $i = 1, \ldots, 3$) in (1) with ν_1, ν_2, ν_3 as the coefficients and obtain

$$\begin{aligned} c_{\vec{e}} &= \prod_{i=1}^{3} (c_{\vec{e}} v_L^{\alpha_i^2} v_R^{\alpha_i^{-2}})^{\nu_i} \\ &= (\prod_{i=1}^{3} ((\vec{g}_L^{\alpha_i^{-1}} \vec{g}_R^{\alpha_i})^{\vec{e}'_i})^{\nu_i}) u^{\sum_{i=1}^{3} \nu_i \rho_{\vec{e}'_i}} h^{r_{\vec{e}} \sum_{i=1}^{3} \nu_i} \\ &= (\vec{g}_L^{\sum_{i=1}^{3} \nu_i \alpha_i^{-1} \vec{e}'_i} \vec{g}_R^{\sum_{i=1}^{3} \nu_i \alpha_i \vec{e}'_i}) u^{\sum_{i=1}^{3} \nu_i \rho_{\vec{e}'_i}} h^{r_{\vec{e}}}\,. \end{aligned}$$

We can compute

$$\vec{e} \leftarrow (\sum_{i=1}^{3} \nu_i \alpha_i^{-1} \vec{e}'_i, \sum_{i=1}^{3} \nu_i \alpha_i \vec{e}'_i) \in \mathbb{Z}_q^n\,, \quad \rho_{\vec{e}} \leftarrow \sum_{i=1}^{3} \nu_i \rho_{\vec{e}'_i} \in \mathbb{Z}_q\,,$$

such that $c_{\vec{e}} = \vec{g}^{\vec{e}} u^{\rho_L} h^{r_{\vec{e}}}$. Similarly, we can repeat this process for (2) and obtain

$$\begin{aligned} c'_{\vec{e}} &= \prod_{i=1}^{3} (c'_{\vec{e}} (v'_L)^{\alpha_i^2} (v'_R)^{\alpha_i^{-2}})^{\nu_i} \\ &= g^{\sum_{i=1}^{3} \nu_i (\alpha_i^{-1} \vec{y}_L + \alpha_i \vec{y}_R)^{\mathrm{T}} \vec{e}'_i} u^{\sum_{i=1}^{3} \nu_i \rho'_{\vec{e}_i}} \\ &= g^{\vec{y}_L^{\mathrm{T}} (\sum_{i=1}^{3} \nu_i \alpha_i^{-1} \vec{e}'_i) + \vec{y}_R^{\mathrm{T}} (\sum_{i=1}^{3} \nu_i \alpha_i \vec{e}'_i)} u^{\sum_{i=1}^{3} \nu_i \rho'_{\vec{e}_i}}\,. \end{aligned}$$

We here derive the same $\vec{e} = (\sum_{i=1}^{3} \nu_i \alpha_i^{-1} \vec{e}_i', \sum_{i=1}^{3} \nu_i \alpha_i \vec{e}_i') \in \mathbb{Z}_q^n$ and can compute $\rho_{\vec{e}}' = \sum_{i=1}^{3} \nu_i \rho_{\vec{e}_i}' \in \mathbb{Z}_q$, such that $c_{\vec{e}}' = g^{\vec{y}^{\mathrm{T}} \vec{e}} u^{\rho_{\vec{e}}'}$. Thus, the emulator $\mathcal{E}$ extracts the witness $\vec{e}$, $\rho_{\vec{e}}$, and $\rho_{\vec{e}}'$ for this round.

Following this procedure, the emulator $\mathcal{E}$ can go from the leaves of the transcript tree to the root of that tree and finally extract the witness of original relation. Note that the extracted witness for the first statement should satisfy that $\vec{y}^{\mathrm{T}} \vec{e} = \Omega$ and $\rho_{\vec{e}}' = 0$. If it is not, since we have

$$c_{\vec{e}}' = g^{\Omega} = g^{\vec{y}^{\mathrm{T}} \vec{e}} u^{\rho_{\vec{e}}'} ,$$

we can easily compute a nontrivial discrete logarithm relation, and this contradicts to the discrete logarithm relation assumption. Similarly, we should have $\rho_{\vec{e}} = 0$ and $c_{\vec{e}} = \vec{g}^{\vec{e}} h^{r_{\vec{e}}}$. If $\rho_{\vec{e}} \neq 0$, let us consider the other forking flow with u'. Denote the corresponding extracted witness in this flow by $(\hat{\vec{e}}, r_{\vec{e}}, \hat{\rho}_{\vec{e}})$. Here we have identical $r_{\vec{e}}$ because $r_{\vec{e}}$ is fixed for $C^{(0)}$. Since $u \neq u'$ and $\rho_{\vec{e}} \neq 0$, we have $(\vec{e}, \rho_{\vec{e}}) \neq (\hat{\vec{e}}, \hat{\rho}_{\vec{e}})$. Hence, we have

$$c_{\vec{e}} = \vec{g}^{\vec{e}} u^{\rho_{\vec{e}}} h^{r_{\vec{r}}} = \vec{g}^{\hat{\vec{e}}} (u')^{\hat{\rho}_{\vec{e}}} h^{r_{\vec{r}}} ,$$

from which we can easily compute a nontrivial discrete logarithm relation, and this contradicts to the discrete logarithm relation assumption. Thus, the emulator $\mathcal{E}$ successfully extracts the witness.

During the extraction, $\mathcal{E}$ uses $4 \times 3^{\log_2(M)}$ transcripts, and thus runs in expected polynomial time in M.[5] Therefore, the protocol has witness-extended emulation. □

F Proof of Theorem 5

Proof. For completeness, if the statement is valid, the equation below holds:

$$d_{\ell+1} = \sum_{\substack{0 \leq i \leq \ell, 1 \leq j \leq \ell+1 \\ j=i}} \vec{u}_i * \vec{v}_j = \sum_{i=1}^{\ell} \vec{u}_i * \vec{v}_i = 0 .$$

Hence, $c_{d_\ell+1} = g^{d_{\ell+1}} h^{r_{d_{\ell+1}}} = 1$ given $r_{d_{\ell+1}} = 0$. Meanwhile, we have

$$\vec{u} * \vec{v} = (\sum_{i=0}^{\ell} x^i \vec{u}_i) * (\sum_{j=1}^{\ell+1} x^{\ell-j+1} \vec{v}_j) = \sum_{k=0}^{2\ell} x^k d_k .$$

Hence, the perfect completeness of the protocol directly follows from the verification conducted by V.

For the honest-verifier zero-knowledge property, we construct a simulator $\mathcal{S}$ as follows.

[5] More information for the analysis of the expected running time could be found in [10] or [39, Section 13.1.3].

For the challenge $x \leftarrow_{\$} \mathbb{Z}_q$, $\mathcal{S}$ picks $r_{\vec{u}}, r_{\vec{v}}, r_{d_0}, \ldots, r_{d_\ell}, r_{d_{\ell+2}}, \ldots, r_{d_{2\ell}} \leftarrow_{\$} \mathbb{Z}_q$ and $\vec{u}, \vec{v} \leftarrow_{\$} \mathbb{Z}_q^M$. Then $\mathcal{S}$ sets $r_{d_{\ell+1}} = 0$ and computes

$$c_{\vec{u}_0}^{(0)} \leftarrow g^{r_{\vec{u}}} \prod_{i=1}^{\ell} (c_{\vec{u}_i}^{(0)})^{-x^i}, \; c_{\vec{u}_0}^{(1)} \leftarrow \vec{g}^{\vec{u}} h^{r_{\vec{u}}} \prod_{i=1}^{\ell} (c_{\vec{u}_i}^{(1)})^{-x^i}, \; t \leftarrow \sum_{\phi=0}^{2\ell} x^{\phi} r_{d_\phi},$$

$$c_{\vec{v}_{\ell+1}}^{(0)} \leftarrow g^{r_{\vec{v}}} \prod_{j=1}^{\ell} (c_{\vec{v}_j}^{(0)})^{-x^{\ell+1-j}}, \; c_{\vec{v}_{\ell+1}}^{(1)} \leftarrow \vec{g}^{\vec{v}} h^{r_{\vec{v}}} \prod_{j=1}^{\ell} (c_{\vec{v}_j}^{(1)})^{-x^{\ell+1-j}}, \; c_{d_0} \leftarrow g^{\vec{u} * \vec{v}} h^{r_{d_0}},$$

and $c_{d_\phi} \leftarrow h^{r_{d_\phi}}$ for $\phi \in \{1, \ldots, 2\ell\} \backslash \{\ell + 1\}$. The simulated transcript is

$$(c_{\vec{u}_0}^{(0)}, c_{\vec{u}_0}^{(1)}, c_{\vec{u}_{\ell+1}}^{(0)}, c_{\vec{u}_{\ell+1}}^{(1)}, \{c_{d_\phi}\}_{\phi=0,\ldots,2\ell}, x, \vec{u}, \vec{v}, r_{\vec{u}}, r_{\vec{v}}, t).$$

Note that this simulated transcript is perfectly indistinguishable from the transcripts of real executions. This is due to the fact that elements of $\{c_{d_\phi}\}_{\phi=1,\ldots,2\ell}$ are all perfectly hiding commitments, and $\vec{u}, \vec{v}, r_{\vec{u}}, r_{\vec{v}}, t$ are uniformly random both in the real protocol and the simulation. In addition, $c_{\vec{u}_0}^{(0)}, c_{\vec{u}_0}^{(1)}, c_{\vec{u}_{\ell+1}}^{(0)}, c_{\vec{u}_{\ell+1}}^{(1)}, c_{d_0}$ are all uniquely determined by the verification equations. Hence, the honest-verifier zero-knowledge property follows.

Finally, we focus on the soundness of the protocol. It remains to prove that the protocol has witness-extended emulation. The emulator $\mathcal{E}$ runs the protocol, and if the transcript is accepted, $\mathcal{E}$ has to extract a witness. After receiving $c_{\vec{u}_0}$, $c_{\vec{v}_{\ell+1}}$, and $\{c_{d_\phi}\}_{\phi \in \{0,\ldots,2\ell\} \backslash \{\ell+1\}}$, $\mathcal{E}$ obtains $2\ell+1$ accepting transcripts with different challenges $\{x_i\}_{i \in [2\ell+1]}$ by rewinding the prover. On average $\mathcal{E}$ will be making $2\ell + 1$ arguments, and thus it runs in expected polynomial time. Now we have for $k \in [2\ell + 1]$

$$\prod_{i=0}^{\ell} (c_{\vec{u}_i}^{(0)})^{x_k^i} = g^{r_{\vec{u}}^{\langle k \rangle}}, \quad \prod_{i=0}^{\ell} (c_{\vec{u}_i}^{(1)})^{x_k^i} = \vec{g}^{\vec{u}^{\langle k \rangle}} h^{r_{\vec{u}}^{\langle k \rangle}}, \quad c_{d_{\ell+1}} = 1,$$
$$\prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(0)})^{x_k^{\ell+1-j}} = g^{r_{\vec{v}}^{\langle k \rangle}}, \; \prod_{j=1}^{\ell+1} (c_{\vec{v}_j}^{(1)})^{x_k^{\ell+1-j}} = \vec{g}^{\vec{v}^{\langle k \rangle}} h^{r_{\vec{v}}^{\langle k \rangle}}, \; \prod_{\phi=0}^{2\ell} c_{d_\phi}^{x_k^\phi} = g^{\vec{u}^{\langle k \rangle} * \vec{v}^{\langle k \rangle}} h^{t^{\langle k \rangle}}.$$

We can easily solve the discrete logarithms $\{r_{\vec{u}_i}\}_{i=0,\ldots,\ell}$ via a system of equations from arbitrary $\ell + 1$ accepting transcripts, such as:

$$r_{\vec{u}_0} x_1^0 + \cdots + r_{\vec{u}_\ell} x_1^\ell = r_{\vec{u}}^{\langle 1 \rangle}$$
$$\vdots$$
$$r_{\vec{u}_0} x_{\ell+1}^0 + \cdots + r_{\vec{u}_\ell} x_{\ell+1}^\ell = r_{\vec{u}}^{\langle \ell+1 \rangle}$$

when $\{x_k\}_{k \in [\ell+1]}$ and $\{r_{\vec{u}}^{\langle k \rangle}\}_{k \in [\ell+1]}$ are known. We can always solve this system of equations since the corresponding Vandermonde matrix of x_k's has full rank. Then given extracted $\{r_{\vec{u}_i}\}_{i=0,\ldots,\ell}$, we can extract $\{\vec{u}_i\}_{i=0,\ldots,\ell}$ from the following system of equations via the same approach.

$$x_1^0\vec{u}_0 + \cdots + x_1^\ell\vec{u}_\ell = \vec{u}^{\langle 1\rangle}$$
$$\vdots$$
$$x_{\ell+1}^0\vec{u}_0 + \cdots + x_{\ell+1}^\ell\vec{u}_\ell = \vec{u}^{\langle \ell+1\rangle}$$

Similarly, we can extract $\{\vec{v}_j\}_{j=1,\ldots,\ell+1}$ and $\{r_{\vec{v}_j}\}_{j=1,\ldots,\ell+1}$. Meanwhile, we can extract $\{d_\phi\}_{\phi=0,\ldots,2\ell}$ and $\{r_{d_\phi}\}_{\phi=0,\ldots,2\ell}$ from the systems of equations

$$d_0x_1^0 + \cdots + d_{2\ell}x_1^{2\ell} = D_1$$
$$\vdots$$
$$d_0x_{2\ell+1}^0 + \cdots + d_{2\ell}x_{2\ell+1}^{2\ell} = D_{2\ell+1}$$

and

$$r_{d_0}x_1^0 + \cdots + r_{d_{2\ell}}x_1^{2\ell} = t^{\langle 1\rangle}$$
$$\vdots$$
$$r_{d_0}x_{2\ell+1}^0 + \cdots + r_{d_{2\ell}}x_{2\ell+1}^{2\ell} = t^{\langle 2\ell+1\rangle}$$

where $D_k \leftarrow \vec{u}^{\langle k\rangle} * \vec{v}^{\langle k\rangle}$. We claim that the extracted $d_{\ell+1}$ and $r_{d_{\ell+1}}$ satisfying $d_{\ell+1} = 0$ and $r_{d_{\ell+1}} = 0$. Otherwise, we have

$$1 = g^{d_{\ell+1}}h^{r_{\ell+1}} = g^0h^0\,,$$

from which we can easily compute a nontrivial discrete logarithm relation, and this contradicts to the discrete logarithm relation assumption. Thus, the emulator $\mathcal{E}$ successfully extracts the witness such that $\sum_{i=1}^{\ell}\vec{u}_i * \vec{v}_i = 0$, and the protocol has witness-extended emulation. □

G Proof of Theorem 6

Proof. We first focus on the case that $\mathsf{P_A}$ is malicious. The analysis for malicious $\mathsf{P_A}$ is very similar to the proof of Theorem 1. For an adversary $\mathcal{A}$ corrupting $\mathsf{P_A}$ in the real world, we construct a simulator $\mathcal{S}$ holding $(\mathsf{vk}, \mathsf{sigk})$ that runs $\mathcal{A}$ as a subroutine and plays the role of $\mathsf{P_B}$ in the ideal world. $\mathcal{S}$ first picks the common reference string $g_i \leftarrow_\$ \mathbb{G}$ for $i \in [N]$, where all g_i's are different. Now we present the simulation procedure for the initiation phase and evaluation phase (denoted by $\mathbf{Game}_0$). The simulator simulates the initiation phase as follows.

1. $\mathcal{S}$ receives h, Φ, Φ', and $\{d_i\}_{i\in[N]}$ from $\mathcal{A}$. Then $\mathcal{S}$ receives the EP π_f (and corresponding random coins) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$. $\mathcal{S}$ verifies whether π_f and the corresponding random coins are correct. If not, $\mathcal{S}$ sends abort_A to $\mathcal{F}_{\mathsf{activePFE}}$ and simulates the termination of $\mathsf{P_B}$. $\mathcal{S}$ also receives s and $\{t_i\}_{i\in[N]}$ from $\mathcal{A}$ for $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ and verifies them following a similar procedure as for $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$.

2. $\mathcal{S}$ computes $P = [p_1, \ldots, p_N]$ as in the protocol.

$\mathcal{S}$ can derive the evaluated circuit C_f from the EP π_f. Then $\mathcal{S}$ sends C_f to $\mathcal{F}_{\mathsf{activePFE}}$ and proceeds to simulate the evaluation phase.

0. Upon receiving $\{\mathsf{c}^{\mathsf{seed}_j^A}\}_{j\in[\lambda]}$ from $\mathcal{A}$, $\mathcal{S}$, as in the protocol, picks uniform κ-bit strings $\{\mathsf{seed}_j^B, \mathsf{witness}_j\}_{j\in[\lambda]}$. Then $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} to extract $\mathcal{A}$'s inputs $\{b_j\}_{j\in[\lambda]}$. Let sets $J_{\mathsf{seed}} = \{j : b_j = 0\}$ and $J_{\mathsf{witness}} = \{j : b_j = 1\}$. Let $\mathcal{S}_{\mathsf{OT}}$ return seed_j^B for $j \in J_{\mathsf{seed}}$ and $\mathsf{witness}_j$ for $j \in J_{\mathsf{witness}}$ to $\mathcal{A}$.
1. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, we let $\hat{\jmath}$ be the unique index in J_{witness}. $\mathcal{S}$ chooses $\alpha_i \leftarrow\$\, \mathbb{Z}_q$ for $i \in [M]$. $\mathcal{S}$ computes $w_i \leftarrow g_i^{\alpha_i}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w_i \leftarrow\$\, \mathbb{G}$ for $i = M+1, \ldots, M+m$ for output-wire labels of output gates. $\mathcal{S}$ also computes $v_i \leftarrow (w_{\pi_f(i)})^{t_i}$ and picks $v_i' \leftarrow\$\, \mathbb{G}$ for $i \in [N]$.
     $\mathcal{S}$ computes $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v_{2i-1}, v_{2i-1}'), (v_{2i}, v_{2i}'), (w_{n+i}, w_{n+i})\}_{i\in[\theta]})$.
     Let $\mathsf{GC}_{\hat{\jmath}} = \{\mathsf{GG}_i\}_{i\in[\theta]}$.
   - If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_A}$ but uses true randomness in this step.
2. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} to extract $\mathcal{A}$'s input x_A. $\mathcal{S}_{\mathsf{OT}}$ returns $\{w_i\}_{i\in[n_A]}$ to $\mathcal{A}$. $\mathcal{S}$ sends x_A to the ideal functionality $\mathcal{F}_{\mathsf{covertPFE}}$, and receives the evaluation result $y \in \{0,1\}^m$ or $\mathsf{nothing}$ depending on the scenario.
   - If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_A}$ but uses true randomness in this step.
3. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, $\mathcal{S}$ computes $\mathsf{c}_{i,\hat{\jmath},0}^{x_B} \leftarrow \mathsf{Com}(w_{n_A+i})$ and $\mathsf{c}_{i,\hat{\jmath},1}^{x_B} \leftarrow \mathsf{Com}(0)$ for $i \in [n_B]$.
     If y is known, we let $\mathtt{h}_j^{\mathsf{O}}$ denotes the hash value of the output-wire labels $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]}$, where $w_{M+i}^{y[i]} = w_{M+i}$ and $w_{M+i}^{1-y[i]} \leftarrow\$\, \mathbb{G}$. Otherwise, we let $\mathtt{h}_j^{\mathsf{O}}$ denotes the hash value of the output-wire labels $\{(w_{M+i}, w_{M+i}')\}_{i\in[m]}$, where $w_{M+i}' \leftarrow\$\, \mathbb{G}$.
     $\mathcal{S}$ then computes $\mathsf{c}_{\hat{\jmath}} \leftarrow \mathsf{Com}(\mathsf{GC}_{\hat{\jmath}}, \{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}, \mathsf{h}_{\hat{\jmath}}^{\mathsf{O}})$, where two elements in each pair $(\mathsf{c}_{i,\hat{\jmath},0}^{x_B}, \mathsf{c}_{i,\hat{\jmath},1}^{x_B})$ are permuted in random order.
   - If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ except for computing c_j using true randomness in this step.

   Then $\mathcal{S}$ computes signature σ_j's as in the protocol, and sends $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.
4. If $|J_{\mathsf{witness}}| \neq 1$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ receives $(\hat{\jmath}, \{\mathsf{seed}_j^B\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ from $\mathcal{A}$. $\mathcal{S}$ verifies that these values are all consistent with those that have been sent and aborts if not.

5. $\mathcal{S}$ sends $\mathsf{GC}_{\hat{\jmath}}$, $\{w_{n_A+i}\}_{i\in[n_B]}$, $\{\mathsf{c}^{x_B}_{i,\hat{\jmath},b}\}_{i\in[n_B],b\in\{0,1\}}$ (in the same order as before), and $\mathsf{h}^{\mathsf{O}}_{\hat{\jmath}}$, together with $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$ and $\{\mathsf{decom}^{\mathsf{c}^{x_B}_{i,\hat{\jmath},0}}\}_{i\in[n_B]}$, to $\mathcal{A}$.
   - If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also sends the mapping $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$ to $\mathcal{A}$. Then $\mathcal{S}$ outputs what $\mathcal{A}$ outputs to conclude the simulation.
   - If $\mathsf{P_A}$ is not allowed to know the evaluation result, $\mathcal{S}$ continues to the next step.
6. $\mathcal{S}$ receives from $\mathcal{A}$ the output-wire labels $\{\tilde{w}_{M+i}\}_{i\in[m]}$. If all elements of $\{\tilde{w}_{M+i}\}_{i\in[m]}$ are consistent with those of $\{w_{M+i}\}_{i\in[m]}$, $\mathcal{S}$ sends $\mathsf{continue}$ to $\mathcal{F}_{\mathsf{covertPFE}}$. Otherwise, $\mathcal{S}$ sends abort_A to $\mathcal{F}_{\mathsf{covertPFE}}$.

It remains to show that the joint distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ and the output of $\mathsf{P_B}$ in the ideal world is indistinguishable from the joint distribution of the view of $\mathcal{A}$ and the output of $\mathsf{P_B}$ in the real world. We define the following games and let the output of each game be the view of $\mathcal{A}$ and output of $\mathsf{P_B}$.

Game$_1$ We modify the evaluation phase of the previous game as follows.

0. Upon receiving $\{\mathsf{c}^{\mathsf{seed}^A_j}\}_{j\in[\lambda]}$ from $\mathcal{A}$, κ-bit strings $\{\mathsf{seed}^B_j, \mathsf{witness}_j\}_{j\in[\lambda]}$ are picked as in the protocol. Then $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} to extract $\mathcal{A}$'s inputs $\{b_j\}_{j\in[\lambda]}$. Let sets $J_{\mathsf{seed}} = \{j : b_j = 0\}$ and $J_{\mathsf{witness}} = \{j : b_j = 1\}$. Let $\mathcal{S}_{\mathsf{OT}}$ return seed^B_j for $j \in J_{\mathsf{seed}}$ and $\mathsf{witness}_j$ for $j \in J_{\mathsf{witness}}$ to $\mathcal{A}$.
1. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, we let $\hat{\jmath}$ be the unique index in J_{witness}. $\mathcal{S}$ chooses $\alpha_0, \alpha_1 \leftarrow\!\!\$\, \mathbb{Z}_q$ for $i \in [M]$. $\mathcal{S}$ computes $w^0_i \leftarrow g_i^{\alpha_0}$ and $w^1_i \leftarrow g_i^{\alpha_1}$ for $i \in [M]$. Then $\mathcal{S}$ picks $w^0_i, w^1_i \leftarrow\!\!\$\, \mathbb{G}$ for $i = M+1, \ldots, M+m$ for output-wire labels of output gates. $\mathcal{S}$ also computes $v^0_i \leftarrow (p_i)^{\alpha_0}$ and $v^1_i \leftarrow (p_i)^{\alpha_1}$ for $i \in [N]$.
     $\mathcal{S}$ computes $\{\mathsf{GG}_i\}_{i\in[\theta]} \leftarrow \mathsf{Gb}(\{i, (v^0_{2i-1}, v^1_{2i-1}), (v^0_{2i}, v^1_{2i}), (w^0_{n+i}, w^1_{n+i})\}_{i\in[\theta]})$.
     Let $\mathsf{GC}_{\hat{\jmath}} = \{\mathsf{GG}_i\}_{i\in[\theta]}$.
   - If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_A}$ but uses true randomness in this step.
2. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} to extract $\mathcal{A}$'s input x_A. $\mathcal{S}_{\mathsf{OT}}$ returns $\{w_i^{x_A[i]}\}_{i\in[n_A]}$ to $\mathcal{A}$.
   - If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_A}$ but uses true randomness in this step.
3. For $j \in J_{\mathsf{seed}}$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ and follows the protocol to run this step. For $j \in J_{\mathsf{witness}}$, $\mathcal{S}$ does the following.
   - If $|J_{\mathsf{witness}}| = 1$, $\mathcal{S}$ computes $\mathsf{c}^{x_B}_{i,\hat{\jmath},0} \leftarrow \mathsf{Com}(w^0_{n_A+i})$ and $\mathsf{c}^{x_B}_{i,\hat{\jmath},1} \leftarrow \mathsf{Com}(w^1_{n_A+i})$ for $i \in [n_B]$. Let $\mathtt{h}^{\mathsf{O}}_{\hat{\jmath}}$ denotes the hash value of the output-wire labels $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$.
     $\mathcal{S}$ computes $\mathsf{c}_{\hat{\jmath}} \leftarrow \mathsf{Com}(\mathsf{GC}_{\hat{\jmath}}, \{\mathsf{c}^{x_B}_{i,\hat{\jmath},b}\}_{i\in[n_B],b\in\{0,1\}}, \mathsf{h}^{\mathsf{O}}_{\hat{\jmath}})$, where two elements in each pair $(\mathsf{c}^{x_B}_{i,\hat{\jmath},0}, \mathsf{c}^{x_B}_{i,\hat{\jmath},1})$ are permuted in random order.

– If $|J_{\mathsf{witness}}| \geq 2$, $\mathcal{S}$ acts as an honest $\mathsf{P_B}$ except for computing c_j using true randomness in this step.

Then $\mathcal{S}$ generates signatures σ_j's as in the protocol, and sends $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

4. If $|J_{\mathsf{witness}}| \neq 1$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ receives $(\hat{\jmath}, \{\mathsf{seed}^B_j\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ from $\mathcal{A}$. $\mathcal{S}$ verifies that these messages are all consistent with those that have been sent and aborts if not.
5. $\mathcal{S}$ sends $\mathsf{GC}_{\hat{\jmath}}$, $\{w^{x_B[i]}_{n_A+i}\}_{i\in[n_B]}$, $\{\mathsf{c}^{x_B}_{i,\hat{\jmath},b}\}_{i\in[n_B],b\in\{0,1\}}$ (in the same order as before), and $\mathsf{h}^{\mathsf{O}}_{\hat{\jmath}}$, together with $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$ and $\{\mathsf{decom}^{\mathsf{c}^{x_B}_{i,\hat{\jmath},x_B[i]}}\}_{i\in[n_B]}$, to $\mathcal{A}$.
   – If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also sends the mapping $\{(w^0_{M+i}, w^1_{M+i})\}_{i\in[m]}$ to $\mathcal{A}$.
   – If $\mathsf{P_A}$ is not allowed to know the evaluation result, $\mathcal{S}$ continues to the next step.
6. $\mathcal{S}$ receives from $\mathcal{A}$ the output-wire labels $\{\tilde{w}_{M+i}\}_{i\in[m]}$. Then $\mathcal{S}$ verify whether all elements of $\{\tilde{w}_{M+i}\}_{i\in[m]}$ are consistent with those of $\{w^0_{M+i}, w^1_{M+i}\}_{i\in[m]}$ as in the protocol.

Based on the analysis in the proof of Theorem 1, the security of the hash function, and the hiding property of the commitment scheme, we can easily see that the output of $\mathbf{Game}_1$ is computationally indistinguishable from the output of $\mathbf{Game}_0$.

$\mathbf{Game}_2$ In this game, Π_{OT} in Step 2 of the evaluation phase is executed honestly when $|J_{\mathsf{witness}}| = 1$. It follows from the security of Π_{OT} that the output of $\mathbf{Game}_2$ is computationally indistinguishable from the output of $\mathbf{Game}_1$.

$\mathbf{Game}_3$ Steps 1–3 of the previous game is modified, such that random coins for $j \in J_{\mathsf{witness}}$ are derived from seed^B_j instead of using true randomness. It is obvious that the output of $\mathbf{Game}_3$ is computationally indistinguishable from the output of $\mathbf{Game}_2$.

$\mathbf{Game}_4$ We modify the previous game as follows. In Step 4, $\mathcal{S}$ continues to run the protocol as an honest $\mathsf{P_B}$ even when $|\mathsf{witness}| \neq 1$. Due to the security of Π_{OT}, it is easy to see that the output of $\mathbf{Game}_4$ is computationally indistinguishable from the output of $\mathbf{Game}_3$.

$\mathbf{Game}_5$ In the game, the executions of Π_{OT} in Step 0 are executed honestly. According to the security of Π_{OT}, the output of $\mathbf{Game}_5$ is computationally indistinguishable from the output of $\mathbf{Game}_4$.

Note that $\mathbf{Game}_5$ corresponds to the real execution of the protocol where $\mathsf{P_B}$ holds input x_B and interacts with $\mathsf{P_A}$, while $\mathbf{Game}_0$ corresponds to the simulated execution in the ideal world. Hence, we complete the proof for malicious $\mathsf{P_A}$.

We now focus on the case that $\mathsf{P_B}$ is malicious (in a covert sense). For an adversary $\mathcal{A}$ corrupting $\mathsf{P_B}$ in the real world, we construct a simulator $\mathcal{S}$ holding vk that runs $\mathcal{A}$ as a subroutine and plays the role of $\mathsf{P_A}$ in the ideal world. $\mathcal{S}$ first computes the common reference string $g_i = g^{\omega_i}$, where $\omega_i \leftarrow_{\$} \mathbb{Z}_q$, for $i \in [N]$. It is easy to see that this common reference string has an identical distribution to that in the real world. Now we present the simulation procedures for the initiation phase and evaluation phase (denoted by $\mathbf{Game}_0$). The simulator $\mathcal{S}$ simulates the initiation phase as follows.

1. $\mathcal{S}$ picks $h \leftarrow\!\!\$\, \mathbb{G}$. Then $\mathcal{S}$ generates $c_i \leftarrow\!\!\$\, \mathbb{G}^2$, ${c'_i}^{(0)} \leftarrow\!\!\$\, \mathbb{G}$, ${c'_i}^{(1)} \leftarrow g^{\rho'_i}$ and $d_i \leftarrow g^{\rho''_i}$, where $\rho'_i \leftarrow\!\!\$\, \mathbb{Z}_q$ and $\rho''_i \leftarrow\!\!\$\, \mathbb{Z}_q$, for $i \in [N]$. Let $\rho_i \leftarrow \rho'_i - \rho''_i \bmod q$. Note that now we have $p_i = g^{\rho_i}$. Then $\mathcal{S}$ acts as $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ to convince $\mathcal{A}$.
2. $\mathcal{S}$ does nothing.

Then in the evaluation phase, $\mathcal{S}$ follows the simulation procedure below.

0. $\mathcal{S}$ chooses uniform κ-bit strings $\{\mathsf{seed}_j^A\}_{j\in[\lambda]}$, computes $\mathsf{c}^{\mathsf{seed}_j^A}$'s as in the protocol, and sends $\{\mathsf{c}^{\mathsf{seed}_j^A}\}_{j\in[\lambda]}$ to $\mathcal{A}$. For all λ execution of Π_{OT}, $\mathcal{S}$ interacts with $\mathcal{A}$ using the input 0 with randomness derived from seed_j^A, and retrieves $\{\mathsf{seed}_j^B\}_{j\in[\lambda]}$ at the end. Let us denote the transcript of the jth execution by trans_j.
1. $\mathcal{S}$ does nothing.
2. $\mathcal{S}$ uses as input 0^{n_A} for all execution of Π_{OT} with randomness derived from seed_j^A. Let $\mathtt{h}_j^{\mathsf{OT}}$ denote the hash value of the transcript for the jth execution of Π_{OT}.
3. $\mathcal{S}$ receives $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ from $\mathcal{A}$.
4. If any signature σ_j are invalid, $\mathcal{S}$ sends abort_B to $\mathcal{F}_{\mathsf{covertPFE}}$ and simulates the termination of $\mathsf{P_A}$. For $j \in [\lambda]$, $\mathcal{S}$ simulates $\mathsf{P_B}$'s execution in Steps 1, 2, and 3a, and particularly computes $\hat{\mathsf{h}}_j^{\mathsf{OT}}$ and $\hat{\mathsf{c}}_j$. Let J be the set of indices, such that $(\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j) \neq (\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j)$.
   - If $|J| = 0$, $\mathcal{S}$ sets $\mathsf{caught} = \mathsf{nothing}$ and continues below.
   - If $|J| = 1$, $\mathcal{S}$ sends cheat to $\mathcal{F}_{\mathsf{covertPFE}}$. If $\mathsf{corrupted}$ is received, $\mathcal{S}$ sets $\mathsf{caught} = \mathsf{true}$. Otherwise if $(\mathsf{undetected}, C_f, x_A)$ is received, $\mathcal{S}$ sets $\mathsf{caught} = \mathsf{false}$. Then $\mathcal{S}$ continues below.
   - If $|J| \geq 2$, $\mathcal{S}$ sends $\mathsf{blatantCheat}$ to $\mathcal{F}_{\mathsf{covertPFE}}$, sends the certificate $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$ to $\mathcal{A}$ for uniform $j \in J$, and simulates the termination of $\mathsf{P_A}$.

Then $\mathcal{S}$ rewinds $\mathcal{A}$ and runs Steps $0' - 4'$ below until[6] $|J'| = |J|$ and $\mathsf{caught}' = \mathsf{caught}$:

0′. $\mathcal{S}$ picks $\hat{\jmath} \leftarrow\!\!\$\, [\lambda]$ and computes $\mathsf{c}^{\mathsf{seed}_{\hat{\jmath}}^A} \leftarrow \mathsf{Com}(0^\kappa)$. $\mathcal{S}$ chooses a uniform κ-bit string seed_j^A, computes $\mathsf{c}^{\mathsf{seed}_j^A}$ as in the protocol for $j \neq \hat{\jmath}$, and sends $\{\mathsf{c}^{\mathsf{seed}_j^A}\}_{j\in[\lambda]}$ to $\mathcal{A}$. For the jth ($j \neq \hat{\jmath}$) execution of Π_{OT}, $\mathcal{S}$ interacts with $\mathcal{A}$ using the input 0 with randomness derived from seed_j^A, and retrieves $\{\mathsf{seed}_j^B\}_{j\in[\lambda], j\neq\hat{\jmath}}$ at the end. For the $\hat{\jmath}$ execution of Π_{OT}, $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extracts $\mathsf{seed}_{\hat{\jmath}}^B$ and $\mathsf{witness}_{\hat{\jmath}}$. Let us denote the transcript of the jth execution by trans_j.

1′. $\mathcal{S}$ does nothing.

[6] Standard techniques [18, 20] can be used to ensure that $\mathcal{S}$ runs in expected polynomial time.

2′. For $j \neq \hat{\jmath}$, $\mathcal{S}$ uses as input 0^{n_A} in the execution of Π_{OT} with randomness derived from seed_j^A. In the $\hat{\jmath}$th execution of Π_{OT}, $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extracts $\{(w_{i,\hat{\jmath}}^0, w_{i,\hat{\jmath}}^1)\}_{i\in[n_A]}$. Let $\mathtt{h}_j^{\mathsf{OT}}$ denote the hash value of the transcript for the jth execution of Π_{OT}.

3′. $\mathcal{S}$ receives $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ from $\mathcal{A}$.

4′. If any signature σ_j are invalid, $\mathcal{S}$ returns to Step 0′. For $j \in [\lambda]$, $\mathcal{S}$ simulates $\mathsf{P_B}$'s execution in Steps 1, 2, and 3a, and particularly computes $\hat{\mathsf{h}}_j^{\mathsf{OT}}$ and $\hat{\mathsf{c}}_j$. Let J' be the set of indices, such that $(\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j) \neq (\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j)$.
If $|J'| = 1$ and $\hat{\jmath} \notin J'$, $\mathcal{S}$ set $\mathsf{caught}' = \mathsf{true}$. If $|J'| = 1$ and $\hat{\jmath} \in J'$, $\mathcal{S}$ set $\mathsf{caught}' = \mathsf{false}$. If $|J'| = 0$, $\mathcal{S}$ set $\mathsf{caught}' = \mathsf{nothing}$.

Then $\mathcal{S}$ follows the procedure below.

4″. If $|J'| = 1$ and $\mathsf{caught}' = \mathsf{true}$, $\mathcal{S}$ generates for the unique index $j \in J'$ a certificate $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$, sends it to $\mathcal{A}$ and halts. Otherwise, $\mathcal{S}$ sends $(\hat{\jmath}, \{\mathsf{seed}_j^B\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ to $\mathcal{A}$.

5. $\mathcal{S}$ receives $\mathsf{GC}_{\hat{\jmath}}$, $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$, $\{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}$ (in the same order as Step 3a), and $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$, together with $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$ and $\{\mathsf{decom}^{\mathsf{c}_{i,\hat{\jmath},x_B[i]}^{x_B}}\}_{i\in[n_B]}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also receives the garbled output mapping $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]}$.

6. If commitments $\mathsf{Com}(\mathsf{GC}_j, \{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}, \mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}; \mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}) \neq \mathsf{c}_{\hat{\jmath}}$, for some $i \in [n_B]$, $\mathsf{Com}(\mathsf{x}_{n_A+i}; \mathsf{decom}^{\mathsf{c}_{i,\hat{\jmath},x_B[i]}^{x_B}}) \notin \{\mathsf{c}_{i,\hat{\jmath},0}^{x_B}, \mathsf{c}_{i,\hat{\jmath},1}^{x_B}\}$, or $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$ is not consistent (if it is verifiable), $\mathcal{S}$ sends abort_B to $\mathcal{F}_{\mathsf{covertPFE}}$ and simulates $\mathsf{P_A}$'s termination. Otherwise, $\mathcal{S}$ follows the options below.
   - If $|J'| = 0$, $\mathcal{S}$ uses seed_j^B and the received information to derive $\mathsf{P_B}$'s input x_B. Then $\mathcal{S}$ sends x_B to $\mathcal{F}_{\mathsf{covertPFE}}$. If $\mathsf{P_B}$ is allowed to receive the evaluation result, $\mathcal{S}$ will receive $y \in \{0,1\}^m$ from $\mathcal{F}_{\mathsf{covertPFE}}$. Using $\mathsf{seed}_{\hat{\jmath}}^B$ to derive the output mapping and sends $\{w_{i,\hat{\jmath}}^{y[i]}\}_{i=M+1,\ldots,M+m}$ to $\mathcal{A}$.
   - If $|J'| = 1$ and $\mathsf{caught} = \mathsf{false}$, $\mathcal{S}$ derives π_f from C_f as in the protocol. Then $\mathcal{S}$ computes $t_i \leftarrow \rho_i \cdot \omega_{\pi_f(i)}^{-1} \bmod q$ for $i \in [N]$. Note that $g_i = g^{\omega_i}$ in the common reference string, and we have $p_i = g^{\rho_i}$. Let $T = [t_1, \ldots, t_N]$. $\mathcal{S}$ uses $\{(w_{i,\hat{\jmath}}^0, w_{i,\hat{\jmath}}^1)\}_{i\in[n_A]}$ and x_A from $\mathcal{F}_{\mathsf{covertPFE}}$, together with T, $\mathsf{GC}_{\hat{\jmath}}$, and $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$ to compute the output $\{\mathsf{y}_i\}_{i\in[m]}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ can derive the output $y \in \{0,1\}^m$ from the output mapping as in the protocol and sends y to $\mathcal{F}_{\mathsf{covertPFE}}$ to finish the simulation. If $\mathsf{P_B}$ is allowed to know the evaluation result, $\mathcal{S}$ sends $\{\mathsf{y}_i\}_{i\in[m]}$ to $\mathcal{A}$ and halts.

It remains to show that the joint distribution of the view of $\mathcal{A}$ simulated by $\mathcal{S}$ and the output of $\mathsf{P_A}$ in the ideal world is indistinguishable from the joint distribution of the view of $\mathcal{A}$ and the output of $\mathsf{P_A}$ in the real world. We define the following games and let the output of each game be the view of $\mathcal{A}$ and output of $\mathsf{P_A}$.

Game$_1$ We modify Step 1 of the initiation phase in this game. The EP π_f derived from C_f is used here. Then the list T, Φ is generated as in the protocol.

Corresponding Φ', c_i''s, and d_i's are also computed as in the protocol. The ideal functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{EncEP}}$ and $\mathcal{F}_{\mathsf{zk}}^{\mathsf{DH}}$ are simulated as in the protocol. Then in Step 6 of the evaluation phase, the list T is directly used for garbled circuit execution. Since the ElGamal encryption scheme is IND-CPA secure, using the same approach as in the proof of Theorem 1, we can prove that the output of $\mathbf{Game}_1$ are computationally indistinguishable from the output of $\mathbf{Game}_0$.

$\mathbf{Game}_2$ We pick a uniform $\hat{\jmath} \leftarrow_{\$} [\lambda]$ at the outset of the game. Then we modify the part of the conditional judgment branch in Step 4 of the evaluation phase as follows.

- If $|J| = 0$, $\mathcal{S}$ does the same as in $\mathbf{Game}_0$.
- If $|J| = 1$, $\mathcal{S}$ sets $\mathsf{caught} = \mathsf{true}$ if $\hat{\jmath} \notin |J|$. Otherwise, $\mathcal{S}$ sets $\mathsf{caught} = \mathsf{false}$.
- If $|J| \geq 2$, $\mathcal{S}$ sends $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$ to $\mathcal{A}$ for uniform $j \in J \backslash \{\hat{\jmath}\}$, and simulates the termination of $\mathsf{P_A}$.

When $|J| = 1$, we have $\hat{\jmath} \notin |J|$ with the same probability ϵ. Meanwhile, if $|J| \geq 2$, the probability that an index j is chosen to generate a certificate is $\frac{|J|}{\lambda} \cdot (1 - \frac{1}{|J|}) \cdot \frac{1}{|J|-1} + (1 - \frac{|J|}{\lambda}) \cdot \frac{1}{|J|} = \frac{1}{|J|}$, which is the same as in $\mathbf{Game}_0$. Therefore, the output of $\mathbf{Game}_2$ is perfectly indistinguishable from the output of $\mathbf{Game}_1$.

$\mathbf{Game}_3$ We modify the previous game as follows. In Step 0 of the evaluation phase, the simulator does not pick $\mathsf{seed}_{\hat{\jmath}}^A$. It computes $\mathsf{c}^{\mathsf{seed}_{\hat{\jmath}}^A} \leftarrow \mathsf{Com}(0^\kappa)$ alternatively. Then true random coins are used in Steps 0 and 2. It is obvious that the output of $\mathbf{Game}_3$ is computationally indistinguishable from the output of $\mathbf{Game}_2$.

$\mathbf{Game}_4$ The previous game is modified, such that $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for the $\hat{\jmath}$th execution of Π_{OT} in Steps 0 and 2 of the evaluation phase, and all $\mathcal{A}$'s inputs are extracted. According to the security of Π_{OT}, the output of $\mathbf{Game}_4$ is computationally indistinguishable from the output of $\mathbf{Game}_3$.

$\mathbf{Game}_5$ Because now Steps 0–3 are identical to Steps $0'$–$3'$ in the simulated evaluation phase, we can "collapse" the rewinding and obtain the following $\mathbf{Game}_5$ that is statistically indistinguishable from $\mathbf{Game}_4$, and the only difference is in the case of an aborted rewinding in the latter game.

0. Pick $\hat{\jmath} \leftarrow_{\$} [\lambda]$ and compute $\mathsf{c}^{\mathsf{seed}_{\hat{\jmath}}^A} \leftarrow \mathsf{Com}(0^\kappa)$. For $j \neq \hat{\jmath}$, choose uniform κ-bit strings seed_j^A, compute $\{\mathsf{c}^{\mathsf{seed}_j^A}\}$ as in the protocol, and send $\{\mathsf{c}^{\mathsf{seed}_j^A}\}_{j \in [\lambda]}$ to $\mathcal{A}$. For the jth ($j \neq \hat{\jmath}$) execution of Π_{OT}, interact with $\mathcal{A}$ using the input 0 with randomness derived from seed_j^A, and retrieve $\{\mathsf{seed}_j^B\}_{j \in [\lambda], j \neq \hat{\jmath}}$ at the end. For the $\hat{\jmath}$th execution of Π_{OT}, use the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extract $\mathsf{seed}_{\hat{\jmath}}^B$ and $\mathsf{witness}_{\hat{\jmath}}$. Let us denote the transcript of the jth execution by trans_j.
1. Do nothing.
2. For $j \neq \hat{\jmath}$, use as input 0^{n_A} for in the execution of Π_{OT} with randomness derived from seed_j^A. In the $\hat{\jmath}$th execution of Π_{OT}, use the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extract $\{(w_{i,\hat{\jmath}}^0, w_{i,\hat{\jmath}}^1)\}_{i \in [n_A]}$. Let $\mathtt{h}_j^{\mathsf{OT}}$ denote the hash value of the transcript for the jth execution of Π_{OT}.

3. Receive $\{\mathsf{c}_j, \sigma_j\}_{j\in[\lambda]}$ from $\mathcal{A}$.
4. If any signature σ_j are invalid, output $\perp$ and halt. For $j \in [\lambda]$, emulate $\mathsf{P_B}$'s execution in Steps 1, 2, and 3a, and particularly compute $\hat{\mathsf{h}}_j^{\mathsf{OT}}$ and $\hat{\mathsf{c}}_j$. Let J' be the set of indices, such that $(\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j) \neq (\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j)$.
   - If $|J| = 0$, send $(\hat{\jmath}, \{\mathsf{seed}_j^B\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ to $\mathcal{A}$ and continue below.
   - If $|J| = 1$ and $\hat{\jmath} \notin |J|$ or $|J| \geq 2$, send for uniform $j \in J\backslash\{\hat{\jmath}\}$ a certificate $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$ to $\mathcal{A}$. Then output $\mathsf{corrupted}$ and halt.
   - If $|J| = 1$ and $\hat{\jmath} \in |J|$, send $(\hat{\jmath}, \{\mathsf{seed}_j^B\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ to $\mathcal{A}$ and continue below.
5. Receive $\mathsf{GC}_{\hat{\jmath}}$, $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$, $\{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}$ (in the same order as Step 3a), and $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$, together with $\mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}$ and $\{\mathsf{decom}^{\mathsf{c}_{i,\hat{\jmath},x_B[i]}^{x_B}}\}_{i\in[n_B]}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, $\mathcal{S}$ also receives the garbled output mapping $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]}$.
6. If commitments $\mathsf{Com}(\mathsf{GC}_j, \{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}, \mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}; \mathsf{decom}^{\mathsf{c}_{\hat{\jmath}}}) \neq \mathsf{c}_{\hat{\jmath}}$, for some $i \in [n_B]$, $\mathsf{Com}(\mathsf{x}_{n_A+i}; \mathsf{decom}^{\mathsf{c}_{i,\hat{\jmath},x_B[i]}^{x_B}}) \notin \{\mathsf{c}_{i,\hat{\jmath},0}^{x_B}, \mathsf{c}_{i,\hat{\jmath},1}^{x_B}\}$, or $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$ is not consistent (if it is verifiable), output $\perp$ and abort.
   Otherwise, follow the options below.
   - If $|J| = 0$, use seed_j^B and the received information to derive $\mathsf{P_B}$'s input x_B. Then compute $y \leftarrow C_f(x_A, x_B)$. If $\mathsf{P_A}$ is allowed to learn the evaluation result, output y and halt. If $\mathsf{P_B}$ is allowed to receive the evaluation result, use seed_j^B to derive the output mapping and send $\{w_{i,\hat{\jmath}}^{y[i]}\}_{i=M+1,\ldots,M+m}$ to $\mathcal{A}$ and halt.
   - If $|J| = 1$, use $\{(w_{i,\hat{\jmath}}^0, w_{i,\hat{\jmath}}^1)\}_{i\in[n_A]}$ and x_A, together with T, $\mathsf{GC}_{\hat{\jmath}}$, and $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$ to compute the output $\{\mathsf{y}_i\}_{i\in[m]}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, derive the output from the output mapping as in the protocol, and then output the result and halt. If $\mathsf{P_B}$ is allowed to know the evaluation result, send $\{\mathsf{y}_i\}_{i\in[m]}$ to $\mathcal{A}$ and halts.

Game$_6$ We modify Step 6 of the evaluation phase in the previous game. If $|J| = 0$, use $\{w_{i,\hat{\jmath}}^{x_A[i]}\}_{i\in[n_A]}$, together with T, $\mathsf{GC}_{\hat{\jmath}}$, and $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$ to compute the output $\{\mathsf{y}_i\}_{i\in[m]}$. If $\mathsf{P_A}$ is allowed to know the evaluation result, derive the output y from the output mapping as in the protocol, and then output y and halt. If $\mathsf{P_B}$ is allowed to know the evaluation result, send $\{\mathsf{y}_i\}_{i\in[m]}$ to $\mathcal{A}$ and halts.

Because $|J| = 0$, we know that the commitment $\mathsf{c}_{\hat{\jmath}}$ commits to a correctly computed garbled circuit, input-wire labels $\{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}$ (in correct order), and the hash value of the output-wire labels. According to the binding property of the commitment scheme and the collision-resistance property of the hash function, $\mathsf{GC}_{\hat{\jmath}}$, $\{\mathsf{c}_{i,\hat{\jmath},b}^{x_B}\}_{i\in[n_B],b\in\{0,1\}}$, and $\mathtt{h}_{\hat{\jmath}}^{\mathsf{O}}$ (may also together with the output mapping $\{(w_{M+i}^0, w_{M+i}^1)\}_{i\in[m]}$) sent by $\mathcal{A}$ are all correct. In addition, since $|J| = 0$, the collision-resistance property of the hash function ensures that $\mathsf{P_A}$'s input-wire labels sent in Π_{OT} are correct. Therefore, using

$\{w_{i,\hat{\jmath}}^{x_A[i]}\}_{i\in[n_A]}$, together with T, $\mathsf{GC}_{\hat{\jmath}}$, and $\{\mathsf{x}_{n_A+i}\}_{i\in[n_B]}$ for the execution of garbled circuit will derive the correct result.

Hence, the output of $\mathbf{Game}_6$ is computationally indistinguishable from the output of $\mathbf{Game}_5$.

$\mathbf{Game}_7$ Here Step 4 in the evaluation phase of the previous game is changed in the following. $\mathsf{P_B}$'s executions are emulated for $j \in [\lambda]\backslash\{\hat{\jmath}\}$. Let $\hat{J}$ be the set of indices, such that $(\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j) \neq (\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j)$.

- If $|\hat{J}| = 0$, send $(\hat{\jmath}, \{\mathsf{seed}_j^B\}_{j\neq\hat{\jmath}}, \mathsf{witness}_{\hat{\jmath}})$ to $\mathcal{A}$ and continue below.
- If $|\hat{J}| \neq 0$, send $\mathsf{cert} = (P, j, \mathsf{trans}_j, \mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j, \sigma_j, \mathsf{seed}_j^A, \mathsf{decom}^{\mathsf{seed}_j^A})$ for uniform $j \in \hat{J}$ to $\mathcal{A}$. Then output $\mathsf{corrupted}$ and halt.

Let the set J as defined before. Note that the condition that $|J| = 0$ or $|J| = 1 \wedge \hat{\jmath} \in |J|$ is equal to the condition $|\hat{J}| = 0$. Meanwhile, the condition that $|J| = 1 \wedge \hat{\jmath} \notin |J|$ or $|J| \geq 2$ is the same as the condition $|\hat{J}| \neq 0$. Thus, the output of $\mathbf{Game}_7$ is perfectly indistinguishable from the output of $\mathbf{Game}_6$.

$\mathbf{Game}_8$ In this game, the $\hat{\jmath}$th execution of Π_{OT} in Step 0 and Step 2 are executed honestly, *i.e.*, use input 1 to the protocol in Step 0 and x_A in Step 2. Following the security of Π_{OT}, the output of $\mathbf{Game}_8$ is computationally indistinguishable from the output of $\mathbf{Game}_7$.

$\mathbf{Game}_9$ We modify Step 0 of the previous game by choosing $\mathsf{seed}_{\hat{\jmath}}^A$ and computing $\{\mathsf{c}^{\mathsf{seed}_{\hat{\jmath}}^A}\}$ as in the protocol. Then for the $\hat{\jmath}$th execution of Π_{OT} in Step 0 and Step 2, use the random coins derived from $\mathsf{seed}_{\hat{\jmath}}^A$. It is easy to see that the output of $\mathbf{Game}_9$ is computationally indistinguishable from the output of $\mathbf{Game}_8$.

Note that $\mathbf{Game}_9$ corresponds to an execution of the protocol for $\mathsf{P_A}$ holding input C_f and x_A in the real world, while $\mathbf{Game}_0$ corresponds to the simulated execution in the ideal world. We also note that the certificate is for an index $j \in J\backslash\{\hat{\jmath}\}$, while only the $\hat{\jmath}$th execution involves $\mathsf{P_B}$'s input. Therefore, even if $\mathcal{A}$ receives cert, $\mathcal{A}$ cannot derive any information about $\mathsf{P_A}$'s input. Hence, we complete the proof for malicious $\mathsf{P_B}$.

We now describe how the protocol achieves public verifiability. From the protocol, it is easy to see that once an honest $\mathsf{P_A}$ outputs $\mathsf{corrupted}_B$, she is able to output a certificate cert to blame $\mathsf{P_B}$'s misbehavior. If $\mathsf{P_B}$ intends to deviate from the protocol covertly, he might deviate in Steps 1, 2, or 3a, *i.e.*, $\mathsf{P_B}$ does not follow the execution specified by the protocol and the corresponding seed. Hence, there exists a message from $\mathsf{P_B}$ that is not consistent with the message he should send according to the protocol and the seed. If an honest $\mathsf{P_A}$ publishes a certificate cert, then $\mathsf{P_A}$ has obtained $\mathsf{P_B}$'s seed for the derandomized execution and detects $\mathsf{P_B}$'s covert cheating in this execution. Since the corresponding transcript is signed by $\mathsf{P_B}$, everyone is able to verify the inconsistency. More precisely, given the verification key vk, a certificate cert, and a common reference string G, anyone can execute the algorithm Judge to check whether the messages from $\mathsf{P_B}$ are consistent with an honest execution. More importantly, thanks to the OT protocol, $\mathsf{P_B}$ does not know whether his misbehavior is detected until $\mathsf{P_A}$

publishes the certificate cert. Thus, $\mathsf{P_B}$ cannot abort before the time that $\mathsf{P_A}$ can generate the certificate.

Finally, we show that the protocol achieves defamation freeness. Assume that a malicious $\mathsf{P_A}$ intends to break the defamation freeness of the protocol and blames an honest $\mathsf{P_B}$. According to the description of the algorithm Judge, the algorithm will output 1 only if $(\mathsf{h}_j^{\mathsf{OT}}, \mathsf{c}_j) \neq (\hat{\mathsf{h}}_j^{\mathsf{OT}}, \hat{\mathsf{c}}_j)$. If c_j is inconsistent, it means that the garbled circuit is not correctly generated using the random coins derived from seed_j^B. However, since $\mathsf{P_B}$ is honest and corresponding material for generating the garbled circuit, *i.e.*, G and P, is signed by $\mathsf{P_B}$, we know that seed_j^B derived from the simulation of Π_{OT} is incorrect, or the signature is forged. On the one hand, the signature scheme is EUF-CMA, a computationally bounded $\mathsf{P_A}$ cannot forge the signature except for a negligible probability. On the other hand, for the simulation of Π_{OT}, the transcript trans_j is already verified and signed by $\mathsf{P_B}$, this means that if the output of Π_{OT} is not the correct seed_j^B, this incorrect output seed_j^B imputes to the random coins used by $\mathsf{P_A}$. Since the commitment $\mathsf{c}^{\mathsf{seed}_j^A}$ is signed by $\mathsf{P_B}$ and Π_{OT} is perfectly correct, the output of the simulation of Π_{OT} cannot be equivocated unless $\mathsf{P_A}$ breaks the binding property of the commitment scheme. Hence, malicious $\mathsf{P_A}$ cannot incur an inconsistent c_j except for a negligible probability. For the hash value $\mathsf{h}_j^{\mathsf{OT}}$, a malicious $\mathsf{P_A}$ can incur an inconsistent $\hat{\mathsf{h}}_j^{\mathsf{OT}}$ only if seed_j^A and seed_j^B produces an incorrect $\hat{\mathsf{h}}_j^{\mathsf{OT}}$. Similar to c_j, a malicious $\mathsf{P_A}$ cannot make Π_{OT} output an incorrect seed_j^B, and thus incur the algorithm Judge to output 1, except for a negligible probability. Therefore, the protocol achieves defamation freeness. $\square$