

# An Efficient 2-Party Private Function Evaluation Protocol Based on Half Gates

Muhammed Ali Bingöl<sup>1,2</sup>, Osman Biçer<sup>3</sup>, Mehmet Sabır Kiraz<sup>4</sup>, Albert Levi<sup>1</sup>

<sup>1</sup> Sabancı University, Faculty of Engineering and Natural Sciences, İstanbul, Turkey

<sup>2</sup> TUBITAK, BILGEM, Kocaeli

<sup>3</sup> Koç University, İstanbul, Turkey

<sup>4</sup> Cyber Technology Institute, De Montfort University, UK

mabingol@sabanciuniv.edu, obicer17@ku.edu.tr, mehmet.kiraz@dmu.ac.uk, levi@sabanciuniv.edu

**Abstract.** Private function evaluation (PFE) is a special case of secure multi-party computation (MPC), where the function to be computed is known by only one party. PFE is useful in several real-life applications where an algorithm or a function itself needs to remain secret for reasons such as protecting intellectual property or security classification level. In this paper, we focus on improving 2-party PFE based on symmetric cryptographic primitives. In this respect, we look back at the seminal PFE framework presented by Mohassel and Sadeghian at Eurocrypt’13. We show how to adapt and utilize the well-known *half gates* garbling technique (Zahur *et al.*, Eurocrypt’15) to their constant round 2-party PFE scheme. Compared to their scheme, our resulting optimization significantly improves the efficiency of both the underlying *Oblivious Evaluation of Extended Permutation* (OEP) and secure 2-party computation (2PC) protocols, and yields a more than 40% reduction in overall communication cost (the computation time is also slightly decreased, and the number of rounds remains unchanged).

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

## 1 Introduction

Imagine that one invents a novel and practical algorithm capable of being directly used to detect and identify criminals in crowds with a high degree of precision based on information about their behaviors obtained from street video recordings. It is obvious that this algorithm would be commercially valuable and that many governmental organizations would like to use it. The inventor has the right to keep the algorithm confidential, and to offer only its use for a certain fee since it is his/her own intellectual property. On the other hand, governmental organizations will generally be unwilling to reveal their records and databases to the parties to whom they do not sufficiently trust. This is an example of the problem that two parties would like to execute a common function with their private inputs and the function is also a private input of one of the parties. Solution for this and such real-life problems are addressed by *Private Function Evaluation* (PFE).

PFE is a special case of secure multi-party computation (MPC) in which  $n$  participants jointly compute a function  $f$  on their private inputs  $x_1, \dots, x_n$ , and one (or some) of the parties obtain the result  $f(x_1, \dots, x_n)$  while revealing nothing more to the parties. The difference of PFE from the standard MPC setting is that here the function  $f$  is also a private input of one of the participants<sup>5</sup>. A PFE solution would be more useful than conventional MPC in various real-life applications, *e.g.*, the ones where the function itself contains private information, or reveals security weaknesses; or the ones where service providers prefer hiding their function, or its specific implementation as their intellectual property. Efficient and practical PFE schemes are becoming increasingly important as many applications require protection of their valuable assets such as private database management systems [PKV<sup>+</sup>14], privacy-preserving intrusion detection system [NSMS14], privacy-preserving checking for credit worthiness [FAZ05] and privacy preserving medical

<sup>5</sup> Note that PFE also covers the case where the party who owns the function does not have any other private input.

applications [BFK<sup>+</sup>09]. Therefore, the task of designing efficient custom PFE protocols for special or generic purposes is addressed in several papers in literature [KS08b, SS09, BFK<sup>+</sup>09, PSS09, KM11, MS13].

Generic PFE solutions are mainly classified into two categories. The first one is the *universal circuit* [Val76] based approach that works with any MPC protocol. The ideal functionality of MPC  $\mathcal{F}_{U_g}$  for a universal circuit  $U_g$  takes as input a certain sized ( $g$ ) boolean circuit representation  $C_f$  of the private function  $f$ , and inputs of parties  $x_1, \dots, x_n$  (i.e.,  $\mathcal{F}_{U_g}(C_f, x_1, \dots, x_n)$ ), and outputs  $f(x_1, \dots, x_n)$ . The works based on this approach mainly aim to reduce the size of universal circuits, and to optimize their implementations using some MPC techniques [KS08b, SS09, KS16, LMS16, GKS17]. The early universal circuit based schemes result in massive circuit sizes [Val76, KS08b, SS09, Sch08], which was the root cause of their inefficiency. By the recent works [KS16] and [GKS17] the universal circuits becomes more practical but the computation cost is still worse than the custom PFE protocols of [MS13, Sad15].

The second approach is falls into designing custom PFE protocols which avoids the use of universal circuits. Following this line of work, several PFE schemes have been proposed [PSS09, KM11, MS13, MSS14, Sad15, BBKL18]. An early attempt on this category is Paus, Sadeghi, and Schneider’s work [PSS09]. They introduce -what they called- a *semi-private function evaluation* in which the type of the gates is a secret of one party, but the circuit topology (i.e., the set of all connections of predecessors and successors of each gate) is public to both parties. Due to the weaker assumption of semi-privacy, their approach does not provide a complete PFE solution. Another significant improvement in this category comes from Katz and Malka’s 2-party PFE (2-PFE) scheme [KM11] with a mechanism for hiding the circuit topology based on asymmetric cryptography primitives (i.e., partially (singly) *homomorphic encryption* (HE) e.g., ElGamal [ElG85] or Paillier [Pai99]). Following the line of [KM11], recently [BBKL18] proposed 2-PFE protocols based on asymmetric cryptography primitives. In this work, they propose a secure 2-PFE scheme based on Decisional Diffie-Hellman (DDH) assumption. Their approach introduces a reusability feature that significantly improves the state-of-the-art.

In [MS13], Mohassel and Sadeghian come up with a framework for PFE that includes several schemes for different settings. They proposed protocols for both arithmetic and boolean circuits. Their protocol for arithmetic circuits (based on partially HE) has a number of rounds equal to the number of gates (see [MS13, p. 570]), whereas the other PFE protocols for boolean circuits have constant number of rounds. Regarding their arithmetic based protocol, for large circuits, the number of rounds is a bottleneck. For boolean circuits, they propose two types of protocols: one is based on partially HE and the other one is based on *oblivious evaluation of switching networks* (OSN). The OSN based protocol of [MS13] is (mostly)<sup>6</sup> based on symmetric cryptographic primitives.

The existing schemes based on asymmetric cryptographic primitives such as [KM11, BBKL18] and partially HE based protocol of [MS13] are promising in terms of linear communication complexity. However, for some applications, protocols primarily based on symmetric cryptography could be favorable.

Considering OSN based 2-PFE scheme of [MS13], they split the PFE task into two sub-functionalities: (1) Circuit topology hiding (CTH), (2) Private gate evaluation (PGE). Briefly speaking, in CTH, a series of procedures is performed: First, the function owner (say  $P_1$ ) detaches the interconnections of the gates to obtain single gates, and keeps the topological mapping of the circuit private. Second,  $P_1$  and the other party (say  $P_2$ ) engage in an *oblivious evaluation of switching network* (OSN)<sup>7</sup> protocol which consists of  $O(g \log(g))$  *oblivious transfer* (OT) operations (throughout this paper,  $g$  denotes the number of gates, and  $\log()$  denotes the logarithm base 2). Next, in PGE, both parties engage in a Yao’s 2-party computation (2PC) protocol [Yao82, LP09] where  $P_1$  and  $P_2$  play the *evaluator* and the *garbler* roles, respectively. Each single gate is garbled into four ciphertexts. By setting all gates as a single gate type (e.g., NAND or NOR), it is possible to avoid the necessity of hiding the gate functionality [MS13].

<sup>6</sup> The only asymmetric cryptographic structure is due to the OT operations of underlying 2PC, therefore can be considered as symmetric based [Sad15, GKS17].

<sup>7</sup> The OSN mechanism is introduced in [MS13] to achieve a solution for the oblivious evaluation of extended permutation (OEP) problem. OEP allows the oblivious transition of each masked gate output to the input(s) of the next connected gate(s).

Recently, in [WmM17], Wang and Malluhi attempt to improve the 2-PFE scheme of Mohassel and Sadeghian by removing only one ciphertext from each garbled gate in the 2PC phase. However, the communication cost of the 2PC phase is quite lower than that of the OSN phase, which means that their scheme reduces the overall cost by less than 1%.

In [Sad15, p. 98] and [LMS16, p. 2] the authors mention that “*the various optimizations that are recently proposed for MPC [KS08a, KMR14, ZRE15] are making general 2PC more practical and it is not obvious if their techniques can also be combined with custom PFE solutions (which remains as an interesting open question)*”. One of the aims of this work is providing an answer to this open question and come-up with an efficient 2-PFE protocol.

*Our contributions.* In this paper, we mainly focus on improving 2-party private function evaluation (2-PFE) based on symmetric cryptographic primitives. In this respect, we first revisit the state-of-the-art Mohassel and Sadeghian’s PFE framework [MS13], then propose a more efficient protocol (secure in the presence of semi-honest adversaries) by adapting<sup>8</sup> the half gates garbling optimization [ZRE15] to their 2-PFE scheme. Our protocol achieves the following significant improvements in both OSN and 2PC phases:

1. Regarding the OSN phase: (1) We reduce the number of required OTs by  $N = 2g$ . Concretely, the technique in [MS13] requires  $2N \log(N) + 1$  OTs, while our protocol requires  $2N \log(N) - N + 1$  OTs. (2) Our protocol reduces the data sizes entering to the OSN protocol by a factor of two. This improvement results in about 40% saving.
2. Regarding the 2PC phase, our scheme garbles each non-output gate (that does not have any direct connection with output wires of the circuit) with only three ciphertexts, and each output gate with only two ciphertexts.

Among the above improvements, the foremost gain comes from the reduction in the input sizes of the OSN protocol. The overall communication cost of our scheme is  $(6N \log(N) + 0.5N + 3)\lambda$  bits<sup>9</sup>, which is a significant improvement compared to [MS13], whose communication cost is  $(10N \log(N) + 4N + 5)\lambda$  bits. This means more than 40% saving in bandwidth size (see Table 5 and Table 6). Also the overall computation cost is also slightly decreased while the number of rounds remains unchanged.

*Organization* In Section 2, we give preliminary information about oblivious transfer, Yao’s garbled circuits, and half gates optimization. In Section 3, we present the 2-PFE framework and scheme of [MS13] in detail. In Section 4, we introduce our 2-PFE scheme. Section 5 provides a simulation based security proof of our 2-PFE scheme in the semi-honest model. In Section 6, we analyze our protocol in terms of communication and computation complexities and compare it with 2-PFE scheme in [MS13]. Finally, Section 7 concludes the paper and point out some future works.

## 2 Preliminaries

This section provides some background information on oblivious transfer, Yao’s garbled circuits, and the state-of-the-art half gates optimization.

### 2.1 Oblivious transfer (OT)

A *k-out-of-m oblivious transfer* protocol is a two-party protocol where one of the parties is the sender (S) who has set of values  $\{x_1, \dots, x_m\}$ , and the other one is the receiver (R) who has  $k$  selection indices. At the

<sup>8</sup> Note that in [WmM17], Wang and Malluhi mention that free-XOR [KS08a] and half gates [ZRE15] techniques cannot be used to improve the efficiency of non-universal circuit based custom PFE protocols such as Katz and Malka’s [KM11] and Mohassel and Sadeghian’s [MS13] works. In contrast to their claim, we adapt and utilize half gates approach to Mohassel and Sadeghian’s and reduce the communication cost in a secure way.

<sup>9</sup>  $\lambda$  is the security parameter throughout this paper.

end of the protocol, R only learns  $k$  of the S's inputs according to his selection indices; whereas S learns nothing. In the OT-hybrid model, the two parties are given access to the ideal OT functionality ( $\mathcal{F}_{OT}$ ) which implies a universally composable OT protocol. Oblivious transfer is a critical underlying protocol used in many MPC constructions [Rab81, EGL85].

OT extension is a way of obtaining many OTs from a few number of OT runs and cheap symmetric cryptographic operations. Ishai *et al.* constructed the first OT extension method [IKNP03], which reduces a given large number of required OTs to a fixed size security parameter. Later, several OT extension schemes based on [IKNP03] are proposed for improving the efficiency [KK13, ALSZ13].

**Table 1.** Garbling an odd gate using half gates technique [ZRE15].

<b>Garbler half gate (<math>p_b</math> known to the garbler)</b>	<b>Evaluator half gate (<math>p_b \oplus v_b</math> known to the evaluator)</b>
Defines the half gate:	Defines the half gate:
$f_G(v_a, p_b) := (\alpha_1 \oplus v_a)(\alpha_2 \oplus p_b) \oplus \alpha_3$	$f_E(v_a, v_b \oplus p_b) := (\alpha_1 \oplus v_a)(p_b \oplus v_b)$
Computes:	Computes:
$T_{Gc} \leftarrow H(w_a^0) \oplus H(w_a^1) \oplus (p_b \oplus \alpha_2)R$	$T_{Ec} \leftarrow H(w_b^0) \oplus H(w_b^1) \oplus w_a^{\alpha_1}$
$w_{Gc}^0 \leftarrow H(w_a^{p_a}) \oplus f_G(p_a, p_b)R$	$w_{Ec}^0 \leftarrow H(w_b^{p_b})$
<u>The garbler sends <math>T_{Gc}</math>.</u>	<u>The garbler sends <math>T_{Ec}</math>.</u>

## 2.2 Yao's protocol

Yao's protocol is essentially a 2PC protocol secure in the semi-honest adversary model. It allows two parties, the *garbler* and the *evaluator*, to evaluate an arbitrary polynomial-sized function  $f(x) = f(x_1, x_2)$ , where  $x_1$  is the garbler's private input and  $x_2$  is the evaluator's private input, without leaking any information about their private inputs to each other beyond what is implied by the pure knowledge of the function output. The main idea is that the garbler prepares an *encrypted* version of  $C_f$  (a boolean circuit representation of  $f$ ). This encrypted version is called the garbled circuit  $\hat{F}$  and sent to the evaluator. The evaluator then computes the output from the garbled version of the circuit without obtaining the garbler's input bits or intermediate values.

Recently, several major optimizations proposed for Yao's protocol, mainly aiming at bandwidth efficiency and or reduction of the garbling and evaluating costs (*e.g.*, point and permute [BMR90], garbled row reduction 3 ciphertexts (GRR3) [NPS99], free-XOR [KS08a], garbled row reduction 2 ciphertexts (GRR2) [PSSW09], pipelining [HEKM11], fleXOR [KMR14], half gates [ZRE15], and garbling gadgets [BMR16]). With the recent optimizations, Yao's protocol has now impressive results from the complexity point of view.

## 2.3 Half gates technique

In [ZRE15], Zahur, Rosulek, and Evans propose an elegant and efficient garbling scheme called *half gates* technique. Their garbling technique is currently known the most efficient optimization in terms of communication complexity compared to any prior scheme. This technique remains compatible with free-XOR [KS08a] while also reducing the ciphertext requirement for each odd gate<sup>10</sup> to two. Here, we briefly describe the garbling procedure of odd gates using the half gates technique, and refer the reader to [ZRE15] for further details and its security proof.

<sup>10</sup> *Odd* and *Even* gates are fan-in-two logic gates. The former has an odd number of TRUE outputs in its truth table; while the latter has an even number of those.

Any odd gate type can be written in the form of Equation (2.1) where  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$  define the gate type, *e.g.*, setting  $\alpha_1 = 0$ ,  $\alpha_2 = 0$ ,  $\alpha_3 = 1$  results in a NAND gate [ZRE15]. Let  $v_i$  denote the one bit truth value on the  $i^{\text{th}}$  wire in a circuit.

$$f_{G_{\text{odd}}}(v_a, v_b) \rightarrow (\alpha_1 \oplus v_a) \wedge (\alpha_2 \oplus v_b) \oplus \alpha_3 \quad (2.1)$$

The garbler garbles an odd gate by following the steps for both half gates in Table 1. The tokens for FALSE and TRUE on the  $i^{\text{th}}$  wire are denoted as  $w_i^0$  and  $w_i$ , respectively. The global free-XOR offset is denoted as  $R$ .

The garbler sets  $R \leftarrow \{0, 1\}^{\lambda-1}1$  globally, and  $w_i^0 \leftarrow \{0, 1\}^\lambda$  and  $w_i^1 \leftarrow w_i^0 \oplus R$  for each wire. We have  $\text{lsb}(R) = 1$  so that  $\text{lsb}(w_i^0) \neq \text{lsb}(w_i^1)$ .  $w_{G_c}^b$  and  $w_{E_c}^b$  denote the tokens for the garbler and the evaluator half gate outputs for truth value  $b$ , respectively.  $T_{G_c}$  and  $T_{E_c}$  denote the  $\lambda$ -bit strings needing to be sent for the garbler and evaluator half gates, respectively. Let  $w_i$  be a token on  $i^{\text{th}}$  wire obtained by the evaluator who does not know its corresponding truth value  $v_i$ . For the  $i^{\text{th}}$  wire, let  $p_i := \text{lsb}(w_i^0)$ , a value only known to the garbler. If two symbols are appended, an AND operation is implied, *i.e.*,  $ab = a \wedge b$ .  $H : \{0, 1\}^\lambda \times \mathbb{Z} \rightarrow \{0, 1\}^\lambda$  denotes a hash function with circular correlation robustness for naturally derived keys<sup>11</sup>, having the security parameter  $\lambda$ .

The token on the output wire of the odd gate for FALSE is  $w_{G_c}^0 \oplus w_{E_c}^0$  since the output of the odd gate is an XOR of half gate outputs. The two ciphertexts computed  $T_{G_c}$  and  $T_{E_c}$  are needed to be sent to the evaluator for each gate.

### 3 2-Party PFE Framework

In [MS13], Mohassel and Sadeghian introduce a generic PFE framework for boolean and arithmetic circuits. In this work, our focus is mainly on private function evaluation based on boolean circuits in 2-party setting *i.e.*, 2-PFE. In order to achieve 2-PFE, Mohassel and Sadeghian show that hiding (i) the parties' private inputs, (ii) the topology of the circuit representation  $C_f$ , and (iii) the functionality of its gates is required. The framework is not concerned with hiding the numbers of gates, input/output wires and the type of the gates of the circuit. The complete task of PFE is classified into two functionalities: (1) Circuit Topology Hiding (CTH), (2) Private Gate Evaluation (PGE).

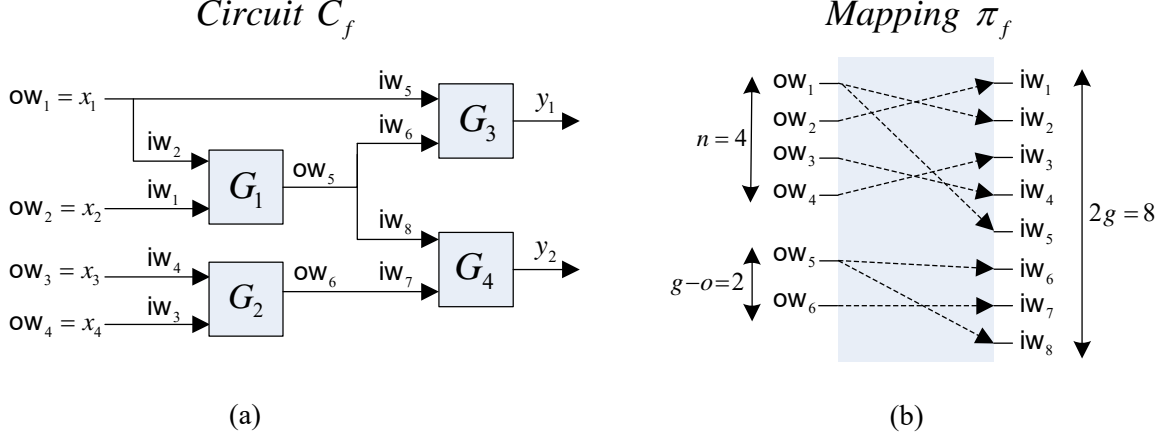
Throughout this paper the party who knows the private function is denoted by  $P_1$ , plays the evaluator role in 2PC; whereas the other party is denoted by  $P_2$  plays the garbler role in 2PC. In a nutshell, in CTH,  $P_1$  extracts the topological mapping  $\pi_f$  (kept private) from the circuit representation  $C_f$ , and converts the whole circuit into a collection of single gates. Then  $P_1$  and  $P_2$  engage in an *oblivious evaluation of switching network* (OSN) protocol where  $P_2$  obliviously obtains tokens on gate inputs. In PGE, a 2PC protocol is performed to obtain the final output. In the rest of this section, we describe the notions related to CTH, and the 2-PFE scheme proposed in [MS13].

#### 3.1 Context of CTH

Let  $n$  and  $m$  denote the number of inputs and outputs of  $C_f$ , respectively. Let  $g$  be the number of gates (size of circuit).  $\text{OW} : \{\text{ow}_1, \dots, \text{ow}_{n+g-m}\}$  denotes the set of outgoing wires which is the union of the input wires of the circuit and the output wires of its non-output gates (having  $M = n + g - m$  elements in total *whose indices are chosen randomly*). Similarly,  $\text{IW} : \{\text{iw}_1, \dots, \text{iw}_{2g}\}$  denotes the set of incoming wires which is the input wires of each gate in the circuit (having  $N = 2g$  elements in total *whose indices are chosen randomly*).

The full description of the topology of a boolean circuit  $C_f$  can be accomplished by a mapping  $\pi_f : \text{OW} \rightarrow \text{IW}$ . The mapping  $\pi_f$  maps  $i$  to  $j$  (*i.e.*,  $\pi_f(i) \rightarrow j$ ), if and only if  $\text{ow}_i \in \text{OW}$  and  $\text{iw}_j \in \text{IW}$  correspond to the same wire in the circuit  $C_f$ . Note that the mapping  $\pi_f$  is not a function if an outgoing wire corresponds to more than one incoming wire, while its inverse  $\pi_f^{-1}$  is always a function. Figure 1 shows an example circuit  $C_f$  and its mapping  $\pi_f$ .

<sup>11</sup> *Circular correlation robustness for naturally derived keys* is the security requirement for a suitable hash function used in half gates garbling. We refer the reader to [ZRE15] for its details.



**Fig. 1.** (a) A circuit representation  $C_f$  of a function  $f$ . (b) The mapping  $\pi_f$  of  $f$ .

From the inclusion-exclusion principle, we obtain Equation (3.2) that gives the number of possible mappings for the given  $M$  and  $N$  values.

$$\rho = \sum_{i=0}^M (-1)^i \binom{M}{i} (M-i)^N \quad (3.2)$$

In the context of CTH,  $\rho$  indicates the number of possible circuit topologies. Thus, the security of CTH is proportional to  $\rho$ . In what follows, we describe the main elements of CTH functionality whose essential target is the oblivious application of the mapping  $\pi_f$ .

*Oblivious evaluation of mapping* A mapping of the form  $\pi : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$  is a permutation if it is a bijection. We next define the *extended permutation* (EP) as follows:

**Definition 3.1 (Extended permutation (EP)).** *Given the positive integers  $M$  and  $N$ , a mapping  $\pi : \{1, \dots, M\} \rightarrow \{1, \dots, N\}$  is called an EP if for all  $y \in \{1, \dots, N\}$ , there exists a unique  $x \in \{1, \dots, M\}$  such that  $\pi(x) = y$ , and its inverse  $\pi^{-1} : \{1, \dots, N\} \rightarrow \{1, \dots, M\}$  is an onto function.*

The ideal 2-party oblivious evaluation of extended permutation (2-OEP) functionality is defined as follows:

**Definition 3.2 (2-OEP functionality).** *The first party  $P_1$ 's inputs are an EP  $\pi : \{1, \dots, M\} \rightarrow \{1, \dots, N\}$ , and a blinding vector for incoming wires  $\mathcal{T} := [t_j \leftarrow \{0, 1\}^\lambda]$  for  $j = 1, \dots, N$ . The other party  $P_2$ 's inputs are a vector for outgoing wires  $W := [w_i \leftarrow \{0, 1\}^\lambda]$  for  $i = 1, \dots, M$ . At the end,  $P_2$  learns  $S := [\sigma_j = w_{\pi^{-1}(j)} \oplus t_j]$  for  $j = 1, \dots, N$  while  $P_1$  learns nothing.*

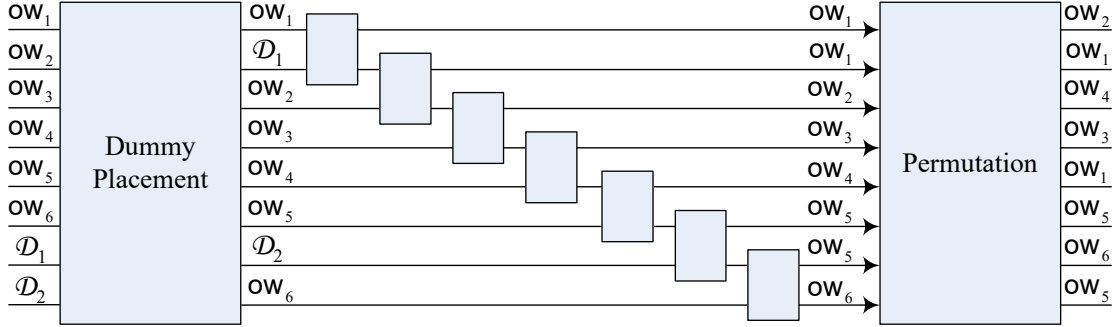
We call any 2-party protocol construction realizing the 2-OEP functionality as a *2-OEP protocol*. Mohassel and Sadeghian have constructed a constant round 2-OEP protocol by introducing the OSN structure. Since we also utilize their 2-OEP protocol in our scheme, here we give some of its details. Mainly, they first construct an extended permutation using switching networks, then provide a method using OTs for oblivious evaluation of the resulting switching network. We refer our reader to [MS13] for the security proof and application of this construction on various MPC protocols.

*EP construction from switching networks.* Each 2-switch takes two  $\lambda$ -bit strings and two selection bits as input, outputting two  $\lambda$ -bit strings [MS13]. Each of the outputs may get the value of any of the input strings depending on the selection bits. This means for input values  $(x_0, x_1)$ , there are four different switch output possibilities. The two selection bits  $s_0$  and  $s_1$  are used for determining the switch output  $(y_0, y_1)$ . In particular, the switch outputs  $y_0 = x_{s_0}$ , and  $y_1 = x_{s_1}$ .

Unlike 2-switches, 1-switches have only one selection bit  $s$ . For an input  $(x_0, x_1)$ , a 1-switch outputs one of the two possible outputs:  $(x_0, x_1)$  if  $s = 0$ , and  $(x_1, x_0)$  otherwise.

**Definition 3.3 (Switching Network (SN)).** A switching network  $SN$  is a collection of interconnected switches whose inputs are  $N$   $\lambda$ -bit strings and a set of selection bits of all switches, and whose outputs are  $N$   $\lambda$ -bit strings.

The mapping  $\pi : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$  related to an  $SN$  ( $\pi(i) = j$ ) implies that when the  $SN$  is executed, the string on the output wire  $j$  gets the value of that on the input wire  $i$ .



**Fig. 2.** The related switching network for the mapping  $\pi_f$  in Figure 1.

A permutation network  $PN$  is a special type of  $SN$  whose mapping is a permutation of its inputs. In contrast to  $SN$ s,  $PN$ s composed of 1-switches. Waksman proposes an efficient  $PN$  construction in [Wak68]. Mainly, this work suggests that a  $PN$  with  $N = 2^k$  inputs can be constructed with  $N \log(N) - N + 1$  switches. In [MS13], the authors propose the construction of an extended permutation by combining  $SN$ s and  $PN$ s. However, extended permutations differ from  $SN$ s in that the number of their inputs  $M$  and that of their outputs  $N$  need not be equal ( $M \leq N$ ).  $N - M$  additional dummy inputs are added to the real inputs of an  $EP$   $\pi : \{1, \dots, M\} \rightarrow \{1, \dots, N\}$  in order to simulate it as an  $SN$ . The  $SN$  design for extended permutation is divided into the following three components (see also Figure 2).

1. **Dummy placement component.** Dummy placement component takes  $N$  input strings composing of real and dummy ones. For each real input that  $\pi$  maps to  $k$  different outputs, the dummy-value placement component's output is the real string followed by  $k - 1$  dummy strings.
2. **Replication component.** Replication component takes the output of the dummy-value placement component as input. If a value is real, it goes unchanged. If it is a dummy value, it is replaced by the real value which precedes it. This can be computed by a series of  $N - 1$  2-switches whose selection bits  $(s_0, s_1)$  are either  $(0,0)$  or  $(0,1)$ . If the selection bits are  $(0,0)$ , that means  $x_1$  is dummy, and  $x_0$  goes both of the outputs. If they are  $(0,1)$ , that means both inputs are real, and both are kept on the outputs in the same order. At the end of this step, all the dummy inputs are replaced by the necessary copies of the real inputs.
3. **Permutation component.** Permutation component takes the output wires of the replication component as input. It outputs a permutation of them so that each string is placed on its final location according to the prescription of mapping  $\pi$ .

An efficient implementation of both dummy placement and permutation blocks is via the use of a Waksman permutation network. Combining these three components, one gets a larger switching network, where the number of switches needed is  $2(N \log(N) - N + 1) + N - 1 = 2N \log(N) - N + 1$  [MS13]. The topology of the whole switching network is the same for all  $N$  input  $EP$ s, and the selection bits specify the input values appearing on the outputs.

*Oblivious evaluation of SN construction (OSN)* We continue with describing Mohassel and Sadeghian’s method for oblivious evaluation of switching networks using OTs.

Adapting the switching network construction to the 2-OEP functionality,  $P_1$  produces the selection bits of the switching network using  $\pi$ , and has a blinding vector  $\mathcal{T}$ .  $P_2$  has an input vector for outgoing wires  $W$ . At the end,  $P_2$  learns the switching network’s blinded output vector for incoming wires  $S$ , and  $P_1$  learns  $\perp$ . We describe the oblivious evaluation of one of its building block, *i.e.*, a single 2-switch  $u$ .

Let the input wires of the 2-switch be  $a$  and  $b$ , and its output wires be  $c$  and  $d$ . Each of the four wires of the switch has a uniformly random string assigned by  $P_2$  as her share of that wire in the preparation stage, namely,  $r_a, r_b, r_c, r_d \leftarrow \{0, 1\}^\lambda$  for  $a, b, c, d$ , respectively.  $P_1$  has the strings  $w_1 \oplus r_a$  and  $w_2 \oplus r_b$  as his shares for the two input wires. The purpose is enabling  $P_1$  to obtain his output shares according to his selection bits. There are four possibilities for  $P_1$ ’s output shares depending on his selection bits  $s_{0u}$  and  $s_{1u}$  (see Table 2).

**Table 2.**  $P_1$  learns one of these rows according to his selection bits.

$(s_{0u}, s_{1u})$	$y_0$	$y_1$
(0,0)	$w_1 \oplus r_c$	$w_1 \oplus r_d$
(0,1)	$w_1 \oplus r_c$	$w_2 \oplus r_d$
(1,0)	$w_2 \oplus r_c$	$w_1 \oplus r_d$
(1,1)	$w_2 \oplus r_c$	$w_2 \oplus r_d$

$P_2$  prepares a table with four rows using  $r_a, r_b, r_c, r_d$  (see Table 3).  $P_1$  and  $P_2$  engage in a 1-out-of-4 OT in which  $P_2$  inputs the four rows that she has prepared, and  $P_1$  inputs his selection bits for the switch  $u$ . At the end,  $P_1$  learns one of the rows as the output in the table. Assume that  $P_1$ ’s selection bits are (1,0). This means  $P_1$  retrieves the third row, *i.e.*,  $(r_b \oplus r_c, r_a \oplus r_d)$ . According to the his selection bits,  $P_1$  XORs his input share  $w_2 \oplus r_b$  with  $r_b \oplus r_c$ , as well as his other input share  $w_1 \oplus r_a$  with  $r_a \oplus r_d$ , and obtains his output shares  $w_2 \oplus r_c$  and  $w_1 \oplus r_d$ .

The oblivious evaluation of the entire SN for EP goes as follows. In an offline stage,  $P_2$  sets a uniformly random  $\lambda$ -bit string to each wire in the switching network.  $P_2$  blinds each element of her input vector  $W$  and the dummy strings which she assigned for  $N - M$  inputs of the switching network with her corresponding shares for input wires (an XOR operation is involved in each blinding).  $P_2$  prepares tables for each switch in the switching network similar to Table 2 and Table 3. However, both tables for each switch in this scenario have two rows since each switch, in fact, has two possible outputs<sup>12</sup>. This means each switch in the entire switching network can be evaluated running 1-out-of-2 OT. Moreover, the construction permits parallel OT runs and or use of OT extension, resulting in a constant round scheme.  $P_2$  needs to send her blinded inputs to  $P_1$ , which can be done during her turn in OT extension in order not to increase the round complexity unnecessarily. Once  $P_1$  gets  $P_2$ ’s blinded inputs which are also his input shares and the outputs of all OTs, he evaluates the entire switching network in topological order, obtaining his output shares.  $P_1$  blinds his output shares with corresponding elements of  $\mathcal{T}$  (again, an XOR operation is involved in each blinding), and sends the resulting vector to  $P_2$ .  $P_2$  unblinds each element using her shares for output wires, and obtains the OEP output  $S$ . The extended permutation in this construction includes  $2N \log(N) - N + 1$  switches in total, requiring  $2N \log(N) - N + 1$  OTs for their oblivious evaluation.

### 3.2 Mohassel and Sadeghian’s 2-PFE scheme

Here we provide an outline of Mohassel and Sadeghian’s 2-PFE construction, and refer the reader to their work for detailed information and its security proof [MS13]. Their protocol is as follows.  $P_2$  first randomly

<sup>12</sup> For the 1-switches in dummy placement and permutation components, the first and second rows of Table 2 and Table 3, and for 2-switches in replacement components, the second and third rows of Table 2 and Table 3 are sufficient.



**Table 3.**  $P_1$  gets one of these rows by engaging in 1-out-of-4 OT with  $P_2$ .

$(s_{0u}, s_{1u})$	$\Omega_0$	$\Omega_1$
(0,0)	$r_a \oplus r_c$	$r_a \oplus r_d$
(0,1)	$r_a \oplus r_c$	$r_b \oplus r_d$
(1,0)	$r_b \oplus r_c$	$r_a \oplus r_d$
(1,1)	$r_b \oplus r_c$	$r_b \oplus r_d$

generates tokens  $w_i^0, w_i^1 \leftarrow \{0, 1\}^\lambda$  for each  $ow_i \in OW$  corresponding to FALSE and TRUE, respectively.  $P_1$  also generates random blinding strings  $t_j^0, t_j^1 \leftarrow \{0, 1\}^\lambda$  for each  $iw_j \in IW$ . And then  $P_1$  and  $P_2$  engage in OSN slightly modified from their 2-OEP protocol, where at the end,  $P_2$  learns  $[\sigma_j^0 = w_{\pi_f^{-1}(j)}^0 \oplus t_j^{b_j}]$  and  $[\sigma_j^1 = w_{\pi_f^{-1}(j)}^1 \oplus t_j^{b_j}]$ .  $P_2$  garbles each gate by encrypting the tokens  $w_c^0, w_c^1$  on its outgoing wire with the blinded strings  $\sigma_a^0, \sigma_a^1, \sigma_b^0, \sigma_b^1$  on its incoming wires according to its truth table.  $P_2$  sends the garbled gates and her garbled input tokens to  $P_1$ .  $P_1$  gets his garbled input tokens using OT which can be done in an earlier stage together with other OTs not to increase round complexity. Using the circuit mapping, his blinding strings, the garbled gates and the garbled inputs  $P_1$  evaluates the whole garbled circuit, and obtains the tokens of output bits of  $f(x)$ . In [MS13], a gate hiding mechanism is not provided for 2-PFE scheme but instead all gates in the circuit are let to be only a NAND gate.

Mohassel and Sadeghian’s scheme involves oblivious evaluation of a switching network made of  $2N \log(N) + 1$  switches. This is composed of an additional  $N$  switches to the ones in their EP construction. The oblivious evaluation of this switching network requires  $2N \log(N) + 1$  OTs [MS13]. All of the OTs in the protocol can be combined for just one invocation of OT extension. The overall computation cost<sup>13</sup> of [MS13] is about  $6N \log(N) + 2N + 12$  symmetric-key cryptographic operations.

## 4 Our Efficient 2-Party PFE Scheme

In what follows, we describe our scheme in detail (see also Figure 4). In the preparation stage,  $P_1$  compiles the function into a boolean circuit  $C_f$  consisting of only NAND gates<sup>14</sup>, and extract the circuit mapping  $\pi_f$  by randomly assigning incoming and outgoing wire indices. Both parties need to have the pre-knowledge of *template of private circuit*  $\tilde{C}_f$  defined as follows:

**Definition 4.1 (Template of Private Circuit ( $\tilde{C}_f$ )).** *A template of private circuit  $\tilde{C}_f$  is some information about a circuit  $C_f$  which consists of: (1) the number of each party’s input bits, (2) the number of output bits, (3) the total numbers of incoming ( $N$ ) and outgoing wires ( $M$ ), (4) the incoming and outgoing wire indices which belong to the same gates, (5) the outgoing wire indices corresponding to each parties inputs, and (6) the incoming wire indices belonging to output gates.*

We continue with describing the main parts of our scheme, namely 2-OEP and 2PC garbling protocols. Our complete 2-party PFE protocol is provided in Figure 3.

### 4.1 Use of 2-OEP protocol

Let  $w_i^0$  and  $w_i^1$  be the tokens for FALSE and TRUE on the  $i$ th outgoing wire  $ow_i \in OW$ , respectively, and  $R$  be the global free-XOR offset [KS08a] throughout the circuit.  $P_2$  sets  $w_i^0 \leftarrow \{0, 1\}^\lambda$  for each  $ow_i$ . The blinding string on the  $j$ th incoming wire  $iw_j \in IW$  is denoted as  $t_j$ .  $P_1$  sets  $t_j \leftarrow \{0, 1\}^\lambda$  for each  $iw_j$ .  $P_1$  and

<sup>13</sup> In [KS16], the computation cost of [MS13] is also computed. We note that there is a minor typo in [KS16, p. 723] i.e., the computation complexity of [MS13] should be  $12g \log(2g) + 4g + 12$  instead of  $12 \log(2g) + 4g + 12$  where  $N = 2g$  and  $g$  is the number of gates.

<sup>14</sup> Any functional-complete gate can be used to rule out the need for a gate hiding mechanism as in [MS13].

### Our 2-PFE Scheme.pdf

**$P_1$ 's Input:** A bit string  $x_1$  and a function  $f$ .

**$P_2$ 's Input:** A bit string  $x_2$ .

**Output:**  $f(x_1, x_2)$ .

#### Preparation:

1.  $P_1$  compiles the private function  $f$  into a boolean circuit  $C_f$  whose the number of input bits, output bits, and gates are  $n$ ,  $o$ , and  $g$ , respectively, extracts the mapping  $\pi_f$  by randomly assigning incoming and outgoing wire indices, and prepare the template of private circuit  $\tilde{C}_f$ .
2.  $P_1$  sends  $\tilde{C}_f$  to  $P_2$ .
3.  $P_2$  randomly generates an  $\lambda$ -bit token  $w_i^0 \leftarrow \{0, 1\}^\lambda$  for FALSE on each  $ow_i \in OW$ . This yields a total of  $M = n + g - o$  pairs. Moreover,  $P_2$  sets a vector  $W^0 := [w_i^0]$  for  $i = 1, \dots, M$ .
4.  $P_1$  generates an  $\lambda$ -bit blinding string  $t_j \leftarrow \{0, 1\}^\lambda$  for each  $iw_j \in IW$ . He sets those values to a blinding vector  $\mathcal{T} := [t_j]$  for  $j = 1, \dots, 2g$ .

#### 2-OEP Protocol:

5.  $P_2$  and  $P_1$  engage in a 2-OEP protocol where  $P_1$ 's inputs are the mapping  $\pi_f$  and  $\mathcal{T}$ , while  $P_2$ 's input is the vector  $W^0$ . At the end,  $P_2$  learns the blinded string vector  $S^0 := [\sigma_j^0 = w_{\pi_f^{-1}(j)}^0 \oplus t_j]$  for  $j = 1, \dots, N$ , while  $P_1$  learns  $\perp$ .

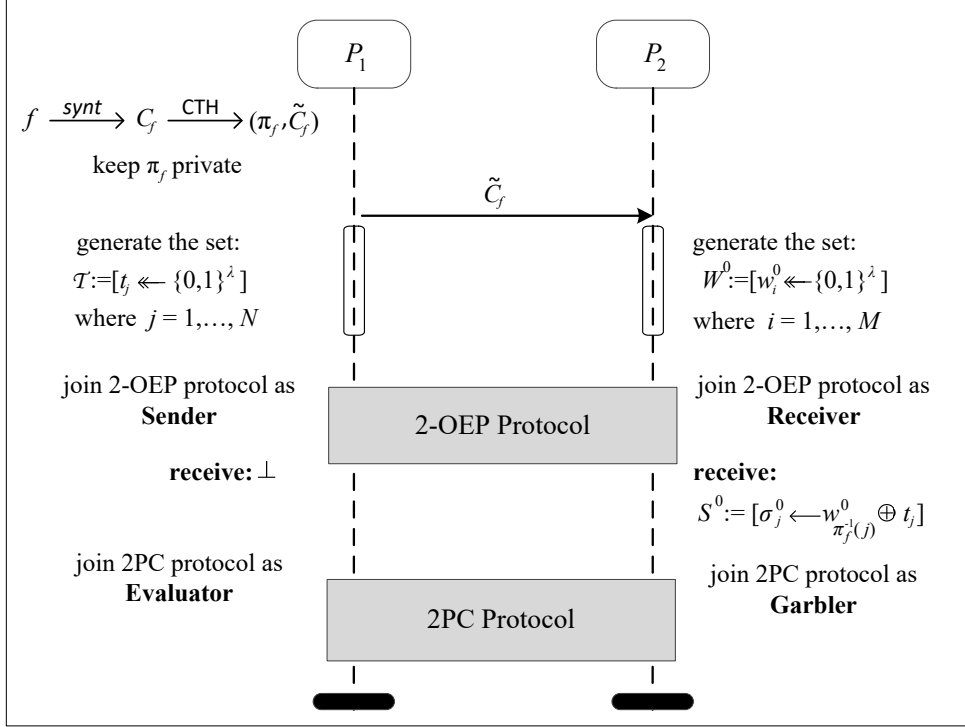
#### 2PC Protocol ( $P_2$ plays the garbler, and $P_1$ plays the evaluator):

6. **Garbling:**  $P_2$  generates a secret  $\lambda$ -bit offset  $R \leftarrow \{0, 1\}^{\lambda-1}$ .  $P_2$  sets the token for TRUE on each  $ow_i$  as  $w_i^1 \leftarrow w_i^0 \oplus R$ , and the blinded for TRUE on each  $iw_j$  as  $\sigma_j^1 \leftarrow \sigma_j^0 \oplus R$ . Moreover,  $P_2$  sets the sets  $W^1 := [w_i^1]$  for  $i = 1, \dots, M$  and  $S^1 := [\sigma_j^1]$  for  $j = 1, \dots, N$ . With the knowledge of  $W^0$ ,  $S^0$ ,  $S^1$  and  $\tilde{C}_f$ ,  $P_2$  garbles each odd gate using the **Gb** procedure in Figure 5, resulting in three ciphertexts per non-output gate and two ciphertexts per output gate.  $P_2$  sends the garbled circuit  $\hat{F}$  and the tokens  $\hat{X}_2$  for her own inputs  $x_2$  to  $P_1$ .  $P_1$  gets tokens  $\hat{X}_1$  for his own input bits  $x_1$  from  $P_2$  using 1-out-of-2 OTs. (If OSN construction is used, these OTs can be jointly executed with the ones for 2-OEP protocol in parallel and with just one invocation of extended OT. For this setting,  $P_2$  needs to pick  $R$  and compute the tokens for TRUE on  $P_1$ 's input wires before 2-OEP protocol.)
7. **Evaluating:** With the knowledge of  $\pi_f$ ,  $\mathcal{T}$ ,  $\hat{F}$  and the garbled input  $\hat{X} = (\hat{X}_1, \hat{X}_2)$ ,  $P_1$  evaluates the whole garbled circuit in topological order. When an outgoing wire  $i$  is mapped to an incoming wire  $j$ , the token  $w_i$  is XORed with  $t_j$  to reach the blinded string  $\sigma_j$ .  $P_1$  evaluates each garbled gate using the **Ev** procedure in Figure 5. At the end,  $P_1$  obtains the tokens for  $f(x_1, x_2)$ .

**Fig. 3.** Our 2-Party Private Function Evaluation Protocol

$P_2$  engage in a 2-OEP protocol where  $P_1$ 's inputs are  $\pi_f$  and a blinding vector for incoming wires  $\mathcal{T} := [t_j]$  for  $j = 1, \dots, N$ , and  $P_2$ 's inputs is a token vector for FALSE on outgoing wires  $W^0 := [w_i^0]$  for  $i = 1, \dots, M$ . At the end,  $P_2$  learns the vector of blinded strings for FALSE  $S^0 := [\sigma_j^0 = w_{\pi_f^{-1}(j)}^0 \oplus t_j]$  for  $j = 1, \dots, N$ , while  $P_1$  learns  $\perp$ .

Since our protocol allows all wires in the circuit to have the same offset  $R$ , unlike [MS13],  $P_1$  needs only a single blinding string  $t_j$  for each wire, and  $P_2$  does not need to input both tokens  $w_i^0$  and  $w_i^1$  to the 2-OEP protocol. This leads to a considerable decrease in communication cost compared to [MS13], in which two blinding strings  $t_j^0$  and  $t_j^1$  for each wire are used, and both  $w_i^0$  and  $w_i^1$  are inputs to the OSN protocol (slightly modified 2-OEP protocol).



**Fig. 4.** Components and high level procedures of our PFE protocol. The private function  $f$  is only known to  $P_1$ .  $P_1$  compiles  $f$  into a boolean circuit  $C_f$ , and extracts the mapping  $\pi_f$  and the template of private circuit  $\tilde{C}_f$ .  $P_1$  sends  $\tilde{C}_f$  to  $P_2$ .  $P_1$  randomly generates the vector  $\mathcal{T}$ .  $P_2$  randomly generates the vector  $W^0$ . They engage in a 2-OEP protocol where  $P_2$  learns  $S^0$  as the output. With the knowledge of  $W^0$ ,  $S^0$  and  $\tilde{C}_f$ ,  $P_2$  garbles each gate and sends the garbled circuit to  $P_1$ . With the knowledge of  $\pi_f$ ,  $\tilde{C}_f$ ,  $\mathcal{T}$ , the garbled circuit and the garbled inputs,  $P_1$  evaluates the whole garbled circuit.

## 4.2 Our 2PC garbling scheme for 2-PFE

This section presents our garbling scheme based on half gates technique [ZRE15]. Similar to half gates technique,  $P_2$  sets  $R \leftarrow \{0, 1\}^{\lambda-1}1$ ,  $w_i^1 \leftarrow w_i^0 \oplus R$  for TRUE on each  $ow_i$ , and  $\sigma_j^1 \leftarrow \sigma_j^0 \oplus R$  for TRUE on each  $iw_j$ . We have  $\text{lsb}(R) = 1$  so that  $\text{lsb}(w_i^0) \neq \text{lsb}(w_i^1)$ , and  $\text{lsb}(\sigma_j^0) \neq \text{lsb}(\sigma_j^1)$ .  $P_2$  follows the steps in Table 4 in order to garble each odd gate.

We now give some necessary notation as follows. Let  $w_c^0$  and  $w_c^1$  denote both tokens on an outgoing wire, while  $\sigma_a^0, \sigma_a^1, \sigma_b^0, \sigma_b^1$  denote the blinded strings on incoming wires. Let also  $v_j$  denote the one bit truth value on the  $j$ th incoming wire in a circuit. Further,  $w_{Gc}^b$  and  $w_{Ec}^b$  denote the tokens for the garbler and the evaluator half gate outputs for truth value  $b$ , respectively.  $T_{Gc}$  and  $T_{Ec}$  denote the  $\lambda$ -bit strings needed to be sent for the garbler and evaluator half gates, respectively.  $\psi_c$  denotes the additional  $\lambda$ -bit string needed to be sent for carrying to the specific output token.  $w_i$  and  $\sigma_j$  are the token on  $i$ th outgoing wire and the blinded string on  $j$ th incoming wire obtained by  $P_1$  while evaluating the garbled circuit, respectively. For the  $j$ th incoming wire, let  $p_j := \text{lsb}(\sigma_j^0)$  be a value only known to  $P_2$ . If two symbols are appended, we imply an AND operation, *i.e.*,  $ab = a \wedge b$ .  $H : \{0, 1\}^\lambda \times \mathbb{Z} \rightarrow \{0, 1\}^\lambda$  denotes a hash function with circular correlation robustness for naturally derived keys, having the security parameter  $\lambda$ . We use a ‘hat’ to represent a sequence or a tuple, for instance,  $\hat{F} = (F_1, F_2, \dots)$  or  $\hat{e} = (e_1, e_2, \dots)$ .

**Table 4.** Adapting half gates technique to our 2-PFE for garbling an odd gate. Here,  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$  define the gate type (e.g.,  $\alpha_1 = 0$ ,  $\alpha_2 = 0$  and  $\alpha_3 = 1$  for a NAND gate, see Equation (2.1)). The token  $w_c^0$  on the output wire equals  $w_{G_c}^0 \oplus w_{E_c}^0 \oplus \psi_c$ . The three ciphertexts  $T_{G_c}$ ,  $T_{E_c}$ , and  $\psi_c$  are sent to  $P_1$  for each gate.

Garbler half gate ( $p_b$ known to the garbler)	Evaluator half gate ( $p_b \oplus v_b$ known to the evaluator)
Defines the half gate:	Defines the half gate:
$f_G(v_a, p_b) := (\alpha_1 \oplus v_a)(\alpha_2 \oplus p_b) \oplus \alpha_3$	$f_E(v_a, v_b \oplus p_b) := (\alpha_1 \oplus v_a)(p_b \oplus v_b)$
Computes:	Computes:
$T_{G_c} \leftarrow H(\sigma_a^0) \oplus H(\sigma_a^1) \oplus (p_b \oplus \alpha_2)R$	$T_{E_c} \leftarrow H(\sigma_b^0) \oplus H(\sigma_b^1) \oplus \sigma_a^{\alpha_1}$
$w_{G_c}^0 \leftarrow H(\sigma_a^{p_a}) \oplus f_G(p_a, p_b)R$	$w_{E_c}^0 \leftarrow H(\sigma_b^{p_b})$
<hr/> Defines the third ciphertext: $\psi_c := w_{G_c}^0 \oplus w_{E_c}^0 \oplus w_c^0$ $P_2$ sends $T_{G_c}$ , $T_{E_c}$ , and $\psi_c$ .	

In accordance with the framework<sup>15</sup> of [BHR12], Figure 5 depicts our complete garbling scheme, composed of the following procedures. The garble procedure  $\text{Gb}$  takes  $1^\lambda$ ,  $\tilde{C}_f$ ,  $S^0$  and  $W^0$  as input, and outputs  $(\hat{F}, \hat{e}, \hat{d})$  where  $\hat{F}$  is the garbled version of  $\tilde{C}_f$ ,  $\hat{e}$  is the encoding information, and  $\hat{d}$  is decoding information.  $\text{Gb}$  calls two private gate garbling procedures: (1)  $\text{Gb}_{\text{NAND}}^*$  garbles non-output NAND gates, and returns  $(T_G, T_E, \psi)$ , (2)  $\text{Gb}_{\text{NAND}}$  garbles output NAND gates, and returns  $(T_G, T_E, Y^0)$ .  $\text{En}$  is the encode algorithm that takes the plaintext input  $\hat{x}$  of the circuit and  $e$  as input, and outputs a garbled input  $\hat{X}$ .  $\text{Ev}$  is the evaluate procedure that takes the inputs  $\hat{F}$ ,  $\hat{X}$ ,  $\pi_f$  and  $\mathcal{T}$ , and outputs garbled output  $\hat{Y}$ .  $\text{De}$  is the decode algorithm that takes  $\hat{Y}$  and  $d$  as input, and outputs the plaintext output  $\hat{y}$  of the circuit.

We highlight that an essential difference of our garbling scheme from the half gates technique is that the former requires an additional ciphertext  $\psi_c$  per gate. This is required because of the nature of 2-PFE, in which the tokens on an outgoing wire are predetermined and specified values, while in the in half gates they are indeed a function of the input strings. Since in our scheme the output tokens of output gates are not predetermined, these gates can be garbled with half gates technique. Each output gate is then garbled with two ciphertexts. Note also that  $P_1$  gets his own garbled inputs by means of OT. This can also be done in an earlier stage together with other OTs in 2-OEP protocol (if OSN construction is used) in order not to increase round complexity. For this setting,  $P_2$  needs to pick  $R$  and compute the tokens for TRUE on  $P_1$ 's input wires before 2-OEP protocol. This setting is compatible with our protocol as well.

## 5 Security of the proposed protocol

In this section, we start by revisiting the code based games of Bellare, Hoang and Rogaway [BHR12] and security notions of Choi *et al.* [CKKZ12] and Zahur *et al.* [ZRE15] as preliminaries. We then provide simulation based security proof of our proposed protocol.

### 5.1 Code based games and security notions

Our work uses the  $\text{prv.sim}_S$  (privacy),  $\text{obv.sim}_S$  (obliviousness) and  $\text{auth}_S$  (authenticity) security definitions of [BHR12] depicted in Figure 7. Considering the  $\text{prv.sim}$  and  $\text{obv.sim}$  games, the Initialize procedure randomly chooses  $\beta \leftarrow \{0, 1\}$ , then the adversary makes a single call to the Garble procedure, and then the Finalize procedure returns  $\beta \stackrel{?}{=} \beta'$ , where  $\beta'$  denotes the guess of the adversary. Regarding all three games,

<sup>15</sup> Bellare, Hoang, and Rogaway introduce the notion of a garbling scheme as a cryptographic primitive. They also describe procedures and security requirements of garbling schemes. We refer the reader to [BHR12, Hoa13] for details concerning definitions and introduction to the formal concepts of garbling schemes.

```

proc Gb( $1^\lambda, \tilde{C}_f, S^0, W^0$ ) :
   $R \leftarrow \{0, 1\}^{\lambda-1}$ 
  for  $i, j \in \tilde{C}_f$  do
     $\sigma_j^1 \leftarrow \sigma_j^0 \oplus R$ 
  for  $ow_i \in \text{Inputs}(\tilde{C}_f)$  do
     $e_i \leftarrow w_i^0$ 
  for each gate  $\tilde{G}_i \in \tilde{C}_f$  do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
    if  $\tilde{G}_i$  is a non-output gate then
       $(T_{G_i}, T_{E_i}, \psi_i) \leftarrow \text{Gb}_{\text{NAND}}^*(\sigma_a^0, \sigma_b^0, w_i^0)$ 
       $F_i^{\text{non-out}} \leftarrow (T_{G_i}, T_{E_i}, \psi_i)$ 
    else
       $(T_{G_i}, T_{E_i}, Y_i^0) \leftarrow \text{Gb}_{\text{NAND}}(\sigma_a^0, \sigma_b^0)$ 
       $F_i^{\text{out}} \leftarrow (T_{G_i}, T_{E_i})$ 
       $Y_i^1 \leftarrow Y_i^0 \oplus R$ 
       $d_i \leftarrow \text{lsb}(Y_i^0)$ 
    end if
  return  $(\hat{F}, \hat{e}, \hat{d})$ 

private proc Gb $_{\text{NAND}}^*$ ( $\sigma_a^0, \sigma_b^0, w^0$ ):
   $p_a \leftarrow \text{lsb}(\sigma_a^0); p_b \leftarrow \text{lsb}(\sigma_b^0)$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^0, k) \oplus H(\sigma_b^1, k) \oplus p_b R$ 
   $w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G \oplus R$ 
   $T_E \leftarrow H(\sigma_b^0, k') \oplus H(\sigma_b^1, k') \oplus \sigma_a^0$ 
   $w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_b (T_E \oplus \sigma_a^0)$ 
   $\psi \leftarrow w_G^0 \oplus w_E^0 \oplus w^0$ 
  return  $(T_G, T_E, \psi)$ 

private proc Gb $_{\text{NAND}}$ ( $\sigma_a^0, \sigma_b^0$ ):
   $p_a \leftarrow \text{lsb}(\sigma_a^0); p_b \leftarrow \text{lsb}(\sigma_b^0)$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^0, k) \oplus H(\sigma_a^1, k) \oplus p_b R$ 
   $w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G \oplus R$ 
   $T_E \leftarrow H(\sigma_b^0, k') \oplus H(\sigma_b^1, k') \oplus \sigma_a^0$ 
   $w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_b (T_E \oplus \sigma_a^0)$ 
   $Y^0 \leftarrow w_G^0 \oplus w_E^0$ 
  return  $(T_G, T_E, Y^0)$ 

proc En( $\hat{e}, \hat{x}$ ):
  for  $e_i \in \hat{e}$  do
     $X_i \leftarrow e_i \oplus x_i R$ 
  return  $\hat{X}$ 

proc Ev( $\hat{F}, \hat{X}, \pi_f, \mathcal{T}$ ):
  put  $\hat{F}$  in topological order using  $\pi_f$ 
  for  $ow_i \in \text{Inputs}(\hat{F})$  and  $j = \pi_f(i)$  do
     $\sigma_j \leftarrow X_i \oplus t_j$ 
  for each gate  $\tilde{G}_i$  {in topo. order} do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
     $s_a \leftarrow \text{lsb}(\sigma_a); s_b \leftarrow \text{lsb}(\sigma_b)$ 
     $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
     $(T_{G_i}, T_{E_i}, \psi_i) \leftarrow F_i^{\text{non-out}}$ 
     $w_{G_i} \leftarrow H(\sigma_a, k) \oplus s_a T_{G_i}$ 
    if  $\tilde{G}_i$  is a non-output gate then
       $w_{E_i} \leftarrow H(\sigma_b, k') \oplus s_b (T_{E_i} \oplus \sigma_a)$ 
       $w_i \leftarrow w_{G_i} \oplus w_{E_i} \oplus \psi_i$ 
      for  $j = \pi_f(i)$  do
         $\sigma_j \leftarrow w_i \oplus t_j$ 
      else
         $(T_{G_i}, T_{E_i}) \leftarrow F_i^{\text{out}}$ 
         $w_{G_i} \leftarrow H(\sigma_a, k) \oplus s_a T_{G_i}$ 
         $w_{E_i} \leftarrow H(\sigma_b, k') \oplus s_b (T_{E_i} \oplus \sigma_a)$ 
         $w_i \leftarrow w_{G_i} \oplus w_{E_i}$ 
         $Y_i \leftarrow w_i$ 
      end if
    return  $\hat{Y}$ 

proc De( $\hat{d}, \hat{Y}$ ):
  for  $d_i \in \hat{d}$  do
     $y_i \leftarrow d_i \oplus \text{lsb}(Y_i)$ 
  return  $\hat{y}$ 

```

**Fig. 5.** Our complete half gate based garbling scheme for 2-PFE.  $\text{Gb}_{\text{NAND}}$  and  $\text{Gb}_{\text{NAND}}^*$  are the original half gate and our modified NAND garbling procedures, respectively. A ‘hat’ represents a sequence or a tuple, for instance,  $\hat{F} = (F_1, F_2, \dots)$  or  $\hat{e} = (e_1, e_2, \dots)$ .

```

proc De( $\hat{d}, \hat{Y}$ ):
  for  $d_i \in \hat{d}$  do
     $k \leftarrow \text{NextIndex}();$  parse  $(h_0, h_1) \leftarrow d_i$ 
    if  $H(Y_i, k) = h_0$  then  $y_i \leftarrow 0$ 
    else if  $H(Y_i, k) = h_1$  then  $y_i \leftarrow 1$ 
    else return  $\perp$ 
  return  $\hat{y}$ 

```

**Fig. 6.** Modification of our garbling scheme in Figure 5 for achieving authenticity (auth) property.

the adversary is allowed to make a single call to the `Garble` procedure. For further information about the simulation based games and related security properties, we refer reader to [BHR12]. The advantages of the corresponding adversary classes are as follows:

$\text{prv.sim}_{G,\Phi,S}$ :	$\text{obv.sim}_{G,\Phi,S}$ :	$\text{auth}_G$ :
$\text{Garble}(f, x)$ : if $\beta = 0$ $(F, e, d) \leftarrow \text{Gb}(1^\lambda, f)$ $X \leftarrow \text{En}(e, x)$ else $(F, X, d) \leftarrow S(1^\lambda, f(x), \Phi(f))$ return $(F, X, d)$	$\text{Garble}(f, x)$ : if $\beta = 0$ $(F, e, d) \leftarrow \text{Gb}(1^\lambda, f)$ $X \leftarrow \text{En}(e, x)$ else $(F, X) \leftarrow S(1^\lambda, \Phi(f))$ return $(F, X)$	$\text{Garble}(f, x)$ : $(F, e, d) \leftarrow \text{Gb}(1^\lambda, f)$ $X \leftarrow \text{En}(e, x)$ return $(F, X)$ $\text{Finalize}(Y)$ : return $\text{De}(d, Y) \notin \{\perp, f(x)\}$

**Fig. 7.** Simulation based games for privacy, obliviousness and authenticity [BHR12]. The function  $S$  is a simulator, and  $G$  denotes a garbling scheme.

$$\begin{aligned}
 \text{Adv}_{G,\Phi,S}^{\text{prv.sim}}(\mathcal{A}, \lambda) &:= \left| \Pr[\text{prv.sim}_{G,\Phi,S}^{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \\
 \text{Adv}_{G,\Phi,S}^{\text{obv.sim}}(\mathcal{A}, \lambda) &:= \left| \Pr[\text{obv.sim}_{G,\Phi,S}^{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \\
 \text{Adv}_G^{\text{auth}}(\mathcal{A}, \lambda) &:= \Pr[\text{auth}_G^{\mathcal{A}}(1^\lambda) = 1]
 \end{aligned} \tag{5.3}$$

In order to provide the security of a scheme, in each game, the adversary must have a negligible advantage. We also utilize the following two oracle definitions of [ZRE15].

- $\text{Circ}_R(x, j, b) = H(x \oplus R, j) \oplus bR$  where  $R \in \{0, 1\}^{\lambda-1}$
- $\text{Rand}(x, j, b)$ : A random function that gives  $\lambda$ -bit output.

Note that the adversary is only allowed to access the oracle  $\text{Circ}_R$  with *legal queries*<sup>16</sup> in order to prevent the adversary from trivially obtaining  $R$  [CKKZ12]. Furthermore, we give the following definition for *natural queries*.

**Definition 5.1.** [ZRE15] *If a series of queries of the form  $(x, j, b)$  to an oracle  $O$  satisfies the following conditions*

- we have  $i = q$  for the  $q$ th query,
- $b \in \{0, 1\}$ ,
- $x$  is naturally derived, i.e., it is obtained by one of these operations:
  - (a)  $x \leftarrow \{0, 1\}^k$ ,
  - (b)  $x \leftarrow x_1 \oplus x_2$ , where  $x_1$  and  $x_2$  are naturally derived,
  - (c)  $x \leftarrow H(x_1, i)$  where  $x_1$  is naturally derived and  $i \in \mathbb{Z}$ ,
  - (d)  $x \leftarrow O(x_1, i, b)$  where  $x_1$  is naturally derived,

then these queries are natural.

If for all PPT adversaries  $\mathcal{A}$  making legal and natural queries

$$\left| \Pr_{\text{Rand}}[\mathcal{A}^{\text{Rand}}(1^\lambda) = 1] - \Pr_R[\mathcal{A}^{\text{Circ}_R}(1^\lambda) = 1] \right| < \epsilon$$

then  $H$  satisfies circular correlation robustness property for naturally derived keys, where  $\epsilon$  is negligible.

<sup>16</sup> A series of queries of the form  $(x, j, b)$  is *legal* if the verbatim value of  $(x, j)$  is never queried with alternating values of  $b$  [CKKZ12].

```

proc  $\mathcal{S}(1^\lambda, \tilde{C}_f, \pi_f, T, \hat{y})$  :
  for  $ow_i \in OW(\tilde{C}_f)$  do
     $w_i^0 \leftarrow \{0, 1\}^\lambda$ 
  for  $iw_j \in IW(\tilde{C}_f)$  do
     $\sigma_j^0 \leftarrow w_{\pi_f^{-1}(j)} \oplus t_j$ 
  for  $ow_i \in Inputs(\tilde{C}_f)$  do
     $X_i \leftarrow w_i^0$ 
  for each gate  $\tilde{G}_i \in \tilde{C}_f$  do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
    if  $\tilde{G}_i$  is a non-output gate then
       $(T_{G_i}, T_{E_i}, \psi_i) \leftarrow \text{Sim}_{\text{NAND}}^*(\sigma_a^0, \sigma_b^0, w_i^0)$ 
       $F_i^{\text{non-out}} \leftarrow (T_{G_i}, T_{E_i}, \psi_i)$ 
    else
       $(T_{G_i}, T_{E_i}, Y_i^0) \leftarrow \text{Sim}_{\text{NAND}}(\sigma_a^0, \sigma_b^0)$ 
       $F_i^{\text{out}} \leftarrow (T_{G_i}, T_{E_i})$ 
       $d_i \leftarrow \text{lsb}(Y_i^0) \oplus y_i$ 
    end if
  return  $(\hat{F}, \hat{X}, \hat{d})$ 

proc  $\mathcal{G}_1^0(1^\lambda, \tilde{C}_f, \pi_f, T, \hat{x})$  : //  $\mathcal{G}_2^{\text{Circ}R}$ 
   $\hat{v} \leftarrow \text{evalWires}(\tilde{C}_f, \pi_f, \hat{x})$ 
  for  $ow_i \in OW(\tilde{C}_f)$  do
     $w_i^{v_i} \leftarrow \{0, 1\}^\lambda$  //  $w_i^{v_i} \leftarrow w_i^{v_i} \oplus R$ 
  for  $iw_j \in IW(\tilde{C}_f)$  do
     $\mathcal{B} := v_{\pi_f^{-1}(j)}, \sigma_j^{\mathcal{B}} \leftarrow w_{\pi_f^{-1}(j)} \oplus t_j$ 
  for  $ow_i \in Inputs(\tilde{C}_f)$  do
     $X_i \leftarrow w_i^{v_i}$ 
  for each gate  $\tilde{G}_i \in \tilde{C}_f$  do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
    if  $\tilde{G}_i$  is a non-output gate then
       $(T_{G_i}, T_{E_i}, \psi_i) \leftarrow \text{Sim}_{\text{NAND}_1}^{*0}(\sigma_a^{v_a}, \sigma_b^{v_b}, w_i^{v_i}, v_a, v_b)$ 
       $F_i^{\text{non-out}} \leftarrow (T_{G_i}, T_{E_i}, \psi_i)$ 
    else
       $(T_{G_i}, T_{E_i}, Y_i^{v_i}) \leftarrow \text{Sim}_{\text{NAND}_1}^0(\sigma_a^{v_a}, \sigma_b^{v_b}, v_a, v_b)$ 
       $F_i^{\text{out}} \leftarrow (T_{G_i}, T_{E_i})$ 
       $Y_i^{v_i} \leftarrow Y_i^{v_i} \oplus R$ 
       $d_i \leftarrow \text{lsb}(Y_i^{v_i}) \oplus v_i$ 
    end if
  return  $(\hat{F}, \hat{X}, \hat{d})$ 

proc  $\mathcal{G}_3(1^\lambda, \tilde{C}_f, \pi_f, T, \hat{x})$  :
   $R \leftarrow \{0, 1\}^{\lambda-1}$ 
  for  $ow_i \in OW(\tilde{C}_f)$  do
     $w_i^0 \leftarrow \{0, 1\}^\lambda, w_i^1 \leftarrow w_i^0 \oplus R$ 
  for  $iw_j \in IW(\tilde{C}_f)$  do
     $\sigma_j^0 \leftarrow w_{\pi_f^{-1}(j)} \oplus t_j, \sigma_j^1 \leftarrow \sigma_j^0 \oplus R$ 
  for  $ow_i \in Inputs(\tilde{C}_f)$  do
     $X_i \leftarrow w_i^{x_i}$ 
  for each gate  $\tilde{G}_i \in \tilde{C}_f$  do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
    if  $\tilde{G}_i$  is a non-output gate then
       $(T_{G_i}, T_{E_i}, \psi_i) \leftarrow \text{Sim}_{\text{NAND}_3}^*(\sigma_a^0, \sigma_b^0, w_i^0)$ 
       $F_i^{\text{non-out}} \leftarrow (T_{G_i}, T_{E_i}, \psi_i)$ 
    else
       $(T_{G_i}, T_{E_i}, Y_i^0) \leftarrow \text{Sim}_{\text{NAND}_3}(\sigma_a^0, \sigma_b^0)$ 
       $F_i^{\text{out}} \leftarrow (T_{G_i}, T_{E_i})$ 
       $Y_i^1 \leftarrow Y_i^0 \oplus R, d_i \leftarrow \text{lsb}(Y_i^0)$ 
    end if
  return  $(\hat{F}, \hat{X}, \hat{d})$ 

private proc  $\text{Sim}_{\text{NAND}}^*(\sigma_a^0, \sigma_b^0, w^0)$  : //  $\text{Sim}_{\text{NAND}}(\sigma_a^0, \sigma_b^0)$  :
   $p_a \leftarrow \text{lsb}(\sigma_a^0); p_b \leftarrow \text{lsb}(\sigma_b^0)$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^0, k) \oplus \text{Rand}(\sigma_a^0, k, p_b)$ 
   $w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G$ 
   $T_E \leftarrow H(\sigma_b^0, k') \oplus \text{Rand}(\sigma_b^0, k', 0) \oplus \sigma_a^0$ 
   $w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_a(T_E \oplus \sigma_a^0)$ 
   $\psi \leftarrow w_G^0 \oplus w_E^0 \oplus w^0$  //  $Y^0 \leftarrow w_G^0 \oplus w_E^0$ 
  return  $(T_G, T_E, \psi)$  //  $(T_G, T_E, Y^0)$ 

private proc  $\text{Sim}_{\text{NAND}_1}^{*0}(\sigma_a^{v_a}, \sigma_b^{v_b}, w_i^{v_i}, v_a, v_b)$  :
  //  $\text{Sim}_{\text{NAND}_1}^0(\sigma_a^{v_a}, \sigma_b^{v_b}, v_a, v_b)$  :
   $s_a \leftarrow \text{lsb}(\sigma_a^{v_a}); s_b \leftarrow \text{lsb}(\sigma_b^{v_b})$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^{v_a}, k) \oplus O(\sigma_a^{v_a}, k, v_b \oplus s_b)$ 
   $w_G^{v_a(v_b \oplus s_b)} \leftarrow H(\sigma_a^{v_a}, k) \oplus s_a T_G$ 
   $T_E \leftarrow H(\sigma_b^{v_b}, k') \oplus O(\sigma_b^{v_b}, k', v_a) \oplus \sigma_a^{v_a}$ 
   $w_E^{v_a s_b} \leftarrow H(\sigma_b^{v_b}, k') \oplus s_b(T_E \oplus \sigma_a^{v_a})$ 
   $\psi \leftarrow w_G^{v_a(v_b \oplus s_b)} \oplus w_E^{v_a s_b} \oplus w_i^{v_i}$ 
  //  $Y \leftarrow w_G^{v_a(v_b \oplus s_b)} \oplus w_E^{v_a s_b}$ 
  return  $(T_G, T_E, \psi)$  //  $(T_G, T_E, Y)$ 

private proc  $\text{evalWires}(\tilde{C}_f, \pi_f, \hat{x})$  :
  for  $iw_j \in \tilde{C}_f$  do  $v_i \leftarrow x_i$ 
  for each gate  $\tilde{G}_i \in \tilde{C}_f$  do
     $\{a, b\} \leftarrow \text{GateInputs}(\tilde{G}_i)$ 
     $v_i \leftarrow \text{NAND}(v_a, v_b)$ 
  return  $\hat{v}$ 

private proc  $\text{Sim}_{\text{NAND}_3}^*(\sigma_a^0, \sigma_b^0, w^0)$  :
  //  $\text{Sim}_{\text{NAND}_3}(\sigma_a^0, \sigma_b^0)$  :
   $p_a \leftarrow \text{lsb}(\sigma_a^0); p_b \leftarrow \text{lsb}(\sigma_b^0)$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^0, k) \oplus H(\sigma_a^1, k) \oplus p_b R$ 
   $w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G \oplus R$ 
   $T_E \leftarrow H(\sigma_b^0, k') \oplus H(\sigma_b^1, k') \oplus \sigma_a^0$ 
   $w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_b(T_E \oplus \sigma_a^0)$ 
   $\psi \leftarrow w_G^0 \oplus w_E^0 \oplus w^0$ 
  //  $Y^0 \leftarrow w_G^0 \oplus w_E^0$ 
  return  $(T_G, T_E, \psi)$  //  $(T_G, T_E, Y^0)$ 

```

**Fig. 8.** The simulator for  $\text{prv.sim}_5$  security, and the hybrids<sup>15</sup> used in the proof. We obtain  $\mathcal{G}_2$  by adding the statements within sharp corner boxes to  $\mathcal{G}_1$ . The use of the statements within rounded-corner boxes alters the procedures from garbling of non-output gate to garbling of output gate. A ‘hat’ represents a sequence or a tuple, for instance,  $\hat{F} = (F_1, F_2, \dots)$  or  $\hat{e} = (e_1, e_2, \dots)$ .

## 5.2 Security Proof

Our security proof is based on the security proofs provided in [KM11] and [ZRE15].

**Theorem 5.1.** *If the following three conditions hold*

- *the 2-OEP protocol securely realizes ideal 2-OEP functionality in presence of semi-honest adversaries,*
- *the hash function  $H$  has circular correlation robustness for naturally derived keys,*
- *the OT scheme for acquisition of  $P_1$ 's garbled input by  $P_2$  securely realizes  $\mathcal{F}_{OT}$  functionality in the OT-hybrid model against semi-honest adversaries,*

*then our scheme is secure against semi-honest adversaries.*

*Proof.* We prove the security of our scheme against corruption of either parties, separately. First, consider the case that  $P_1$  is corrupted. Since the ideal 2-OEP functionality outputs  $\perp$  for  $P_1$ , and the transcripts received by  $P_1$  during OT reveals nothing other than  $P_1$ 's garbled input due to the ideal execution  $\mathcal{F}_{OT}$  in the OT-hybrid model, we only need to prove that the 2PC phase does not give any private information about  $P_2$ 's input to  $P_1$ . For any probabilistic polynomial time adversary  $\mathcal{A}_1$ , controlling  $P_1$  in the real world, we construct a simulation game based on `prv.sim` game from [BHR12] as follows. The simulation involves `Initialize`, `Garble`, and `Finalize` procedures. The `Initialize` procedure picks a value  $\beta \leftarrow \{0, 1\}$  randomly. Then,  $\mathcal{A}_1$  makes a single call to the `Garble` procedure (see `prv.sim` game of Figure 7). Note that  $\mathcal{S}$  denotes the simulation function, and `Gb` denotes the actual garbling (Figure 8 shows the procedure for  $\mathcal{S}$ ). We highlight that in our simulation, the side-information  $\phi(f)$  is replaced by  $(\tilde{C}_f, \pi_f, \mathcal{T})$ , since they are already known to  $P_1$ . Finally, in the `Finalize`( $\beta'$ ) procedure,  $\mathcal{A}_1$  tries to make a guess  $\beta'$  for the value of  $\beta$ , and the procedure outputs  $\beta \stackrel{?}{=} \beta'$ . We now prove that the simulation function output  $(\hat{F}, \hat{X}, \hat{d})$  is computationally indistinguishable from  $(F, X, d)$  by using the chain of hybrids as follows (see also Figure 8).

1.  $\mathcal{S} =_c \mathcal{G}_1^{\text{Rand}}$ : Since both generated  $(\hat{F}, \hat{X}, \hat{d})$  outputs include uniformly random values for components, their distributions are identical. More concretely, since the truth values of wires  $v_i$ 's are used only as a superscript for the tokens  $W^{v_i}$  by  $\mathcal{G}_1$ , these  $W_i^{v_i}$ 's could have been named  $W_i^0$  for all  $i$  values.
2.  $\mathcal{G}_1^{\text{Rand}} =_c \mathcal{G}_1^{\text{Circ}_R}$ : Only the oracle  $O$  changed from `Rand` to `CircR`. Due to our assumption about the hash function, these two hybrids are computationally indistinguishable.
3.  $\mathcal{G}_1^{\text{Circ}_R} =_c \mathcal{G}_2^{\text{Circ}_R}$ :  $\mathcal{G}_2$  is obtained by the addition of the statements within sharp corner boxes to  $\mathcal{G}_1$  in Figure 8. Here, the variable  $R$  in  $\mathcal{G}_2$  refers to the  $R$  of the oracle `CircR`. The only difference between the two hybrids is that some extra values that are not used computed by  $\mathcal{G}_2$  (those extra values will be used in  $\mathcal{G}_3$ ).
4.  $\mathcal{G}_2^{\text{Circ}_R} =_c \mathcal{G}_3$ :  $\mathcal{G}_3$  does not need to compute  $v_i$  for non-input wires and to randomly sample  $W_i^{v_i}$ , instead it randomly samples  $W_i^0$ . Next, it sets  $W_i^1 \leftarrow W_i^0 \oplus R$  instead of setting  $W_i^{v_i} \leftarrow W_i^{v_i} \oplus R$ . The algebraic relationships among variables remain unchanged. The oracle calls are also expanded in `SimAnd3` to correspond to  $O = \text{Circ}_R$ .

Note that  $\mathcal{G}_3$  computes  $(\hat{F}, \hat{X}, \hat{d})$  as  $(\hat{F}, \hat{e}, \hat{d}) \leftarrow \text{Gb}(1^\lambda, f)$ ;  $\hat{X} \leftarrow \text{En}(\hat{e}, \hat{x})$ , which is exactly how these values are computed in the real interaction in the `prv.simS` game. Therefore, the advantage of  $\mathcal{A}_1$  in the `prv.sim` game

$$\text{Adv}_{\mathcal{G}, \mathcal{S}}^{\text{prv.sim}}(\mathcal{A}, \lambda) := \left| \Pr[\text{prv.sim}_{\mathcal{G}, \mathcal{S}}^{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

is negligible. Hence, our scheme satisfies the security notion of `prv.simS` and `obv.simS`<sup>17</sup>. This proves that our scheme is secure against the corrupted  $P_1$ .

Second, consider the case that  $P_2$  is corrupted. For any probabilistic polynomial-time adversary  $\mathcal{A}_2$ , controlling  $P_2$  during our protocol in the real world, we construct a simulator  $\mathcal{S}'$  that simulates  $\mathcal{A}_2$ 's view in the ideal world.  $\mathcal{S}'$  runs  $\mathcal{A}_2$  on  $P_2$ 's input, and  $\tilde{C}_f$  as follows.

<sup>17</sup> The proof for `obv.simS` differs from that of `prv.simS` only in that in `obv.simS`, the simulator neither computes  $\hat{d}$ , nor receives  $\hat{y}$ . So providing a proof for `prv.simS` also implies a proof for `obc.simS`.



1.  $S'$  asks  $\mathcal{A}_2$  to generate  $\hat{W}^0 := [\hat{w}_i^0 \leftarrow \{0, 1\}^\lambda]$  for each  $\text{ow}_i \in \text{OW}$  and receives  $\hat{W}^0$ .
2.  $S'$  then picks  $\hat{t}_j \leftarrow \{0, 1\}^\lambda$  for  $j = 1, \dots, N$ , and computes  $\hat{S}^0 = [\hat{\sigma}_j \leftarrow \hat{w}_{\pi_f^{-1}(j)}^0 \oplus \hat{t}_j]$  and gives  $\hat{S}^0$  to  $\mathcal{A}_2$ .

In the real execution of our protocol,  $P_2$  receives only the message  $S^0$  in Round 2 (apart from the exchanged messages during the OT protocol for  $P_1$ 's garbled input). However, the transcripts received by  $P_2$  during OT do not leak any information to  $P_2$  because of  $\mathcal{F}_{OT}$  in the OT-hybrid model. Due to one-time pad security, in  $P_2$ 's view, the distributions of  $\hat{S}^0$  and  $S^0$  are identical (i.e.,  $U_{\hat{S}^0} \approx_c U_{S^0}$ ). This concludes the security proof of our scheme.

{replace the last three lines of  $S$  with the following ones:}

```

k ← NextIndex(); r ← {0, 1}^λ
if y_i = 0
  then d_i ← (H(Y_i^0, k), r)
  else d_i ← (r, H(Y_i^0, k))
end if
return (F̂, X̂, d̂)

```

**Fig. 9.** The required modifications on Figure 8 in order to show auth property.

In order to achieve the authenticity property (i.e., **auth**), it is required to show that the probability of an adversary finding a set  $\hat{Y}' \neq \text{Ev}(\hat{F}, \hat{X})$  such that  $\text{De}(\hat{d}, \hat{Y}') \neq \perp$  is negligible. In accordance with [ZRE15], our garbling scheme in Figure 5 can be modified as in Figure 6 to achieve authenticity property (i.e., the antepenultimate line of **Gb** in Figure 5 can be modified as  $k \leftarrow \text{NextIndex}(); d_i \leftarrow (H(Y_i^0, k), H(Y_i^1, k))$ , and  $\text{De}(\hat{d}, \hat{Y})$  procedure in Figure 5 can be modified as  $\text{De}(\hat{d}, \hat{Y})$  procedure in Figure 6).

**Theorem 5.2.** *Our modified scheme (see Figure 5 and Figure 6) satisfies the security notion of **auth** with any  $H$  that has correlation robustness for naturally derived keys.*

*Proof (Proof Sketch).* We execute the simulator  $S$  (in Figure 8 with the modifications in Figure 9), and obtain  $(\hat{F}, \hat{X}, \hat{d})$ . Then, we hand  $(\hat{F}, \hat{X})$  to the adversary, and receive  $\hat{Y}'$  from the adversary. After that we run the decoding procedure (see procedure **De** in Figure 6) on  $\hat{d}$  and the output of adversary  $\hat{Y}'$ . If the result is  $\text{De}(\hat{d}, \hat{Y}') = \perp$ , then the adversary fails, otherwise it succeeds. The adversary can win the game by guessing a correct value  $r$  with probability at most  $1/2^\lambda$  where  $\lambda$  is the security parameter. The rest of the proof utilizes the same sequence of hybrids in the proof of Theorem 5.1.

## 6 Performance Comparison

In this section, we evaluate the performance of our protocol, and compare it with Mohassel and Sadeghian's 2-party PFE scheme [MS13]. Without loss of generality, in order for a fair comparison, we assume that the 2-OEP protocol of our scheme is also realized by the OSN construction in [MS13], and that the OSN phases in both protocols are optimized with the General OT extension scheme of Asharov, Lindell, Schneider, and Zohner [ALSZ13]. Similar results can be obtained by using other Ishai *et al.* based OT extension schemes [IKNP03, KK13] as well.

Regarding the OSN phase, the total number of OTs in our 2-PFE protocol is  $2N \log(N) - N + 1$ , while it is  $2N \log(N) + 1$  in [MS13] (see Section 3.1). Moreover, our protocol requires only one of the tokens on a wire entering the OSN phase, so the size of the rows in Table 3 which enter each OT is reduced by a factor of two [MS13], further resulting in a significant decrease in communication cost. Regarding the 2PC phase,

**Table 5.** Analysis of communication costs for 2-PFE schemes (see Section 3.1 for details of transfers in the OSN phases).

			MS'13 [MS13]		Our Protocol	
			Num. of Strings	Str. Length (bits)	Num. of Strings	Str. Length (bits)
OSN	Before OT Ext.	$P_2 \rightarrow P_1$	$N$	$2\lambda$	$N$	$\lambda$
	During OT Ext.	$P_1 \rightarrow P_2$	$\lambda$	$2N \log(N) + 1$	$\lambda$	$2N \log(N) - N + 1$
		$P_2 \rightarrow P_1$	$4N \log(N) - N + 2$	$2\lambda$	$4N \log(N) - 2N + 2$	$\lambda$
	After OT Ext.	$P_1 \rightarrow P_2$	$N$	$2\lambda$	$N$	$\lambda$
2PC	Garbled Circ.	$P_2 \rightarrow P_1$	$2N$	$\lambda$	$1.5N$	$\lambda$
<b>TOTAL (bits)</b>			$(10N \log(N) + 4N + 5)\lambda$		$(6N \log(N) + 0.5N + 3)\lambda$	

**Table 6.** Communication cost comparison of 2-PFE schemes in terms of  $\lambda$ -bits.

Num. of Gates	MS'13 [MS13]			Our Protocol			Overall Reduction
	OSN Phase	2PC Phase	Total	OSN Phase	2PC Phase	Total	
$2^8$	47,109	1,024	48,133	27,139	768	27,907	42.0%
$2^{10}$	229,381	4,096	233,477	133,123	3,072	136,195	41.7%
$2^{12}$	1,081,349	16,384	1,097,733	630,787	12,288	643,075	41.4%
$2^{14}$	4,980,741	65,536	5,046,277	2,916,355	49,152	2,965,507	41.2%
$2^{16}$	22,544,389	262,144	22,806,533	13,238,275	196,608	13,434,883	41.1%
$2^{18}$	100,663,301	1,048,576	101,711,877	59,244,547	786,432	60,030,979	41.0%
$2^{20}$	444,596,229	4,194,304	448,790,533	262,144,003	3,145,728	265,289,731	40.9%

our scheme garbles each non-output gate with three ciphertexts, and each output gate with two ciphertexts. This yields more than 25% reduction compared to the same phase in the scheme in [MS13].

Table 5 shows the number of strings and their corresponding lengths sent in each turn in both schemes (see also Section 3.1 for details of transfers in the OSN phases). We omit the OTs for  $P_1$ 's garbled input, the transfers for decoding the garbled output, and the base OTs in the OT extension scheme [ALSZ13]. The strings sent by  $P_2$  during the OT extension in [MS13], in fact, consists of  $4N \log(N) - 2N + 2$  of  $\lambda$ -bit strings and  $2N$   $\lambda$ -bit strings. The data sent by  $P_2$  before OT extension can also be sent during  $P_2$ 's turn in OT extension for a saving in the number of rounds. Table 6 reflects the communication cost reduction achieved by our 2-PFE protocol for the circuits with different number of gates.

Recently, in [WmM17], Wang and Malluhi have attempted to improve the 2-PFE scheme in [MS13] by removing only one ciphertext from each garbled gate (in 2PC phase) while remaining the cost of OSN phase unchanged. However, the influence of 2PC phase in [MS13] on overall communication cost is quite low (see Table 6). Reducing the bandwidth use in the 2PC phase by 25% only results in less than 1% reduction in the total cost. For instance, given a circuit with 1024 gates, their optimization reduces the communication cost of the 2PC phase from 4,096  $\lambda$ -bit strings to 3,072 of them, while the OSN phase cost remains 229,38  $\lambda$ -bits. Therefore, the overall gain from their optimization for this setting is  $\sim 0.4\%$ .

Considering the computational complexity, although both schemes asymptotically require  $O(N \log(N))$  operations, our scheme achieves a linear time improvement over [MS13]. More precisely, in OSN phase, our scheme eliminates  $N$  oblivious transfer (OT) operations. This results in a decrease of  $2N$  symmetric encryptions performed by  $P_2$  ( $P_1$ 's computation cost remains the same in this phase). Regarding the 2PC phase, our scheme requires one additional operation per gate (during the Ev procedure). This yields additional  $0.5N$  symmetric operations to be performed by  $P_1$  ( $P_2$ 's computation cost remains the same in this phase). Therefore, our scheme reduces the overall computation cost by  $1.5N$  symmetric operations.

The round complexity of our scheme does not differ from the 2-PFE scheme in [MS13]. Namely, both protocols consist of a constant-round OT extension scheme in OSN phase, and our 2PC phase consists of the same number of rounds as in the garbling scheme used in [MS13].

## 7 Conclusion

In this paper, we proposed an efficient and secure protocol for 2-PFE. The motivation behind our work is that the bandwidth of various channels is the main constriction for many secure computation applications, including the ones for PFE. Our optimization significantly improves Mohassel and Sadeghian’s 2-PFE scheme [MS13] in both OSN and 2PC phases in terms of communication complexity. In particular, in OSN phase, our protocol reduces the number of required OTs and data sizes entering the protocol. In 2PC phase, our half gate based scheme garbles each non-output gate with three ciphertexts, and each output gate with two ciphertexts. All in all, our protocol improves the state-of-the-art by saving more than 40% of the overall communication cost. We conclude with the following two open questions:

1. Although the 2-OEP protocol in [MS13], which we utilize in our protocol, is quite efficient for many circuit sizes, fails to be so in large-sized circuits due to its  $O(g \log(g))$  complexity. This fact arises the following question: *Can we have a 2-OEP protocol that has linear asymptotic complexity while also being efficient in small circuit sizes?*
2. Our 2-PFE protocol permits only one gate functionality (*e.g.*, NAND or NOR) in a boolean circuit. This yields another important future challenge: *Can we have a gate hiding mechanism in 2-PFE schemes permitting the use of various gates in logic circuit representations?*

## References

- ALSZ13. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the ACM Computer and Communications Security Conference (CCS '13)*, pages 535–548, Berlin, Germany, 4-8 November 2013. ACM, NY, USA.
- BBKL18. Osman Bicer, Muhammed Ali Bingol, Mehmet Sabir Kiraz, and Albert Levi. Highly efficient and reusable private function evaluation with linear complexity. Cryptology ePrint Archive, Report 2018/515, 2018. <https://eprint.iacr.org/2018/515>.
- BFK<sup>+</sup>09. Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. Secure evaluation of private linear branching programs with medical applications. In *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, pages 424–439, Saint-Malo, France, 21-23 September 2009. Springer-Verlag, Berlin, Heidelberg.
- BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'12)*, pages 784–796, Raleigh, North Carolina, USA, 16-18 October 2012. ACM, NY, USA.
- BMR90. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'90)*, pages 503–513, Baltimore, Maryland, USA, 13-16 May 1990. ACM, NY, USA.
- BMR16. Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pages 565–577, Vienna, Austria, 24-28 October 2016. ACM, NY, USA.
- CKKZ12. Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the “Free-XOR” technique. In *Proceedings of the Theory of Cryptography Conference (TCC'12)*, pages 39–53, Taormina, Sicily, Italy, 12-19 March 2012. Springer-Verlag Berlin, Heidelberg.
- EGL85. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, jun 1985.
- EIG85. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theor.*, 31(4):469–472, September 1985.
- FAZ05. Keith Frikken, Mikhail Atallah, and Chen Zhang. Privacy-preserving credit checking. In *Proceedings of the 6th ACM Conference on Electronic Commerce, (EC'05)*, pages 147–154, Vancouver, BC, Canada, 05-08 June 2005. ACM, New York.
- GKS17. Daniel Günther, Ágnes Kiss, and Thomas Schneider. More efficient universal circuit constructions. In *Proceedings of the Advances in Cryptology – ASIACRYPT 2017*, pages 443–470, Hong Kong, China, 3-7 December 2017. Springer-Verlag Berlin, Heidelberg.

- HEKM11. Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*, pages 35–50, San Francisco, CA, 8-12 August 2011. USENIX Association Berkeley, CA, USA.
- Hoal13. Viet Tung Hoang. *Foundations of Garbled Circuits*. PhD thesis, University of California Davis, 2013.
- IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Proceedings of the Advances in Cryptology – CRYPTO 2003*, pages 145–161. Springer, Berlin, Heidelberg, Santa Barbara, CA, USA, 17-21 August 2003.
- KK13. Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In *Proceedings of the Advances in Cryptology - CRYPTO 2013*, pages 54–70, Santa Barbara, CA, USA, 18-12 August 2013. Springer, Berlin, Heidelberg.
- KM11. Jonathan Katz and Lior Malka. Constant-round private function evaluation with linear complexity. In *Proceedings of the Advances in Cryptology – ASIACRYPT 2011*, pages 556–571, Seoul, South Korea, 4-8 December 2011. Springer, Berlin, Heidelberg.
- KMR14. Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In *Proceedings of the Advances in Cryptology – CRYPTO 2014*, pages 440–457, Santa Barbara, CA, USA, 17-21 August 2014. Springer, Berlin Heidelberg.
- KS08a. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages, and Programming (ICALP'08)*, pages 486–498, Reykjavik, Iceland, 06-13 July 2008. Springer Berlin, Heidelberg.
- KS08b. Vladimir Kolesnikov and Thomas Schneider. A practical universal circuit construction and secure evaluation of private functions. In *Proceedings of the Financial Cryptography and Data Security (FC'08)*, pages 83–97, Cozumel, Mexico, 28-31 January 2008. Springer-Verlag Berlin Heidelberg.
- KS16. Ágnes Kiss and Thomas Schneider. Valiant’s universal circuit is practical. In *Proceedings of the Advances in Cryptology – EUROCRYPT 2016*, pages 699–728, Vienna, Austria, 8-12 May 2016. Springer-Verlag Berlin, Heidelberg.
- LMS16. Helger Lipmaa, Payman Mohassel, and Saeed Sadeghian. Valiant’s universal circuit: Improvements, implementation, and applications. Cryptology ePrint Archive, Report 2016/017, 2016. <http://eprint.iacr.org/2016/017>.
- LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.
- MS13. Payman Mohassel and Saeed Sadeghian. How to hide circuits in mpc an efficient framework for private function evaluation. In *Proceedings of the Advances in Cryptology – EUROCRYPT 2013*, pages 557–574, Athens, Greece, 26-30 May 2013. Springer, Berlin, Heidelberg.
- MSS14. Payman Mohassel, Saeed Sadeghian, and Nigel P. Smart. Actively secure private function evaluation. In *Proceedings of the Advances in Cryptology – ASIACRYPT 2014*, pages 486–505, Kaoshiung, Taiwan, 7-11 December 2014. Springer Berlin, Heidelberg.
- NPS99. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the ACM Conference on Electronic Commerce (EC'99)*, pages 129–139, Denver, Colorado, USA, 3-5 November 1999. ACM Press, NY, USA.
- NSMS14. Salman Niksefat, Babak Sadeghiyan, Payman Mohassel, and Seyed Saeed Sadeghian. ZIDS: A privacy-preserving intrusion detection system using secure two-party computation protocols. *Comp. J.*, 57(4):494–509, 2014.
- Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the Advances in Cryptology – EUROCRYPT 1999*, pages 223–238, Prague, Czech Republic, 2-6 May 1999. Springer Berlin, Heidelberg.
- PKV<sup>+</sup>14. Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos Keromytis, and Steve Bellovin. Blind Seer: A scalable private DBMS. In *Proceedings of IEEE Symposium on Security and Privacy (SP'14)*, pages 359–374, San Jose, CA, 18-21 May 2014. IEEE Computer Society, Washington, DC.
- PSS09. Annika Paus, Ahmad-Reza Sadeghi, and Thomas Schneider. Practical secure evaluation of semi-private functions. In *Proceedings of the Applied Cryptography and Network Security (ACNS'09)*, pages 89–106, Paris-Rocquencourt, France, 2-5 June 2009. Springer-Verlag Berlin, Heidelberg.
- PSSW09. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Proceedings of the Advances in Cryptology – ASIACRYPT 2009*, pages 250–267, Tokyo, Japan, 6-10 December 2009. Springer-Verlag Berlin, Heidelberg.
- Rab81. Michael O. Rabin. How to exchange secrets with oblivious transfer. Harvard University Technical Report. Available at Cryptology ePrint Archive, Report 2005/187, 1981. <http://eprint.iacr.org/2005/187>.

- Sad15. SeyedSaeed Sadeghian. *New Techniques for Private Function Evaluation*. PhD thesis, University of Calgary, 2015. <https://prism.ucalgary.ca/handle/11023/2657>.
- Sch08. Thomas Schneider. Practical secure function evaluation. Master's thesis, Friedrich-Alexander University Erlangen-Nürnberg, Germany, February 27, 2008. <http://thomaschneider.de/papers/S08Thesis.pdf>.
- SS09. Ahmad-Reza Sadeghi and Thomas Schneider. Generalized universal circuits for secure evaluation of private functions with application to data classification. In *Proceedings of the Information Security and Cryptology (ICISC'08)*, pages 336–353, Seoul, Korea, 3-5 December 2009. Springer-Verlag, Berlin, Heidelberg.
- Val76. Leslie G. Valiant. Universal Circuits (Preliminary Report). In *Proceedings of the ACM Symposium on Theory of Computing (STOC'76)*, pages 196–203, Hershey, Pennsylvania, USA, 3-5 May 1976. ACM New York, NY, USA.
- Wak68. Abraham Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, January 1968.
- WmM17. Yongge Wang and Qutaibah m. Malluhi. Reducing garbled circuit size while preserving circuit gate privacy. Cryptology ePrint Archive, Report 2017/041, 2017. <http://eprint.iacr.org/2017/041>.
- 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, Chicago, IL, 3-5 November 1982. IEEE Computer Society, Washington, DC, USA.
- ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In *Proceedings of the Advances in Cryptology - EUROCRYPT 2015*, pages 220–250, Sofia, Bulgaria, 26 - 30 April 2015. Springer, Berlin Heidelberg.