Highly Efficient and Reusable Private Function Evaluation with Linear Complexity

Osman Biçer[1], Muhammed Ali Bingöl[2,3], Mehmet Sabır Kiraz[3], Albert Levi[2]

[1] Koç University, Graduate School of Sciences and Engineering, İstanbul, Turkey
[2] Sabancı University, Faculty of Engineering and Natural Sciences, İstanbul, Turkey
[3] TÜBİTAK BİLGEM Kocaeli, Turkey

Abstract. Private function evaluation aims to securely compute a function $f(x_1, \ldots, x_n)$ without leaking any information other than what is revealed by the output, where f is a private input of one of the parties (say Party_1) and x_i is a private input of the i-th party Party_i. In this work, we propose a novel and secure *two-party private function evaluation* (2PFE) scheme based on DDH assumption. Our scheme introduces a reusability feature that significantly improves the state-of-the-art. Accordingly, our scheme has two variants, one is utilized in the first evaluation of the function f, and the other is utilized in its subsequent evaluations. To the best of our knowledge, this is the first and most efficient special purpose PFE scheme that enjoys a reusablity feature. Our protocols achieve linear communication and computation complexities and a constant number of rounds which is at most three.

Keywords: Private function evaluation, Secure 2-party computation, Communication complexity, Cryptographic protocol.

1 Introduction

Private function evaluation (PFE) is a special case of secure multi-party computation where the function to be evaluated is also a private input of one of the parties. In this paper, we consider the two-party PFE (2PFE) setting where the first party (say Party_1) has a function input f (compiled into a boolean circuit C_f) and optionally[4] a private input bit string x_1, whereas the other party (say Party_2) has an input bit string x_2. The parties aim to evaluate f on x_1 and x_2 so that at least one of them would obtain the resulting $f(x_1, x_2)$ without any of them deducing any information about the other one's private input beyond what $f(x_1, x_2)$ itself reveals.

Efficient and practical PFE schemes are becoming increasingly important as many real-world applications require protection of their valuable assets. For example, many software companies targeting the global market are extremely concerned about illegal reproduction of their software products. Software obfuscation methods usually prevent reverse engineering, but still allow direct copying

[4] Note that PFE also comprises the setting where $x_1 = \perp$.

of programs. Another solution could be providing the software-as-a-service in the cloud to eliminate the risk of exposure. However, this solution also causes another issue, i.e., threatening the privacy of customer data, since computations need to take place at the hands of software vendors. Fully homomorphic encryption (FHE) can also be a potential solution to such problems [Gen09], but, unfortunately, it is still far from being practical [HS15]. Another decent approach targeting those problems falls into the category of PFE. Compared to FHE, PFE is currently much closer to practical use. Moreover, in many occasions such a PFE scheme is quite beneficial, including the ones where a service provider may opt keeping the functionality and/or its specific implementation confidential, and the ones where the disclosure of the function itself means revelation of sensitive information, or causes a security weakness.

The current research goal for secure computation protocols (including PFE) is efficient and practical solutions with low round, communication, and computation complexities. Among these three measures, as also pointed out by Beaver, Micali, and Rogaway, the number of rounds is the most valuable resource [BMR90]. The other important research goal in this area is the minimization of communication complexity. Since hardware trends show that computation power progresses more rapidly compared to communication channels, the main bottleneck for many applications will be the bandwidth usage.

1.1 Related Work

First proposed by Andrew Yao [Yao82, Yao86], secure two-party computation (2PC) comprises the techniques for joint evaluation of a function by two parties on their respective secret inputs. In recent years, there have been a promising progress over the original Yao's protocol [BMR90, NPS99, KS08a, PSSW09, KMR14, ZRE15, KKS16]. As a consequence of these improvements, secure computation techniques now have promising results.

2PFE differs from the standard 2PC in that the latter involves both parties evaluating a publicly known function on their private inputs, whereas in the former, the function itself is also a private input. 2PFE concept is first appeared in [AFK87, AF90]. So far, there are basically two main approaches that PFE solutions are built upon.

The first one is based on a universal circuit which takes a boolean circuit C with circuit size less than g and an input $(x_1, \ldots, x_n)$, and outputs $C(x_1, \ldots, x_n)$. The idea is that if the regular secure computation techniques can be applied on a universal circuit, then a PFE scheme can be obtained. Consequently, the efforts targeting the efficiency of *universal circuit based PFEs* have generally been towards reducing the size of universal circuits, and the cost of their secure computation [KS08b, SS09, KS16, GKS17].

The second approach is avoiding the use of universal circuits and designing special purpose PFE protocols. Following this line of work, several PFE schemes have been proposed, e.g., as [PSS09,KM11,MS13,Sad15,BBKL17]. A remarkable work embracing this approach is *singly homomorphic encryption* based 2PFE scheme of Katz and Malka applied on boolean circuits [KM11]. [KM11] utilizes

a singly homomorphic scheme (e.g., ElGamal [EG85] or Paillier [Pai99]) for the generation of the two random tokens[5] on each wire, later utilized in the 2PC stage. They first propose a basic version of their protocol in [KM11, Sect. 3.1] and for the efficiency concerns they propose a more efficient variant in [KM11, Sect. 3.2]. Both schemes have only three rounds, and provides $O(g)$ asymptotic complexity in terms of communication and computation, where g denotes the circuit size. The latter one reduces the communication and offline computation complexity.

In [MS13], Mohassel and Sadeghian proposed 2PFE schemes, for boolean circuits and arithmetic circuits. Considering boolean circuits, they propose two types of protocols: one is based on oblivious evaluation of switching networks (OSN) (what we call [MS13]-OSN) and the other one is based on singly homomorphic encryption (what we call [MS13]-HE). Even though [MS13]-OSN is efficient for small sized circuits, it is still inefficient for large circuits due to its $O(g \log(g))$ communication and computation complexities. It fails to outperform asymptotically linear communication and computation complexities of [KM11]. On the other hand, [MS13]-HE provides linear communication and computation complexities and slightly outperforms [KM11, Sect. 3.2]. We remark that to the best of our knowledge, a reusability feature cannot be adapted[6] to [KM11], [MS13]-HE, or [MS13]-OSN.

[MS13] also proposes a protocol for arithmetic circuits based on partial (singly) homomorphic encryption. This protocol has equal number of rounds to its gates (see [MS13, p. 570]), whereas the other PFE protocols for boolean circuits have constant number of rounds. For large circuits the number of rounds will be a bottleneck[7]. [MS13] also proposes a multi-party PFE variant based on OSN that remains the most efficient one to date. Their proposals are essentially secure in the semi-honest model, and has later been extended to the malicious model by [MSS14].

Recently [BBKL17] improves the OSN based 2PFE protocol of [MS13]. They show how to utilize the elegant *half gates* technique [ZRE15] to their 2PFE scheme. Compared to [MS13]-OSN protocol, the optimization of [BBKL17] improves by more than 40% reduction in overall communication cost. However, it still has the inherited asymptotical complexities of [MS13]-OSN.

[5] Throughout this paper, the term "token" stands for a random bit string generated for a wire of the boolean circuit, and has hidden semantics of either 0 or 1.

[6] This is due to the fact that the blinding operations in these protocols are one-time pads (XOR or cyclic addition), therefore, reusing the blinded values inevitably leaks information about the truth values of intermediate wires. On the other hand, our mechanism relies on DDH so that the blinding values would remain unknown to the respective parties.

[7] We can intuitively say that as the latency between parties increases, so does the cost of each additional communication round (we refer to [SZ13] that backs up this discussion). A similar analysis on trade-offs between boolean and arithmetic circuit based protocols has also been addressed in [CKMZ14, p. 527].

1.2 Our Contributions

In this work, we propose a highly efficient 2PFE scheme for boolean circuits secure in the semi-honest model. Our scheme enjoys the cost reduction due to the reusability of tokens that will be used in the 2PC stage. This eliminates some of the computations and exchanged messages in the following re-executions for the same function. Therefore, one of the strongest aspects of our proposed protocol is the remarkable cost reduction if the same function is evaluated more than once (possibly on varying inputs). We highlight that such a cost reduction is not applicable to the protocols of [KM11] and [MS13] since they require running the whole protocol from scratch. In this respect, we present two variants of our scheme: (1) single execution protocol, (2) re-execution protocol. The former protocol is utilized in the first evaluation of the function, while the latter one is utilized in the second or later evaluations of the same function between the two parties. We note that the latter protocol is more efficient than the former one due to the fact that it benefits from the reusable tokens generated already in the first evaluation. The latter case is likely to be encountered more frequently in practice, compared to the cases where the function is evaluated just once between the two given parties.

In what follows, we summarize the contributions of our single and re-execution protocols in comparison with the state-of-the-art protocols.

- Regarding our single execution protocol, independent of the circuit size, our scheme provides 12.5% reduction in communication cost over [KM11, Sect. 3.2]. Compared to [MS13]-OSN, [BBKL17], and [GKS17], our protocol asymptotically reduces the communication cost. Namely, while the asymptotic communication costs of those protocols are equal to $O(g \log(g))$, our first protocol provides $O(g)$ communication complexity (similar to the ones in [KM11]) where g is the number of gates. To illustrate the significance of this asymptotical improvement, our cost reduction is about 94% over [MS13]-OSN, about 88% over [BBKL17], and 68% over [GKS17] for a thousand-gate circuit. Moreover, for a billion-gate circuit, our cost reduction is about 98% over [MS13]-OSN, about 96% over [BBKL17], and about 89% over [GKS17]. Note that the number of protocol rounds of [MS13]-OSN is equal to 6, while that of our first protocol is equal to 3 (similar to the ones in [KM11]).
- Regarding our re-execution protocol, also independent of the circuit size, our scheme achieves about 50% reduction in communication cost over [KM11, Sect. 3.2]. Similar to our first protocol, the asymptotic communication complexity of our second protocol is also superior to [MS13]-OSN, [BBKL17], and [GKS17] (while being equal to those of [KM11]). To clarify, the cost reduction is about 96% over [MS13]-OSN, about 93% over [BBKL17], and about 81% over [GKS17] for a thousand-gate circuit. Moreover, for a billion-gate circuit, the cost reduction is about 99% over [MS13]-OSN, about 97% over [BBKL17], and about 94% over [GKS17]. Moreover, the number of rounds of our re-execution protocol is equal to 1, or 2, or 3 depending on the

Table 1. Comparison with existing constant-round 2PFE schemes (the OSN based [MS13], the HE based [MS13] protocol, both [KM11] protocols, [BBKL17], and [GKS17]) in terms of overall communication (in bits) and online computation costs (in terms of symmetric-key operations), offline computation costs (in terms of symmetric-key operations), and the number of rounds. M, N, λ, and ρ denote the number of outgoing wires (i.e., equal to $n+g-m$), the number of incoming wires (i.e., equal to $2g$), the security parameter, and the computation cost ratio, respectively. The costs of [GKS17] are approximated.

	Communication	**Online Comp.**	**Offline Comp.**	**Rounds**
[KM11, Sec.3.1]	$(4M+10N)\lambda$	$(\rho+2.5)N$	$4(M+N)\rho$	3
[KM11, Sec.3.2]	$(2M+7N)\lambda$	$(\rho+2.5)N$	$2(M+N)\rho$	3
[MS13]-OSN	$(10N\log_2 N+4N+5)\lambda$	$6N\log_2 N+2.5N+3$	$O(\lambda)$	6
[MS13]-HE	$(2M+6N)\lambda$	$(\rho+2.5)N$	$2(M+N)\rho$	3
[BBKL17]	$(6N\log_2 N+0.5N+3)\lambda$	$6N\log_2 N+N+3$	$O(\lambda)$	6
[GKS17]	$(2N\log_2 N)\lambda$	$0.7N\log_2 N$	$2N\log_2 N$	3
Our 1st Pro.	$(2M+6N)\lambda$	$(4\rho+2.5)N$	$(3M-1)\rho$	3
Our 2nd Pro.	$4N\lambda$	$(\rho+0.5)N$	$2(M+N)\rho+2$	1 / 2 / 3

input string length of Party_1 (i.e., owner of f).[8] This reflects the improvement of our re-execution variant over other 2PFE protocols also in terms of round complexity.

– We also deal with the case that Party_1 runs the 2PFE protocol for the same private function with various Party_2s separately. This is a common scenario where Party_1 may run a business with many customers for her algorithm/software. Trivially, our re-execution protocol can be utilized between the same two parties in the second and subsequent evaluations after the first evaluation. In order to eliminate the requirement of running the single execution protocol with each Party_2, we propose a more efficient mechanism for the generation of the reusable tokens by employing a threshold based system.

Table 1 compares the existing 2PFE schemes in terms of overall communication and online computation costs, and the number of rounds for λ-bit security. [MS13] results in $O(N\log_2 N)$ whereas [KM11] and our protocols achieve linear complexities[9] in terms of communication and computation. We also provide Table 2 that depicts a comparison in terms of overall communication costs for various circuit sizes. Note that although the complexity of [MS13]-HE is also comparable to our single execution protocol, in the later executions parties can enjoy the massive cost reduction of our reusability feature, which is not possible for [MS13]-HE.

[8] More concretely, if Party_1 has $x_1=\perp$, then the number of rounds is equal to 1. If Party_1 has a non-empty input x_1 in such that the OT extension is not applicable for its garbled input, then the number of rounds is equal to 2. Otherwise, the number of rounds is equal to 3.

[9] Note that $M \leq N$, therefore $O(M+N)=O(N)$.

Table 2. Comparison with the existing constant-round 2PFE schemes OSN based protocol of [MS13], HE based [MS13] protocol, both [KM11] protocols, [BBKL17], and [GKS17]) in terms of overall communication costs for various circuit sizes. Here we assume that $N = 2M$ (i.e., the number of output bits of the function equals the number of its input bits) and $\lambda = 128$. The values calculated for [GKS17] are approximated.

	Number of Gates				
	2^{10}	2^{15}	2^{20}	2^{25}	2^{30}
[KM11, Sec.3.1]	0.38 MB	12.00 MB	0.38 GB	12.00 GB	384.00 GB
[KM11, Sec.3.2]	0.25 MB	8.00 MB	0.25 GB	8.00 GB	256.00 GB
[MS13]-OSN	3.56 MB	164.00 MB	6.69 GB	264.00 GB	10,048.00 GB
[MS13]-HE	0.22 MB	7.00 MB	0.22 GB	7.00 GB	224.00 GB
[BBKL17]	1.89 MB	90.50 MB	3.77 GB	151.00 GB	5,776.00 GB
[GKS17]	0.68 MB	32.00 MB	1.31 GB	52.00 GB	1,984.00 GB
Our 1st Pro.	0.22 MB	7.00 MB	0.22 GB	7.00 GB	224.00 GB
Our 2nd Pro.	0.13 MB	4.00 MB	0.13 GB	4.00 GB	128.00 GB

1.3 Organization

In Section 2, we give a preliminary background that is used throughout the paper. Section 3 presents our baseline mechanism for 2PFE and further precomputation optimization on it for resource efficiency. In Section 4, we further propose two new methods to achieve more efficient re-executions between the two parties and in the presence of multiple Party_2s. Section 5 provides the complexities of our resulting protocols, and compare them with the other state-of-the-art 2PFE protocols. Finally, in Section 6, we give the security proofs of our protocols in the semi-honest model.

2 Preliminaries

This section provides some background information on the DDH assumption and the state-of-the-art generic 2PFE framework.

2.1 Decisional Diffie-Hellman Assumption

The Decisional Diffie-Hellman (DDH) assumption for $\mathbb{G}$ provides that the following two ensembles are computationally indistinguishable

$$\{(P_1, P_2, a \cdot P_1, a \cdot P_2) : P_i \in \mathbb{G}, a \in_R \mathbb{Z}_q^*\} \approx_c$$

$$\{(P_1, P_2, a_1 \cdot P_1, a_2 \cdot P_2) : P_i \in \mathbb{G}, a_1, a_2 \in_R \mathbb{Z}_q^*\}.$$

where $X \approx_c Y$ denotes that the sets X and Y are computationally indistinguishable. The security of our protocols is based on the following lemma of Naor and Reingold [NR04] providing a natural generalization of the DDH assumption for $m > 2$ generators.

Lemma 1 ([NR04]). *Under the DDH assumption on $\mathbb{G}$, for any positive integer m,*

$$\{(P_1, \ldots, P_m, a \cdot P_1, \ldots, a \cdot P_m) : P_i \in \mathbb{G}, a \in_R \mathbb{Z}_q^*\} \approx_c$$

$$\{(P_1, \ldots, P_m, a_1 \cdot P_1, \ldots, a_m \cdot P_m) : P_i \in \mathbb{G}, a_1, \ldots, a_m \in_R \mathbb{Z}_q^*\}.$$

There exist certain elliptic curve groups where the DDH assumption holds. We will not go through the details of these primitives and refer the reader to [Bon98, HMV03]. The main advantage of the elliptic curve DDH assumption over the discrete logarithm based DDH assumption is that the discrete logarithm DDH problem requires sub-exponential time [LV01] while the current best algorithms known for solving the elliptic curve DDH problem requires exponential time resulting the same security with smaller key sizes. Therefore, in general, the elliptic curve based systems are more practical than the classical discrete logarithm systems since smaller parameters may be chosen to ensure the same level of security. For example, for the 112-bit symmetric key security level, a 2048-bit large prime number is required for a discrete logarithm group, whereas only a 224-bit prime p is sufficient for a NIST-elliptic curve over $\mathbb{F}_p$ [Gir16].

2.2 Notations and Concept of 2PFE Framework

In a two-party private function evaluation (2PFE) scheme, Party_1 has a function input f (compiled into a boolean circuit C_f) and optionally a private input bit string x_1, whereas Party_2 has an input bit string x_2. The parties aim to evaluate f on x_1 and x_2 so that at least one of them would obtain the resulting $f(x_1, x_2)$. The recent 2PFE schemes [KM11, MS13] conform to a generic 2PFE framework (formalized by [MS13]) that basically reduces the 2PFE problem to hiding both parties' input strings and topology of the circuit. The framework is not concerned with hiding the gates since it allows only one type of gate in the circuit structure.

We briefly describe the generic 2PFE framework as follows. Before starting the 2PFE protocol, Party_1 compiles the function into a boolean circuit C_f consisting of only one type of gates (e.g., NAND gates). Let g, n, and m denote the number of gates (circuit size), the number of inputs, and the number of outputs of C_f, respectively. Let $\mathsf{OW} = (\mathsf{ow}_1, \ldots, \mathsf{ow}_{n+g-m})$ denote the set of outgoing wires that is the union of the input wires of the circuit and the output wires of its non-output gates. Note that the total number of elements in OW is $M = n + g - m$. Similarly, let $\mathsf{IW} = (\mathsf{iw}_1, \ldots, \mathsf{iw}_{2g})$ denote the set of incoming wires that is the union of the input wires of each gate in the circuit. Note also that the total number of elements in IW is $N = 2g$. Throughout this paper, M and N denote the numbers of outgoing and incoming wires, respectively. Let

π_f be a mapping such that $j \leftarrow \pi_f(i)$ if and only if $\mathsf{ow}_i \in \mathsf{OW}$ and $\mathsf{iw}_j \in \mathsf{IW}$ correspond to the same wire in the circuit C_f.

We define the public information of the circuit C_f as $\mathsf{PubInfo}_{C_f}$ which is composed of: (1) the number of each party's input bits, (2) the number of output bits, (3) the total number of incoming wires N and that of outgoing wires M, (4) the incoming and outgoing/output wire indices that belong to each gate, (5) the outgoing wire indices corresponding to each party's input bits. Note that, it is a common assumption among PFE schemes [KM11,MS13,BBKL17] that both parties have pre-agreement on the number of gates (g), the number of input wires (n), the number of output wires (m), the number of input bits of Party_1 (q). Both parties generate $\mathsf{PubInfo}_{C_f}$ at the beginning of the protocol execution (without an additional round of communication). Namely, each party runs the following deterministic procedure to obtain $\mathsf{PubInfo}_{C_f}$ on public input (g, n, m, q):

- Set $N := 2g$, $M := n + g - m$.
- For $i = 1, \ldots, g$, set iw_{2i-1} and iw_{2i} as the incoming wires of the gate G_i.
- For $i = 1, \ldots, g - m$, set ow_i as the outgoing wire of the gate G_i.
- For $i = 1, \ldots, q$, set ow_{g-m+i} as the outgoing wire corresponding to Party_1's i-th input bit.
- For $i = 1, \ldots, n - q$, set $\mathsf{ow}_{g-m+q+i}$ as the outgoing wire corresponding to Party_2's i-th input bit.
- For $i = 1, \ldots, m$, set the output wire y_i as the output of G_{g-m+i}.
- Return $\mathsf{PubInfo}_{C_f} := (M, N, \mathsf{OW}, \mathsf{IW}, y)$.

Next, Party_1 generates π_f (i.e., the connection between incoming and outgoing wire indices) using the following randomized procedure on input $(C_f, \mathsf{OW}, \mathsf{IW})$.

- Randomly permute the indices $1, \ldots, g - m$, and assign it to an ordered set A.
- For $i = 1, \ldots, g - m$, assign $G_{A[i]}$ to the i-th non-output gate in topological order.
- For $i = 1, \ldots, m$, assign G_{g-m+i} to i-th output gate.
- Extract π_f from C_f according to the connections between ows and iws.
- Return π_f.

During the protocol execution, Party_1 and Party_2 first engage in a mapping evaluation protocol where Party_2 obliviously obtains the tokens on gate inputs, and then they mutually run a 2PC protocol (i.e., conventional Yao's scheme) where Party_2 garbles each gate separately using those tokens, and Party_1 evaluates the garbled circuit.

3 Our Primary Proposal

In this section, as a warm up, we first present our baseline scheme without any optimization. We then optimize it by offline/online decomposition, and obtain a more efficient mechanism in terms of online computation in Figure 1.

3.1 The Description of the Baseline Scheme

Prior to the protocol execution, both parties should have a pre-agreement on a cyclic group $\mathbb{G}$ of large prime order $q \in O(\lambda)$ with a generator P. In accordance with the generic 2PFE framework (see Section 2.2), our protocol follows the following procedures. In the preparation stage, Party_1 compiles the function into a boolean circuit C_f consisting of only one type of gates (e.g., NAND gates). Each party runs the deterministic procedure described in Section 2.2 to generate $\mathsf{PubInfo}_{C_f}$ on input (g, n, m, q). Party_1 then extracts π_f. Party_2 picks M other generators for the generator set $\mathcal{P} := (P_1, \ldots, P_M)$, and then sends $\mathcal{P}$ to Party_1. Party_1 assigns the random set $T := (t_1, \ldots, t_j)$ such that $t_j \in_R \mathbb{Z}_q^*$ for $j = 1, \ldots, N$. Next, Party_1 computes $Q_j := t_j \cdot P_{\pi_f^{-1}(j)}$ for $j = 1, \ldots, N$ where $\pi_f^{-1}(j)$ denotes the index of the outgoing wire connected to iw_j, and sends the set $\mathcal{Q} := (Q_1, \ldots, Q_N)$ to Party_2. Now, both parties have the knowledge of the set $(\mathcal{P}, \mathcal{Q})$, which we define as *Reusable Mapping Template*[10] ($\mathsf{ReuseTemp}_f$).

Definition 1 (Reusable Mapping Template). *Let π_f and π_f^{-1} be defined as above. A Reusable Mapping Template is a set $\mathsf{ReuseTemp}_f := (\mathcal{P}, \mathcal{Q})$ such that $\mathcal{P} := (P_1, \ldots, P_M)$ where P_i is a generator of the group picked for ow_i by Party_2 and $\mathcal{Q} := (Q_1, \ldots, Q_N)$ where $Q_j := t_j \cdot P_{\pi_f^{-1}(j)}$ is a group element generated for iw_j by Party_1 for $t_j \in_R \mathbb{Z}_q^*$, $i = 1, \ldots, M$, and $j = 1, \ldots, N$.*

Party_2 then picks $\alpha_0, \alpha_1 \in_R \mathbb{Z}_q^*$, keeps them private, and computes the following four ordered sets

$$\mathcal{W}^0 := (W_1^0, \ldots, W_M^0 : W_i^0 \twoheadleftarrow \alpha_0 \cdot P_i,\ i = 1, \ldots, M),$$
$$\mathcal{W}^1 := (W_1^1, \ldots, W_M^1 : W_i^1 \twoheadleftarrow \alpha_1 \cdot P_i,\ i = 1, \ldots, M),$$
$$\mathcal{V}^0 := (V_1^0, \ldots, V_N^0 : V_j^0 \twoheadleftarrow \alpha_0 \cdot Q_j,\ j = 1, \ldots, N),$$
$$\mathcal{V}^1 := (V_1^1, \ldots, V_N^1 : V_j^1 \twoheadleftarrow \alpha_1 \cdot Q_j,\ j = 1, \ldots, N).$$

Party_2 next generates the following two randomly chosen ordered sets for output wires of the circuit

$$Y^0 := (y_1^0, \ldots, y_o^0 : y_i^0 \twoheadleftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o),$$
$$Y^1 := (y_1^1, \ldots, y_o^1 : y_i^1 \twoheadleftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o)$$

where ℓ is the bit length of a group element (i.e., $\ell = \lceil \log_2(q) \rceil$).

Both parties then engage in a 2PC protocol where Party_2 and Party_1 play the garbler and evaluator roles, respectively. Party_2 garbles the whole circuit by using $\mathcal{W}^0$, $\mathcal{W}^1$, $\mathcal{V}^0$, $\mathcal{V}^1$, Y^0, Y^1, and $\mathsf{PubInfo}_{C_f}$. Note that in contrast to

[10] The efficiency of our scheme mostly results from the reusability of $\mathsf{ReuseTemp}_f$ during token generations in re-executions for the same private function. Although [KM11] also involves homomorphic encryption for token generation, it requires all protocol steps to be repeated in each re-execution.

the usual garbling in [KM11, MS13], in our garbling phase, Party_2 has group elements instead of random tokens. To use group elements as keys, we now define an instantiation of a dual-key cipher (DKC) notion of [BHR12] using a pseudorandom function as

$$\mathsf{Enc}_{P_1,P_2}(m) := [H(P_1, P_2, \mathsf{gateID})]_\ell \oplus \mathsf{m}$$

where P_1 and P_2 are two group elements used as keys, m is the ℓ-bit plaintext, gateID is the index number of the gate, $H : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \to \{0,1\}^{\ell+\tau}$ is a hash-function (which we model as a random oracle), τ is an integer such that $\tau > 2\log_2(4g)$ for preventing collisions in the τ rightmost bits of hashes, and $[H(X)]_\ell$ denotes the truncated hash value (of the message X) which is cropped to the ℓ leftmost bits of $H(X)$ for some X. Also, we denote $[H(X)]_\tau$ for the truncated hash value (of the message X) which is cropped to the τ rightmost bits of $H(X)$ for some X. The former truncated hash value is used for encryption, while the latter is utilized for the point and permute optimization of Beaver et al. [BMR90]. Note that the encryption scheme Enc is based on the encryption scheme in [LPS08] and differs from it only by utilization of group elements as keys.

Let G_a be a non-output NAND gate for some $a \in \{1, \ldots, g\}$. Let also $\mathsf{iw}_i, \mathsf{iw}_j$ be the incoming wires and ow_z be the outgoing wire of G_a where $i, j \in \{1, \ldots, M\}$ and $z \in \{1, \ldots, N\}$. To garble G_a, Party_2 prepares the following four ciphertexts

$$\mathsf{ct}_a^1 := \mathsf{Enc}_{V_i^0,V_j^0}(\overline{W_z^1}),\ \mathsf{ct}_a^2 := \mathsf{Enc}_{V_i^0,V_j^1}(\overline{W_z^1}),$$

$$\mathsf{ct}_a^3 := \mathsf{Enc}_{V_i^1,V_j^0}(\overline{W_z^1}),\ \mathsf{ct}_a^4 := \mathsf{Enc}_{V_i^1,V_j^1}(\overline{W_z^0})$$

where $\overline{W_z^1}$ and $\overline{W_z^1}$ are the ℓ-bit string representations of the group elements. Similarly, let G_b be an output NAND gate for some $b \in \{1, \ldots, g\}$. Let also $\mathsf{iw}_i, \mathsf{iw}_j$ be the incoming wires and z be the output wire index of G_b where $i, j \in \{1, \ldots, M\}$ and $z \in \{1, \ldots, o\}$. To garble G_b, Party_2 prepares the following four ciphertexts

$$\mathsf{ct}_b^1 := \mathsf{Enc}_{V_i^0,V_j^0}(y_z^1),\ \mathsf{ct}_b^2 := \mathsf{Enc}_{V_i^0,V_j^1}(y_z^1),$$

$$\mathsf{ct}_b^3 := \mathsf{Enc}_{V_i^1,V_j^0}(y_z^1),\ \mathsf{ct}_b^4 := \mathsf{Enc}_{V_i^1,V_j^1}(y_z^0).$$

For the point and permute optimization [BMR90], for each gate G_a in the circuit, Party_2 picks random indices $I_a^1, I_a^2 \in \{1, \ldots, \tau\}$ such that

$$\{(\mathsf{X}[I_a^1], \mathsf{X}[I_a^2]), (\mathsf{Y}[I_a^1], \mathsf{Y}[I_a^2]), (\mathsf{Z}[I_a^1], \mathsf{Z}[I_a^2]), (\mathsf{T}[I_a^1], \mathsf{T}[I_a^2])\} =$$

$$\{(0,0), (0,1), (1,0), (1,1)\} \text{ where}$$

$\mathsf{X} = [H(V_i^0, V_j^0, \mathsf{gateID})]_\tau$, $\mathsf{Y} = [H(V_i^0, V_j^1, \mathsf{gateID})]_\tau$, $\mathsf{Z} = [H(V_i^1, V_j^0, \mathsf{gateID})]_\tau$, $\mathsf{T} = [H(V_i^1, V_j^1, \mathsf{gateID})]_\tau$, and $\mathsf{S}[I_a^i]$ denotes the I_a^i-th bit of the bit string S. We denote each garbled gate GG_a, which is then composed of four ℓ-bit ciphertexts, ct_a^1, ct_a^2, ct_a^3, and ct_a^4, and an index pair (I_a^1, I_a^2). Note that the set

of ciphertexts in the GG_a are ordered according to I_a^1-th and I_a^2-th bits of their corresponding X, Y, Z, and T values. For example, let $\mathsf{X} = 011001\ldots1$, $\mathsf{Y} = 101111\ldots0$, $\mathsf{Z} = 110001\ldots0$, and $\mathsf{T} = 010111\ldots1$. If $(I_a^1, I_a^2) = (1, 5)$ then $(\mathsf{X}[1], \mathsf{X}[5]) = (0, 0)$, $(\mathsf{Y}[1], \mathsf{Y}[5]) = (1, 1)$, $(\mathsf{Z}[1], \mathsf{Z}[5]) = (1, 0)$, $(\mathsf{T}[1], \mathsf{T}[5]) = (0, 1)$, and therefore, we have $GG_a = (\mathsf{ct}_a^1, \mathsf{ct}_a^4, \mathsf{ct}_a^3, \mathsf{ct}_a^2, (I_a^1, I_a^2))$. A trivial method for finding such a pair (I_a^1, I_a^2) could be as follows. First, Party_2 can find I_a^1 such that $\{\mathsf{X}[I_a^1], \mathsf{Y}[I_a^1], \mathsf{Z}[I_a^1], \mathsf{T}[I_a^1]\} = \{0, 0, 1, 1\}$ with probability of $6/16$ in each trial. Then, I_a^2 could also be found with probability of $4/16$ in each trial. Therefore, the expected number of trials to find a pair of (I_a^1, I_a^2) is 7.

Party_2 garbles all the gates in the circuit in the above-mentioned way, and obtains the garbled circuit F. Party_2 then sends F and its garbled input X_2 (i.e., the W_i group elements for outgoing wires corresponding to x_2) to Party_1. As usual, Party_1 gets its own garbled input X_1 (i.e., the W_i group elements for outgoing wires corresponding to x_1) from Party_2 using oblivious transfers (OT) (or one invocation of the OT extension schemes [IKNP03, KK13, ALSZ13]). Note that this does not increase the round complexity of our overall protocol, since the exchanges needed for OTs can be jointly run with the earlier exchanged messages.

Using F, the garbled input $X = (X_1, X_2)$, T, and π_f, Party_1 evaluates the whole garbled circuit in topological order. If an outgoing wire ow_d is mapped to an incoming wire iw_e, then the group element V_e of the e-th incoming wire is computed by the multiplication of the group element W_d of the d-th outgoing wire and the blinding value t_e (i.e., if $\pi_f(d) = e$, then $V_e = t_e \cdot W_d$). Each garbled gate GG_a can be evaluated when both group elements (V_i, V_j) on its incoming wires $(\mathsf{iw}_i, \mathsf{iw}_j)$ are computed. To evaluate each GG_a, Party_1 first computes $H(V_i, V_j, \mathsf{gateID})$, and then XORs the ciphertext in the GG_a pointed by I_a^1-th and I_a^2-th bits of the $H(V_i, V_j, \mathsf{gateID})_\tau$. At the end, Party_1 obtains the token set $Y = (y_1, \ldots, y_o)$ for the output bits $f(x_1, x_2)$.

3.2 Single Execution Protocol

We now introduce an optimized implementation of our baseline scheme by carrying out some of the computations in the offline stage. In general, such precomputation techniques enhance real-time performance at the cost of extra preliminary computations and storage consumption. Besides, in today's technological perspectives, memory consumption is rarely considered to be a serious drawback since storage units are abundant in many recent devices. We provide our optimized single execution protocol with a precomputation phase in Figure 1. The computations that can be carried out in the precomputation phase include the generation of $\mathcal{P}$, and the computation of the sets $\mathcal{W}^0$ and $\mathcal{W}^1$ by Party_2.

4 Optimized Proposal with Re-execution Feature

In this section, we optimize our baseline scheme presented in Section 3 by utilizing the Reusable Mapping Template $\mathsf{ReuseTemp}_f$ when the same private function

First Protocol: Our Single Execution 2PFE Scheme

Party_1**'s Input:** $x_1 \in \{0,1\}^*$, a boolean circuit C_f consisting of NAND gates (compiled from the function f) and a mapping π_f (extracted from C_f).
Party_2**'s Input:** $x_2 \in \{0,1\}^*$.
Pre-shared Information: A group $\mathbb{G}$ of prime order q with a generator P and $\mathsf{PubInfo}_{C_f}$.
Output: $f(x_1, x_2)$.

Precomputation phase

1. Party_2 runs generates the set $\mathcal{P}$ of M random generators. It also picks $\alpha_0, \alpha_1 \in_R \mathbb{Z}_q^*$, and prepares the group element sets $\mathcal{W}^0 := (W_1^0, \ldots, W_M^0 : W_i^0 \twoheadleftarrow \alpha_0 \cdot P_i,\ i = 1, \ldots, M)$ for FALSEs and $\mathcal{W}^1 := (W_1^1, \ldots, W_M^1 : W_i^1 \twoheadleftarrow \alpha_1 \cdot P_i,\ i = 1, \ldots, M)$ for TRUEs, where P_i is the i-th element in $\mathcal{P}$ and each W_i^b is a token for $\mathsf{ow}_i \in \mathsf{OW}$, $b \in \{0,1\}$. Party_2 stores $\mathcal{P}$, $\mathcal{W}^0$, $\mathcal{W}^1$, α_0, and α_1.

Online phase

Round 1:
2. Party_2 sends $\mathcal{P}$ to Party_1.

Round 2:
3. Party_1 generates the blinding set $T := (t_1, \ldots, t_j : t_j \in_R \mathbb{Z}_q^*,\ j = 1, \ldots, N)$, computes the set $\mathcal{Q} = (Q_1, \ldots, Q_N : Q_j \leftarrow t_j \cdot P_{\pi_f^{-1}(j)},\ j = 1, \ldots, N)$. If there is a possibility of re-execution of the protocol for the same function, then Party_1 stores $\mathsf{ReuseTemp}_f := (P_1, \ldots, P_M, Q_1, \ldots, Q_N)$ (see Figure 2 for our re-execution protocol). Party_1 sends $\mathcal{Q}$ to Party_2.

Round 3:

4. Party_2 prepares the group element sets $\mathcal{V}^0 := (V_1^0, \ldots, V_N^0 : V_j^0 \twoheadleftarrow \alpha_0 \cdot Q_j,\ j = 1, \ldots, N)$ for FALSEs and $\mathcal{V}^1 := (V_1^1, \ldots, V_N^1 : V_j^1 \twoheadleftarrow \alpha_1 \cdot Q_j,\ j = 1, \ldots, N)$ for TRUEs for $\mathsf{iw}_j \in \mathsf{IW}$. If there is a possibility of re-execution of the protocol for the same function, then Party_2 stores $\mathsf{ReuseTemp}_f = (P_1, \ldots, P_M, Q_1, \ldots, Q_N)$ (see Figure 2 for the re-execution protocol). Next, Party_2 generates two random token sets for output wires of the circuit $Y^0 := (y_1^0, \ldots, y_o^0 : y_i^0 \twoheadleftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o)$ and $Y^1 := (y_1^1, \ldots, y_o^1 : y_i^1 \twoheadleftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o)$.

5. The 2PC protocol now starts from this stage where Party_2 becomes the garbler and Party_1 becomes the evaluator. Using $\mathcal{W}^0, \mathcal{W}^1, \mathcal{V}^0, \mathcal{V}^1, Y^0, Y^1$, and $\mathsf{PubInfo}_{C_f}$, Party_2 prepares the garbled circuit F by garbling each gate as follows. Party_2 prepares the following four ciphertexts to garble a non-output NAND gate G_a whose incoming wires iw_i and iw_j, and outgoing wire is ow_z: $\mathsf{Enc}_{V_i^0, V_j^0}(\overline{W_z^1})$, $\mathsf{Enc}_{V_i^0, V_j^1}(\overline{W_z^1})$, $\mathsf{Enc}_{V_i^1, V_j^0}(\overline{W_z^1})$, $\mathsf{Enc}_{V_i^1, V_j^1}(\overline{W_z^0})$. Similarly, Party_2 prepares the following four ciphertexts to garble an output NAND gate G_b whose incoming wires iw_i and iw_j, and output wire index is z: $\mathsf{Enc}_{V_i^0, V_j^0}(y_z^1)$, $\mathsf{Enc}_{V_i^0, V_j^1}(y_z^1)$, $\mathsf{Enc}_{V_i^1, V_j^0}(y_z^1)$, $\mathsf{Enc}_{V_i^1, V_j^1}(y_z^0)$. Each garbled gate GG_a is then composed of four ℓ-bit ciphertexts and two $\log_2(\tau)$-bit indices, I_a^1 and I_a^2 (see Section 3.1 for garbling details). Party_2 sends F and the garbled input X_2 for its own input x_2 to Party_1. Party_1 also obtains the garbled input X_1 for its own input x_1 from Party_2 using parallel 1-out-of-2 OTs (or a more efficient OT extension scheme).[a]

6. Using F, the garbled input $X = (X_1, X_2)$, T, and π_f, Party_1 evaluates the whole garbled circuit in topological order. If an outgoing wire ow_d is mapped to an incoming wire iw_e, then the group element V_e of the e-th incoming wire is computed by the multiplication of the group element W_d of the d-th outgoing wire with the blinding value t_e (i.e., if $\pi_f(d) = e$, then $V_e = t_e \cdot W_d$). Each garbled gate GG_a can be evaluated whenever both group elements (V_i, V_j) on its incoming wires (iw_i, iw_j) are computed. To evaluate each GG_a, Party_1 first computes $H(V_i, V_j, \mathsf{gateID})$, and then XORs the ciphertext in the GG_a pointed by I_a^1-th and I_a^2-th bits of $[H(V_i, V_j, \mathsf{gateID})]_\tau$. At the end, Party_1 obtains the token set $Y = (y_1, \ldots, y_o)$ for the output bits $f(x_1, x_2)$.

---

[a] Note that the OT protocol rounds can be combined with the former protocol rounds for minimization of the overall rounds.

Fig. 1. Our Optimized Single Execution 2PFE Protocol via decomposition of offline/online computations

is evaluated more than once. This optimization result in a more efficient protocol of our protocol shown in Figure 2. We also propose an efficient method for executions with multiple Party_2s.

4.1 Re-execution Protocol

One of the novelties of our scheme over the state-of-the-art is that our scheme results in a significant cost reduction when the same private function is evaluated more than once between the same or varying evaluating parties. This feature is quite beneficial in relevant real-life scenarios where individuals (or enterprises) can mutually and continuously have long-term business relationship instead of a single deal. Note that such a cost reduction is not available in the protocols of [KM11] and [MS13], since they require all token generation and 2PC procedures repeated in all executions. However, our scheme involves $\mathsf{ReuseTemp}_f$ that is reusable for the generation of tokens on incoming and outgoing wires. The reusability of $\mathsf{ReuseTemp}_f$ incurs a massive reduction in protocol overhead since a large part of costs in existing 2PFE protocols [KM11, MS13] results from the generation of these tokens.

With the re-execution optimization, for the k-th evaluation, Party_2 again picks $\alpha_{0,k}, \alpha_{1,k} \in_R \mathbb{Z}_q^*$ values during the precomputation phase. Party_2 then prepares the sets $\mathcal{W}_k^0$, $\mathcal{W}_k^1$, $\mathcal{V}_k^0$, and $\mathcal{V}_k^1$ as in the single execution protocol. The online phase then includes only the 2PC stage that also runs the same way as in Section 3. During the evaluation procedure of the 2PC stage, Party_1 always use the same T in all protocol runs. We provide our optimized re-execution protocol in Figure 2.

4.2 Executing with Various Party_2s

In the previous section, we addressed the case where the same two parties would like to evaluate the same function multiple times. In this section, we deal with the case that Party_1 would like to run the 2PFE protocol for the same private function with various Party_2s separately. This is a relevant scenario where Party_1 may run a business with many customers for her algorithm/software. Suppose that a cryptological research institution invents a practical algorithm for breaking RSA. Since such an algorithm would clearly attract a substantial demand, the institution may prefer preserving the details of the algorithm selling only its use. On the other hand, in many cases the clients would not like to share the keys (i.e., private inputs) with the institution. This is one of the several scenarios that a 2PFE protocol for the same private function with various Party_2s is suitable for.

First of all, we recall that the execution of our second protocol in Figure 2 requires the preknowledge of $\mathsf{ReuseTemp}_f := (\mathcal{P}, \mathcal{Q})$ by Party_2 and the set T by Party_1. Trivially, once $\mathsf{ReuseTemp}_f$ and T is produced in the first execution with any Party_2 as in our first protocol in Figure 1, then they can be stored, and our second protocol can be made use of in the subsequent executions with the same Party_2. We are here interested in a more efficient mechanism running

Second Protocol: Our k-th Re-execution 2PFE Scheme

Party_1**'s Input:** $x_{1,k} \in \{0,1\}^*$, a boolean circuit C_f consisting of NAND gates (compiled from the function f), a mapping π_f (extracted from C_f), and the blinding set $T := (t_1, \ldots, t_j : t_j \in_R \mathbb{Z}_q^*, j = 1, \ldots, N)$.
Party_2**'s Input:** $x_{2,k} \in \{0,1\}^*$.
Pre-shared Information: A cyclic group $\mathbb{G}$ of prime order q with a generator P, $\mathsf{PubInfo}_{C_f}$, and $\mathsf{ReuseTemp}_f$ [a].
Output: $f(x_{1,k}, x_{2,k})$.

Precomputation phase of the k-th execution:

1. Party_2 picks $\alpha_{0,k}, \alpha_{1,k} \in_R \mathbb{Z}_q^*$ and prepares the group element sets $\mathcal{W}_k^0 := (W_{1,k}^0, \ldots, W_{M,k}^0 : W_{i,k}^0 \leftarrow \alpha_{0,k} \cdot P_i,\ i = 1, \ldots, M)$ for FALSEs and $\mathcal{W}_k^1 := (W_{1,k}^1, \ldots, W_{M,k}^1 : W_{i,k}^1 \leftarrow \alpha_{1,k} \cdot P_i,\ i = 1, \ldots, M)$ for TRUEs for $\mathsf{ow}_i \in \mathsf{OW}$, and $\mathcal{V}_k^0 := (V_{1,k}^0, \ldots, V_{N,k}^0 : V_{j,k}^0 \leftarrow \alpha_{0,k} \cdot Q_j,\ j = 1, \ldots, N)$ for FALSEs and $\mathcal{V}_k^1 := (V_{1,k}^1, \ldots, V_{N,k}^1 : V_{j,k}^1 \leftarrow \alpha_{1,k} \cdot Q_j,\ j = 1, \ldots, N)$ for TRUEs for $\mathsf{iw}_j \in \mathsf{IW}$. Next, Party_2 generates two random token sets for output wires of the circuit $Y_k^0 := (y_{1,k}^0, \ldots, y_{o,k}^0 : y_{i,k}^0 \leftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o)$ and $Y_k^1 := (y_{1,k}^1, \ldots, y_{o,k}^1 : y_{i,k}^1 \leftarrow_R \{0,1\}^\ell,\ i = 1, \ldots, o)$.

2. The 2PC protocol now starts from this stage where Party_2 becomes the garbler and Party_1 becomes the evaluator. Using $\mathcal{W}_k^0$, $\mathcal{W}_k^1$, $\mathcal{V}_k^0$, $\mathcal{V}_k^1$, Y_k^0, Y_k^1, and $\mathsf{PubInfo}_{C_f}$, Party_2 prepares the garbled circuit F_k by garbling each gate as follows. Party_2 prepares the following four ciphertexts to garble a non-output NAND gate G_a whose incoming wires are iw_i and iw_j, and outgoing wire is ow_z: $\mathsf{Enc}_{V_{i,k}^0, V_{j,k}^0}(\overline{W_{z,k}^1})$, $\mathsf{Enc}_{V_{i,k}^0, V_{j,k}^1}(\overline{W_{z,k}^1})$, $\mathsf{Enc}_{V_{i,k}^1, V_{j,k}^0}(\overline{W_{z,k}^1})$, $\mathsf{Enc}_{V_i^1, V_j^1}(\overline{W_{z,k}^0})$. Similarly, Party_2 also prepares the following four ciphertexts to garble an output NAND gate G_b whose incoming wires are iw_i and iw_j, and output wire index is z: $\mathsf{Enc}_{V_{i,k}^0, V_{j,k}^0}(y_{z,k}^1)$, $\mathsf{Enc}_{V_{i,k}^0, V_{j,k}^1}(y_{z,k}^1)$, $\mathsf{Enc}_{V_{i,k}^1, V_{j,k}^0}(y_{z,k}^1)$, $\mathsf{Enc}_{V_i^1, V_j^1}(y_{z,k}^0)$. Each garbled gate $GG_{a,k}$ is then composed of four ℓ-bit ciphertexts and two $\log_2(\tau)$-bit bit indices, $I_{a,k}^1$ and $I_{a,k}^2$ (see Section 3.1 for garbling details). Party_2 stores F_k, $\mathcal{W}_k^0$, $\mathcal{W}_k^1$, Y_k^0, and Y_k^1.

Online phase of the k-th execution

Round 1:
3. Party_2 sends F_k and the garbled input $X_{2,k}$ for its own input $x_{2,k}$ to Party_1.

4. Party_1 gets the garbled input $X_{1,k}$ for its own input $x_{1,k}$ from Party_2 using parallel 1-out-of-2 OTs (or a more efficient OT extension scheme).

5. Using F_k, the garbled input $X_k = (X_{1,k}, X_{2,k})$, T, and π_f, Party_1 evaluates the whole garbled circuit in topological order. If an outgoing wire ow_d is mapped to an incoming wire iw_e, then the group element V_e of the e-th incoming wire is computed by the multiplication of the group element W_d of the d-th outgoing wire and the blinding value t_e (i.e., if $\pi_f(d) = e$, then $V_{e,k} = t_e \cdot W_{d,k}$). Each garbled gate $GG_{a,k}$ can be evaluated whenever both group elements $(V_{i,k}, V_{j,k})$ on its incoming wires $(\mathsf{iw}_i, \mathsf{iw}_j)$ are computed. To evaluate each $GG_{a,k}$, Party_1 first computes $H(V_{i,k}, V_{j,k}, \mathsf{gateID})$, and then XORs the ciphertext in the $GG_{a,k}$ pointed by $I_{a,k}^1$-th and $I_{a,k}^2$-th bits of $[H(V_{i,k}, V_{j,k}, \mathsf{gateID})]_\tau$. At the end, Party_1 obtains the token set $Y_k = (y_{1,k}, \ldots, y_{o,k})$ for the output bits $f(x_{1,k}, x_{2,k})$.

---

[a] $\mathsf{ReuseTemp}_f$ is already computed in the first execution of the function f as in Figure 1.

Fig. 2. Our Optimized Re-Execution 2PFE Protocol via Reusable Mapping Template

with various Party_2s by eliminating the costs of our first protocol for generating the preknowledge. The goal of this mechanism is to generate the generator set $\mathcal{P}$ in such a way that Party_1 does not know the relation between any two of its elements. T and $\mathcal{Q}$ can be subsequently computed, once the generator set $\mathcal{P}$ is given to Party_1. In order to do so, we utilize a distributed system[11] based on a t-out-of-n threshold mechanism (fault tolerant against arbitrary behaviour of up to t malicious and colluding authorities) which takes $(\mathbb{G}, q, P, M)$ as input and outputs $\mathcal{P}$.

In the offline stage of our new mechanism, the generator set $\mathcal{P}$ is generated by the distributed authorities, and given to Party_1. Next, Party_1 computes the sets T and $\mathsf{ReuseTemp}_f$. It then publishes $\mathsf{PubInfo}_{C_f}$ and $\mathsf{ReuseTemp}_f$ so that any prospective k-th party $\mathsf{Party}_{2,k}$ can utilize them in a 2PFE protocol run. This offline stage is dealt with only once, and its outputs (i.e., T and $\mathsf{ReuseTemp}_f$) are used in later re-executions. Note that the flow of re-executions for all $\mathsf{Party}_{2,k}$s is exactly the same as our second protocol in Figure 2. We would like to stress that the costs of any execution in our new mechanism with a distributed system does not differ from the second protocol.

5 Complexity Analysis

In this section, we first present the costs of our first and second protocols (given in Figure 1 and Figure 2, respectively) in terms of communication, online computation, and round complexities. We then compare these protocols with the existing boolean circuit based 2PFE schemes. For [KM11] and homomorphic encryption based protocol of [MS13], it is assumed that the elliptic curve ElGamal is used for the singly homomorphic encryption scheme (as suggested in their paper). Also, for [KM11] and our protocols, we assume that each element of $\mathbb{G}$ has a length $\ell = 2\lambda$ bits for λ-bit security.

5.1 Complexity of Our Protocols

Communication cost. In our first protocol, the overall communication overhead is $(2M + 6N)\lambda$ bits, composed of (i) the set $\mathcal{P}$ (M of 2λ-bit strings) is sent by Party_2 in Round 1, (ii) the set $\mathcal{Q}$ (N of 2λ-bit strings) is sent by Party_1 in Round 2, (iii) the garbled circuit ($2N$ of 2λ-bit strings) is sent by Party_2 in Round 3, where M is the number of outgoing wires and N is the number of incoming wires ($N = 2g$). Considering our second protocol, the use of $\mathsf{ReuseTemp}_f$ eliminates the transmission of $(2M + 2N)\lambda$ bits (required for token generation). Therefore, in total only $4N\lambda$ bits (required for the garbled circuit) are transmitted.

[11] One can also suggest a single semi-trusted authority for generation of the generator set $\mathcal{P}$. However, the knowledge of the relations among the elements of $\mathcal{P}$ by a single party may violate the privacy of inputs, and therefore, it is better to distribute the trust among multiple authorities.

Computation cost. In terms of online computation complexity, our first protocol requires $4N$ elliptic curve point multiplications, composed of (i) N operations by Party_1 in Round 2, (ii) $2N$ operations by Party_2 in Round 3, (iii) N operations by Party_1 during the evaluation of the garbled circuit. There is also a relatively small cost of $2.5N$ symmetric-key operations during the 2PC stage (composed of $2N$ operations by Party_2 for garbling and $0.5N$ operations by Party_1 for evaluating). Our second protocol reduces the online computation costs to N elliptic curve point multiplications and $0.5N$ symmetric-key operations (carried out only by Party_1). Note that Beaver's OT pre-computation technique [Bea95] can be used for decomposing OT's for Party1's input bits into online/offline stages. This eliminates online public-key operations of OT by carrying out them offline.

Number of rounds. Our first protocol has 3 rounds. The number of rounds in our second protocol is equal to 1, or 2, or 3 depending on the input string length of Party_1. Namely, if Party_1 has $x_1 = \perp$, then the number of rounds is equal to 1. If Party_1's input bits are not many, it is more efficient to use separate OTs for Party1's input tokens in parallel instead of an OT extension scheme. There exists OT schemes with 2 rounds (e.g., [Bea95] and [NP01]). Hence, this choice results in a PFE scheme with overall 2 rounds. If Party_1's input bits are many, then using an OT extension scheme is more efficient. Note that Ishai based OT extension schemes are composed of $O(\lambda)$ parallel OTs (again can be realized by Naor and Pinkas's OT [NP01]) and an additional round. Similarly, this choice results in a PFE scheme with overall 3 rounds.

5.2 Comparison

We now compare our 2PFE protocols with the state-of-the-art constant-round switching network (OSN) and homomorphic encryption (HE) based [MS13], singly homomorphic encryption based [KM11], switching network based [BBKL17], and universal circuit based [GKS17]. In our scheme, we utilize an EC cyclic group where the DDH assumption holds for state-of-the-art efficiency. For [KM11], we take into account both protocols: (1) their "$\mathcal{C}$-PFE protocol" (see [KM11, Sect. 3.1]) and (2) their "A More Efficient Variant" (see [KM11, Sect. 3.2]). For a fair comparison, we assume that the point and permute optimization [BMR90] is directly applied to the both [MS13] and [KM11] during the 2PC protocol[12]. Note that each re-execution of a private function in any of [MS13] and [KM11] has the same cost as its first execution. Also for a fair comparison, we assume that EC-ElGamal is used as HE scheme in the related protocols.

Considering communication complexity, [MS13]-OSN costs $(10N \log_2 N + 4N + 5)\lambda$ bits, composed of $(10N \log_2 N + 2N + 5)\lambda$ in OSN phase and $2N\lambda$ in the 2PC phase. The communication cost of the [KM11, Sect. 3.2] is $(4M + 6N)\lambda$ bits, including M of 4λ-bit ciphertexts sent by Party_2 in Round 1, N of 4λ-bit ciphertexts sent by Party_1 in Round 2, and $2N$ of λ-bit ciphertexts sent by Party_2 in Round 3.

[12] In [MS13] and [KM11], for the 2PC phases, the authors do not suggest any optimization. However, a point and permute optimization is available for both schemes.

Among all the state-of the art PFE protocols, our second protocol performs the best result in terms of round complexity. Namely, the number of rounds in our second protocol is equal to 1 if Party_1 has $x_1 = \perp$, or 2 if Party_1 has a non-empty input x_1 in such that the OT extension is not applicable for its garbled input, or 3 otherwise. Note that the arithmetic circuit based protocol of [MS13] provides a linear round complexity.

In terms of online computation overhead, [MS13]-OSN requires $6N \log_2 N + 2.5N + 3$ symmetric-key operations, consisting of $6N \log_2 N + 3$ operations that takes place in the OSN phase (due to the OT extension scheme on behalf of $2N \log_2 N + 1$ OTs) and $2.5N$ operations for garbling and evaluating the garbled circuit. The total online computation cost of [KM11, Sect. 3.2]is N online public key decryptions (roughly the same number of elliptic curve point multiplications) by Party_2 and relatively small additional cost of $2.5N$ symmetric-key operations for garbling and evaluating the garbled circuit. In order to compare the complexity of symmetric-key and asymmetric-key based operations, we define the computation cost ratio ρ as the cost of an elliptic curve point multiplication divided by the cost of a symmetric-key operation for the same security level. The value of ρ depends upon several factors, such as the software implementations, the symmetric-key encryption scheme, the availability of short-cut algorithms, the type of chosen elliptic curve, the hardware infrastructure, and the type of utilized processors. For example, according to [EIV18], in a setting where curve25519, and SHA256 are picked as the EC and the hash function, respectively, and the operations take place on an Intel Xeon Processor E3-1220 v6 (amd64, 4x3GHz), the value of ρ is roughly 130. Therefore, for circuits with more than 2^{20} gates, our scheme beats [MS13] also in terms of computation complexity.

For all given circuit sizes, the communication costs of our second protocol are significantly lower than that of existing 2PFE protocols. To illustrate, for a circuit with 2^{30} gates, compared to [MS13], our first and second protocols result in 97.8%, and 98.7% communication cost reduction, respectively. Moreover, compared to [KM11], for any given circuit size, our first and second protocols achieve about 12%, and 50% reduction in communication cost, respectively.

6 Security of Our Protocols

In this section, we give simulation-based security proofs of our single execution protocol in Figure 1, re-execution protocol in Figure 2, and our mechanism with various Party_2s in Sect. 4.2 in accordance with the security proof of [KM11].

Theorem 1. *If the following three conditions hold then the 2PFE protocol proposed in Figure 1 is secure against semi-honest adversaries: (1) the DDH assumption is hard in the cyclic group $\mathbb{G}$, (2) the hash-function $H : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \to \{0,1\}^{\ell+\tau}$ involved in the instantiation of DKC scheme is modeled as a random oracle, (3) the OT scheme securely realizes $\mathcal{F}_{OT}$ functionality in the OT-hybrid model against semi-honest adversaries.*

Proof. First, consider the case that Party_1 is corrupted. For any probabilistic polynomial time adversary $\mathcal{A}_1$, controlling Party_1 in the real world, we construct a simulator $\mathcal{S}_1$ that simulates $\mathcal{A}_1$'s view in the ideal world. $\mathcal{S}_1$ runs $\mathcal{A}_1$ on Party_1's inputs, f and x_1, the function output token set $Y = (y_1, \ldots, y_o)$, the pre-shared group parameters, and $\mathsf{PubInfo}_{C_f}$ as follows.

1. $\mathcal{S}_1$ generates the generator set $\tilde{\mathcal{P}} := (\tilde{P}_1, \ldots, \tilde{P}_M)$. $\mathcal{S}_1$ also prepares the group element sets $\tilde{\mathcal{W}}^0 := (\tilde{W}_1^0, \ldots, \tilde{W}_M^0 : \tilde{W}_i^0 \twoheadleftarrow \tilde{\alpha}_{0,i} \cdot P,\ \tilde{\alpha}_{0,i} \in_R \mathbb{Z}_q^*,\ i = 1, \ldots, M)$ and $\tilde{\mathcal{W}}^1 := (\tilde{W}_1^1, \ldots, \tilde{W}_M^1 : \tilde{W}_i^1 \twoheadleftarrow \tilde{\alpha}_{1,i} \cdot P,\ \tilde{\alpha}_{1,i} \in_R \mathbb{Z}_q^*,\ i = 1, \ldots, M)$. $\mathcal{S}_1$ gives $\tilde{\mathcal{P}}$ to $\mathcal{A}_1$.
2. $\mathcal{S}_1$ receives the blinding set $T := (t_1, \ldots, t_j : t_j \in_R \mathbb{Z}_q^*,\ j = 1, \ldots, N)$ from $\mathcal{A}_1$, and prepares the sets $\tilde{\mathcal{V}}^0 := (\tilde{V}_1^0, \ldots, \tilde{V}_N^0 : \tilde{V}_j^0 \twoheadleftarrow t_j \cdot \tilde{W}^0_{\pi_f^{-1}(j)},\ j = 1, \ldots, N)$ and $\tilde{\mathcal{V}}^1 := (\tilde{V}_1^1, \ldots, \tilde{V}_N^1 : \tilde{V}_j^1 \twoheadleftarrow t_j \cdot \tilde{W}^1_{\pi_f^{-1}(j)},\ j = 1, \ldots, N)$.
3. $\mathcal{S}_1$ prepares the garbled circuit $\tilde{F}$ by garbling each gate as follows. $\mathcal{S}_1$ garbles each non-output NAND gate by encrypting only the group element for FALSE on its outgoing wire with all four possible input token combinations (i.e., for a gate whose incoming wires are iw_i and iw_j, outgoing wire is ow_z, $\mathcal{S}_1$ prepares the following four ciphertexts: $\tilde{\mathsf{ct}}_a^1 = \mathsf{Enc}_{\tilde{V}_i^0, \tilde{V}_j^0}(\overline{\tilde{W}_z^0})$, $\tilde{\mathsf{ct}}_a^2 = \mathsf{Enc}_{\tilde{V}_i^0, \tilde{V}_j^1}(\overline{\tilde{W}_z^0})$, $\tilde{\mathsf{ct}}_a^3 = \mathsf{Enc}_{\tilde{V}_i^1, \tilde{V}_j^0}(\overline{\tilde{W}_z^0})$, $\tilde{\mathsf{ct}}_a^4 = \mathsf{Enc}_{\tilde{V}_i^1, \tilde{V}_j^1}(\overline{\tilde{W}_z^0})$. To garble an output NAND gate whose incoming wires are iw_i and iw_j, and output wire is z, $\mathcal{S}_1$ prepares the four ciphertexts: $\tilde{\mathsf{ct}}_b^1 = \mathsf{Enc}_{\tilde{V}_i^0, V_j^0}(y_z)$, $\tilde{\mathsf{ct}}_b^2 = \mathsf{Enc}_{\tilde{V}_i^0, \tilde{V}_j^1}(y_z)$, $\tilde{\mathsf{ct}}_b^3 = \mathsf{Enc}_{\tilde{V}_i^1, \tilde{V}_j^0}(y_z)$, $\tilde{\mathsf{ct}}_b^4 = \mathsf{Enc}_{\tilde{V}_i^1, \tilde{V}_j^1}(y_z)$. For each garbled gate $\tilde{GG}_a$, $\mathcal{S}_1$ then permutes $\tilde{\mathsf{ct}}_a^2$, $\tilde{\mathsf{ct}}_a^3$, $\tilde{\mathsf{ct}}_a^4$, and picks $\tilde{I}_a^1, \tilde{I}_a^2 \in_R \{1, \ldots, \tau\}$, and places $\tilde{\mathsf{ct}}_a^1$ in the order pointed by $\tilde{I}_a^1$-th and $\tilde{I}_a^2$-th bits of $[H(\tilde{V}_i^0, \tilde{V}_j^0, \mathsf{gateID})]_\tau$ among the other three ciphertexts. Each garbled gate $\tilde{GG}_a$ is then composed of four ℓ-bit ciphertexts and two $\log_2(\tau)$-bit random values $\tilde{I}_a^1$ and $\tilde{I}_a^2$.
4. $\mathcal{S}_1$ gives $\tilde{F}$ to $\mathcal{A}_1$ along with the simulated garbled input consisting of only the group elements for $\mathsf{FALSE}s$ on both parties' input wires $\tilde{X} = (\tilde{X}_1, \tilde{X}_2)$. This completes our simulation.

In what follows, we prove that the information obtained by Party_1 in the real execution $(\mathcal{P}, \mathcal{W}, F)$ is identically distributed to $(\tilde{\mathcal{P}}, \tilde{\mathcal{W}}, \tilde{F})$, where for outgoing wires, Party_1 obtains the group elements $\mathcal{W} = (W_1, \ldots, W_M)$ while $\mathcal{A}_1$ obtaining the group elements $\tilde{\mathcal{W}} = (\tilde{W}_1^0, \ldots, \tilde{W}_M^0)$. We now show the computational indistinguishability of $(\mathcal{P}, \mathcal{W})$ and $(\tilde{\mathcal{P}}, \tilde{\mathcal{W}})$ by utilizing Lemma 1, which ultimately ties the security of our protocol to the DDH assumption. More concretely, we need to show

$$\{(P_1, \ldots, P_M, W_1, \ldots, W_M)\} \approx_c \{(\tilde{P}_1, \ldots, \tilde{P}_M, \tilde{W}_1^0, \ldots, \tilde{W}_M^0)\}$$

$$\{(r_1 \cdot P, \ldots, r_M \cdot P, \alpha_{\mathsf{b}_1} \cdot (r_1 \cdot P), \ldots, \alpha_{\mathsf{b}_M} \cdot (r_M \cdot P))\} \approx_c \{(\tilde{r}_1 \cdot P, \ldots, \tilde{r}_M \cdot P, \tilde{\alpha}_{0,1} \cdot P, \ldots, \tilde{\alpha}_{0,M} \cdot P)\}$$

where $\mathsf{b}_i \in \{0, 1\}$ is the semantic value on ow_i and $\tilde{P}_i = \tilde{r}_i \cdot P$. For the sake of a simpler representation, we replace $\alpha_{\mathsf{b}_i} r_i$ with r_{M+i}, and $\tilde{\alpha}_{0,i}$ with $\tilde{r}_{M+i}$ for $i =$

$1, \ldots, M$. Note that $(r_1, \ldots, r_{2M})$ is not identically distributed to $(\tilde{r}_1, \ldots, \tilde{r}_{2M})$, while it is only sufficient to show that

$$\{(r_1 \cdot P, \ldots, r_{2M} \cdot P)\} \approx_c \{(\tilde{r}_1 \cdot P, \ldots, \tilde{r}_{2M} \cdot P)\}.$$

For this purpose, we generate a new set $\mathcal{R} := (R_1, \ldots, R_{2M})$ by picking $2M$ random generators. Hence, we now need to show

$$\{(R_1, \ldots, R_{2M}, r_1 \cdot P, \ldots, r_{2M} \cdot P)\} \approx_c \{(R_1, \ldots, R_{2M}, \tilde{r}_1 \cdot P, \ldots, \tilde{r}_{2M} \cdot P)\}$$

Thanks to Lemma 1 and the underlying DDH assumption, we have both

$$\{(R_1, \ldots, R_{2M}, \gamma \cdot R_1, \ldots, \gamma \cdot R_{2M})\} \approx_c \{(R_1, \ldots, R_{2M}, r_1 \cdot P, \ldots, r_{2M} \cdot P)\}$$

and

$$\{(R_1, \ldots, R_{2M}, \gamma \cdot R_1, \ldots, \gamma \cdot R_{2M})\} \approx_c \{(R_1, \ldots, R_{2M}, \tilde{r}_1 \cdot P, \ldots, \tilde{r}_{2M} \cdot P)\}$$

where $\gamma \in_R \mathbb{Z}_q^*$. Hence, the following sets are computationally indistinguishable

$$\{(r_1 \cdot P, \ldots, r_{2M} \cdot P)\} \approx_c \{(\tilde{r}_1 \cdot P, \ldots, \tilde{r}_{2M} \cdot P)\}$$

which effectively concludes the proof for $\{(\mathcal{P}, \mathcal{W})\} \approx_c \{(\tilde{\mathcal{P}}, \tilde{\mathcal{W}})\}$. Furthermore, since the same values in T are used among the outgoing wire tokens and incoming wire tokens in both the real and the ideal executions, we have $\{(\mathcal{P}, \mathcal{W}, \mathcal{V})\} \approx_c \{(\tilde{\mathcal{P}}, \tilde{\mathcal{W}}, \tilde{\mathcal{V}})\}$ where for each incoming wire $\mathcal{V} = (V_1, \ldots, V_N)$ is the set of tokens obtained by Party_1 and $\tilde{\mathcal{V}} = (\tilde{V}_1^0, \ldots, \tilde{V}_N^0)$ is the set of tokens obtained by $\mathcal{A}_1$. In contrast to [KM11], it is relatively simple to prove the computational indistinguishability of F and $\tilde{F}$ in our scheme since we use a hash function modeled as random oracle during garbling. Once the distribution of four hash outputs for each gate (in the real and ideal executions) are proven to be computationally indistinguishable random values, outputs of our instantiation of DKC is also proven to be computationally indistinguishable. This results in the computational indistinguishability of each garbled gate GG_a and $\tilde{GG}_a$, and eventually computational indistinguishability of F and $\tilde{F}$. For a gate whose incoming wires are iw_i and iw_j, in the real execution, we have four hash outputs involved in the garbling

$$H(V_i^0, V_j^0, \mathsf{gateID}), H(V_i^0, V_j^1, \mathsf{gateID}), H(V_i^1, V_j^0, \mathsf{gateID}), H(V_i^1, V_j^1, \mathsf{gateID}).$$

Similarly, for each gate, in the ideal execution, we have the following four hash outputs in the garbling

$$H(\tilde{V}_i^0, \tilde{V}_j^0, \mathsf{gateID}), H(\tilde{V}_i^0, \tilde{V}_j^1, \mathsf{gateID}), H(\tilde{V}_i^1, \tilde{V}_j^0, \mathsf{gateID}), H(\tilde{V}_i^1, \tilde{V}_j^1, \mathsf{gateID}).$$

Since in Party_1's view, resulting from the indistinguishability of $\mathcal{V}$ and $\tilde{\mathcal{V}}$, the hash inputs are computationally indistinguishable, and therefore, the hash outputs are computationally indistinguishable random values. This completes the proof for $\{(\mathcal{P}, \mathcal{W}, F)\} \approx_c \{(\tilde{\mathcal{P}}, \tilde{\mathcal{W}}, \tilde{F})\}$.

We now consider the case that Party_2 is corrupted. For any probabilistic polynomial-time adversary $\mathcal{A}_2$, controlling Party_2 during our first protocol in the real world, we construct a simulator $\mathcal{S}_2$ that simulates $\mathcal{A}_2$'s view in the ideal world. $\mathcal{S}_2$ runs $\mathcal{A}_2$ on Party_2's input, and the pre-shared group parameters, and $\mathsf{PubInfo}_{C_f}$ as follows.

1. $\mathcal{S}_2$ asks $\mathcal{A}_2$ to generate $\tilde{\mathcal{P}} \leftarrow \textsc{Init}(\mathbb{G}, q, P, M)$ and receives $\tilde{\mathcal{P}}$.
2. $\mathcal{S}_2$ then picks $\tilde{t}_j \in_R \mathbb{Z}_q^*$ for $j = 1, \ldots, N$, and computes $\tilde{Q}_j \leftarrow \tilde{t}_j \cdot P$ which are now random group elements in $\mathbb{G}$. $\mathcal{S}_2$ assigns $\tilde{\mathcal{Q}} = (\tilde{Q}_1, \ldots, \tilde{Q}_N)$, and gives $\tilde{\mathcal{Q}}$ to $\mathcal{A}_2$. This completes our simulation.

In the real execution of our protocol, Party_2 receives only the message $\mathcal{Q} := (Q_1, \ldots, Q_N : Q_j \leftarrow t_j \cdot P_{\pi_f^{-1}(j)}, j = 1, \ldots, N)$ in Round 2 (apart from the exchanged messages during the OT protocol for Party_1's garbled input). However, the transcripts received by Party_2 during the OT do not leak any information to Party_2 due to the ideal execution $\mathcal{F}_{OT}$ in the OT-hybrid model. Due to DDH assumption, in Party_2's view, the distributions of $\tilde{\mathcal{Q}}$ and $\mathcal{Q}$ are identical (i.e., $\tilde{\mathcal{Q}} \approx_c \mathcal{Q}$). This concludes the proof for the single execution protocol. □

Theorem 2. *If the 2PFE protocol proposed in Figure 1 is secure against semi-honest adversaries (i.e., the three conditions in Theorem 1 are satisfied), then the 2PFE protocol proposed in Figure 2 is also secure against semi-honest adversaries.*

Proof (Sketch). The main difference of the re-execution protocol from the first one is the utilization of $\mathsf{ReuseTemp}_f$. Therefore, the proof will be complete once we show that the utilization of the sets $\mathcal{W}_k^0$, $\mathcal{W}_k^1$, $\mathcal{V}_k^0$, and $\mathcal{W}_k^1$ computed from the same $\mathsf{ReuseTemp}_f$ in the k-th execution gives Party_1 no advantage in deducing Party_2's inputs.

We now show that in Party_1's view, $(\mathcal{W}_k, \mathcal{V}_k, \mathcal{W}_{k+1}, \mathcal{V}_{k+1})$ in two consecutive real executions are computationally indistinguishable from $(\tilde{\mathcal{W}}_1, \tilde{\mathcal{V}}_1, \tilde{\mathcal{W}}_2, \tilde{\mathcal{V}}_2)$ where $\tilde{\mathcal{W}}_1 := (\tilde{W}_{1,1}, \ldots, \tilde{W}_{M,1} : \tilde{W}_{i,1} = \tilde{q}_{i,1} \cdot P,\ \tilde{q}_{i,1} \in_R \mathbb{Z}_q^*,\ i = 1, \ldots, M)$, $\tilde{\mathcal{V}}_1 := (\tilde{V}_{1,1}, \ldots, \tilde{V}_{N,1} : \tilde{V}_{j,1} \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j),1},\ j = 1, \ldots, N)$, $\tilde{\mathcal{W}}_2 := (\tilde{W}_{1,2}, \ldots, \tilde{W}_{M,2} : \tilde{W}_{i,2} = \tilde{q}_{i,2} \cdot P,\ \tilde{q}_{i,2} \in_R \mathbb{Z}_q^*,\ i = 1, \ldots, M)$, and $\tilde{\mathcal{V}}_2 := (\tilde{V}_{1,2}, \ldots, \tilde{V}_{N,2} : \tilde{V}_{j,2} \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j),2},\ j = 1, \ldots, N)$. More concretely, we have

$$\{(\overline{(1,k)}, \ldots, \overline{(M,k)}, t_1 \cdot \overline{(\pi_f^{-1}(1), k)}, \ldots, t_N \cdot \overline{(\pi_f^{-1}(N), k)}, \overline{(1,k+1)}, \ldots, \overline{(M,k+1)},$$

$$t_1 \cdot \overline{(\pi_f^{-1}(1), k+1)}, \ldots, t_N \cdot \overline{(\pi_f^{-1}(N), k+1)})\} \approx_c \{(\tilde{q}_{1,1} \cdot P, \ldots, \tilde{q}_{M,1} \cdot P, t_1 \cdot (\tilde{q}_{\pi_f^{-1}(1),1} \cdot P),$$

$$\ldots, t_N \cdot (\tilde{q}_{\pi_f^{-1}(N),1} \cdot P), \tilde{q}_{1,2} \cdot P, \ldots, \tilde{q}_{M,2} \cdot P, t_1 \cdot (\tilde{q}_{\pi_f^{-1}(1),2} \cdot P), \ldots, t_N \cdot (\tilde{q}_{\pi_f^{-1}(N),2} \cdot P))\}$$

where $\overline{(i,j)}$ is the abbreviation for $\alpha_{\mathsf{b}_{i,j},j} \cdot P_i$, and $\mathsf{b}_{i,k} \in \{0,1\}$ is the semantic bit value of ow_i in the k-th execution. The proof of their indistinguishability relies on the same flow as the proof of Theorem 1, which depends on Lemma 1 and ultimately on the DDH assumption. □

Theorem 3. *If the threshold system is secure against malicious adversaries at most $t-1$ of whom are allowed to collude, and the 2PFE protocol proposed in Figure 2 is secure against semi-honest adversaries; then our mechanism with various* Party_2*s in Sect. 4.2 is also secure against semi-honest adversaries.*

Proof (Sketch). First, the Party_1's view in the 2PFE mechanism is equivalent to the one in the protocol in Figure 2. Observe that the generator set is generated by the distributed system and the tokens (that are used in preparation of the garbled input X_k and the garbled circuit F_k) are computed from $\alpha_{0,k}$ or $\alpha_{1,k}$ in each evaluation as in Figure 2. Therefore, the 2PFE mechanism prevents Party_1 from deducing any information about $\mathsf{Party}_{2,k}$'s input.

Second, $\mathsf{Party}_{2,k}$s cannot obtain any information about Party_1's input in none of the executions since the OT outputs are only obtained by Party_1 due the $\mathcal{F}_{OT}$ functionality in the OT-hybrid model. Also, due to Theorem 1, no one can obtain information about π_f from the $\mathsf{ReuseTemp}_f$. Moreover, any $\mathsf{Party}_{2,k}$ has a negligible advantage on distinguishing the exchanged messages in an evaluation between Party_1 and $\mathsf{Party}_{2,l}$ from a random string due to the underlying DDH assumption for $l \neq k$. More concretely, the tokens (that are used in preparation of $\mathsf{Party}_{2,l}$'s garbled input $X_{2,l}$ and the garbled circuit F_l) are computed by multiplying the elements of the $\mathsf{ReuseTemp}_f$ with the private values $\alpha_{0,l}$ or $\alpha_{1,l}$ of $\mathsf{Party}_{2,l}$.

□

References

AF90. Martín Abadi and Joan Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, Feb 1990.

AFK87. M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 195–203. ACM, 1987.

ALSZ13. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 535–548, New York, NY, USA, 2013. ACM.

BBKL17. Osman Biçer, Muhammed Ali Bingöl, Mehmet Sabır Kiraz, and Albert Levi. Towards Practical PFE: An Efficient 2-Party Private Function Evaluation Protocol Based on Half Gates. Cryptology ePrint Archive, Report 2017/415, 2017. `https://eprint.iacr.org/2017/415`.

Bea95. Donald Beaver. Precomputing Oblivious Transfer. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 97–109, London, UK, UK, 1995. Springer-Verlag.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of Garbled Circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796, New York, NY, USA, 2012. ACM.

BMR90. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 503–513. ACM, 1990.

Bon98. Dan Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

CFA+12. Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2012.

CKMZ14. Seung Geol Choi, Jonathan Katz, Alex J. Malozemoff, and Vassilis Zikas. Efficient three-party computation from cut-and-choose. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 513–530, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

EG85. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

EIV18. ECRYPT-II and Vampire. eBACS: ECRYPT Benchmarking of Cryptographic Systems, 2018. `http://http://bench.cr.yp.to/` (accessed on 2018-01-22).

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

Gir16. Damien Giry. Keylength – cryptographic key length recommendation, 2016. `http://www.keylength.com/` (accessed on 2017-05-17).

GKS17. Daniel Günther, Ágnes Kiss, and Thomas Schneider. More Efficient Universal Circuit Constructions. In *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*, pages 443–470, 2017.

HMV03. Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

HS15. Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 641–670, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer Berlin Heidelberg, 2003.

KK13. Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 54–70, 2013.

KKS16. Carmen Kempka, Ryo Kikuchi, and Koutarou Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In *Advances in Cryptology – ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 967–997, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

KM11. Jonathan Katz and Lior Malka. Constant-round private function evaluation with linear complexity. In *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*, pages 556–571, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

KMR14. V. Kolesnikov, P. Mohassel, and M. Rosulek. Flexor: Flexible garbling for xor gates that beats free-xor. In *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 440–457, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

KS08a. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II*, pages 486–498, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

KS08b. Vladimir Kolesnikov and Thomas Schneider. A practical universal circuit construction and secure evaluation of private functions. In *Financial Cryptography and Data Security: 12th International Conference, FC 2008, Cozumel, Mexico, January 28-31, 2008. Revised Selected Papers*, pages 83–97, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

KS16. Ágnes Kiss and Thomas Schneider. Valiant's universal circuit is practical. In *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*, pages 699–728, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

LPS08. Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks: 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, pages 2–20, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

LV01. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

MS13. Payman Mohassel and Saeed Sadeghian. How to hide circuits in mpc an efficient framework for private function evaluation. In *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 557–574. Springer Berlin Heidelberg, 2013.

MSS14. Payman Mohassel, Saeed Sadeghian, and Nigel P. Smart. Actively secure private function evaluation. In *Advances in Cryptology ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 486–505. Springer Berlin Heidelberg, 2014.

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 448–457, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

NPS99. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.

NR04. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, March 2004.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

PSS09. Annika Paus, Ahmad-Reza Sadeghi, and Thomas Schneider. Practical secure evaluation of semi-private functions. In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 89–106, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

PSSW09. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 250–267, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

Sad15. SeyedSaeed Sadeghian. *New Techniques for Private Function Evaluation.* PhD thesis, University of Calgary, 2015.

Sil09. J.H. Silverman. *The Arithmetic of Elliptic Curves.* Graduate Texts in Mathematics. Springer New York, 2009.

SS09. Ahmad-Reza Sadeghi and Thomas Schneider. Generalized universal circuits for secure evaluation of private functions with application to data classification. In *Information Security and Cryptology – ICISC 2008: 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*, pages 336–353, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

SZ13. Thomas Schneider and Michael Zohner. Gmw vs. yao? efficient secure two-party computation with low depth circuits. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 275–292, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Yao82. Andrew C. Yao. Protocols for Secure Computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

Yao86. A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer Berlin Heidelberg, 2015.