

EFFICIENT AND SECURE SCHEMES FOR  
PRIVATE FUNCTION EVALUATION

by

MUHAMMED ALİ BİNGÖL

Submitted to the Institute of Engineering and Natural Sciences  
in partial fulfillment of  
the requirements for the degree of  
Doctor of Philosophy

Sabanci University

January 2019

EFFICIENT AND SECURE SCHEMES FOR  
PRIVATE FUNCTION EVALUATION

APPROVED BY:

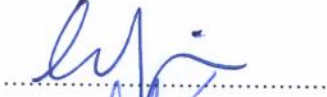
Prof. Albert Levi  
(Dissertation Supervisor)



Prof. Erkey Savaş



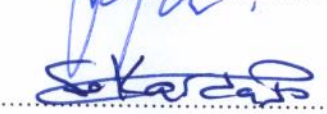
Prof. Cem Güneri



Assoc. Prof. Mehmet Sabır Kiraz



Asst. Prof. Süleyman Kardeş



DATE OF APPROVAL: 04/01/2019

© Muhammed Ali Bingöl 2019  
All Rights Reserved

# ABSTRACT

## EFFICIENT AND SECURE SCHEMES FOR PRIVATE FUNCTION EVALUATION

MUHAMMED ALİ BİNGÖL

Ph.D. Dissertation, January 2019

**Supervisor:** Prof. Albert Levi

**Keywords:** Cryptographic Protocols, Private Function Evaluation, Secure Computation, Communication and Computation Complexity, Security Analysis.

Development of computing devices with the proliferation of the Internet has prompted enormous opportunities for cooperative computation. These computations could occur between trusted or partially trusted partners, or even between competitors. Secure multi-party computation (MPC) protocols allow two or more parties to collaborate and compute a public functionality using their private inputs without the need for a trusted third-party. However, the generic solutions for MPC are not adequate for some particular cases where the function itself is also sensitive and required to be kept private. Private function evaluation (PFE) is a special case of 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. Recently, designing efficient PFE protocols have been a challenging and attractive task for cryptography researchers.

In this dissertation, we mainly focus on improving *two-party private function evaluation* (2PFE) schemes. Our primary goal is enhancing the state-of-the-art by designing secure and cost-efficient 2PFE protocols for both symmetric and asymmetric cryptography based solutions. In this respect, we first aim to improve 2PFE protocols based on (mostly) symmetric cryptographic primitives. 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 2PFE scheme. Compared to their scheme, our resulting optimization significantly improves both underlying *oblivious extended permutation* (OEP) and secure *2-party computation* (2PC) protocols, and yields a more than 40% reduction in overall communication cost. We next propose a novel and highly efficient 2PFE scheme based on the decisional Diffie-Hellman (DDH) assumption. Our scheme consists of two protocols, one is utilized in the initial execution, and the other is in the subsequent runs. One of the novelties of our scheme over the state-of-the-art is that it results in a significant cost reduction when the same private function is evaluated more than once between the same or varying parties. To the best of our knowledge, this is the most efficient and the first 2PFE scheme that enjoys reusability feature. Our protocols achieve linear communication and computation complexities, and a constant number of rounds which is at most three (depending on the size of the inputs of the party that holds the function).

# ÖZET

## GİZLİ FONKSİYON DEĞERLENDİRME İÇİN VERİMLİ VE GÜVENLİ ŞEMALAR

MUHAMMED ALİ BİNGÖL

Doktora Tezi, Ocak 2019

**Danışman:** Prof. Dr. Albert Levi

**Anahtar Sözcükler:** Kriptografik Protokoller, Gizli Fonksiyon Değerlendirme, Güvenli Hesaplama, İletişim ve Hesaplama Karmaşıklığı, Güvenlik Analizi.

Hesaplama cihazlarının gelişmesi ve Internet'in yaygınlaşması ile birlikte işbirlikçi hesaplama için büyük imkanlar doğmuştur. Bir fonksiyon veya algoritma üzerinde ortak hesaplama ihtiyacı, birbirlerine güvenen, kısmen güvenen veya kesinlikle güvenmeyen taraflar arasında olabilmektedir. Literatürde güvenli *çok taraflı hesaplama* (İng. multi-party computation - MPC) olarak bilinen protokoller, iki veya daha fazla tarafın, güvenilir bir üçüncü tarafa ihtiyaç duymadan ortak bir fonksiyonu birlikte hesaplamalarına imkan sağlar. Ancak MPC için önerilen genel çözümler, fonksiyonun kendisinin de hassas olduğu ve gizli tutulması gerektiği bazı özel durumlar için yeterli değildir. *Gizli fonksiyon değerlendirme* (İng. private function evaluation - PFE) fonksiyonun yalnızca bir tarafça bilinmesine imkan sağlayan özel bir MPC durumuna karşılık gelir. PFE protokolleri, bir algoritma veya bir fonksiyonun gizlilik

seviyesi veya fikri mülkiyeti gibi nedenlerle gizli kalmasını gerektiren çeşitli problemler için çözüm sağlar. Son zamanlarda, verimli PFE protokollerinin tasarlanması, kriptografi araştırmacıları için zorlayıcı ve ilgi çeken bir alan haline gelmiştir.

Bu tez çalışmasında *iki taraflı gizli fonksiyon değerlendirme* (İng. two-party private function evaluation - 2PFE) protokollerinin geliştirilmesi hedeflenmiştir. Öncelikli hedefimiz, simetrik ve asimetrik şifreleme kategorilerinde güvenli ve daha verimli PFE protokolleri tasarlayarak literatürü bu alandaki çalışmalarımız ile geliştirmektir. Bu amaçla, ilk olarak simetrik kriptografik yapıtaşlarına dayalı 2PFE protokollerini geliştirmeyi amaçladık. Eurocrypt'13'te Mohassel ve Sadeghian tarafından sunulan ve bu kategorideki en iyi sonuçlar ortaya koyan PFE protokolünü ele aldık. İyi bilinen *yarım kapılı* karmaşık devreler tekniğininin (Zahur et al., Eurocrypt'15) 2PFE şemasına nasıl uyarlayıp kullanacağımızı gösterdik. Protokollerini karşılaştırdığımızda, sonuçta elde ettiğimiz optimizasyonumuz, hem *kayıtsız genişletilmiş permütasyon* (İng. oblivious extended permutation - OEP) hem de güvenli *iki taraflı hesaplama* (İng. two-party computation - 2PC) alt protokollerinin verimliliğini önemli ölçüde iyileştirmiş ve iletişim maliyetinde % 40'ın üzerinde verimlilik sağlamıştır. Bunun yanı sıra, *kararsal Diffie-Hellman* (İng. decisional Diffie-Hellman - DDH) varsayımına dayanan yeni ve özgün 2PFE şeması önermekteyiz. Şemamız, literatürdeki çalışmalarını önemli ölçüde geliştirmekle birlikte yeniden kullanılabilirlik özelliğini sunarak sonraki hesaplamalar için verimliliği oldukça artırır. Önerdiğimiz şemamız iki protokolden oluşmaktadır, birincisi fonksiyonunun ilk defa uygulamasında, ikincisi ise sonraki uygulamalarda kullanılır. Bildiğimiz kadarıyla, önermiş olduğumuz bu şema, literatürdeki en verimli ve yeniden kullanılabilirlik özelliğine sahip ilk 2PFE tasarımıdır. Önermiş olduğumuz protokoller lineer iletişim ve hesaplama karmaşıklıklarına sahipken protokollerin mesaj tur sayısı en fazla üçtür.

*to my beloved family*



## ACKNOWLEDGMENTS

I wish to thank all people who have helped and inspired me during my Ph.D. study. First of all, I would like to express my sincere gratitude to my dissertation advisor, Prof. Albert Levi, for his endless support, worthwhile guidance and invaluable patience throughout my Ph.D. studies. I am happy to have such a supportive supervisor and it has been a privilege to study under his guidance. I would also like to thank my dissertation committee members, Prof. ErKay Savaş and Prof. Cem Güneri, for their supports and invaluable feedbacks starting from my thesis proposal period. I am also indebted to the other members of my thesis jury, Assoc. Prof. Mehmet Sabır Kiraz and Asst. Prof. Süleyman Kardaş, for reviewing my dissertation and providing valuable suggestions and inquiries. Despite their busy schedule, I really appreciate their agreement to be members of my committee and letting my dissertation defense be a memorable moment.

I would like to thank to all my colleagues in TUBITAK BİLGEM, especially Soner Ay, Dr. Şenol İşçi and Mehmet Emin Gönen for their strong friendship. My deeply thanks to Atakan Arslan for his support and his great friendship over the years. Also many thanks go to Osman Biçer and again Assoc. Prof. Mehmet Sabır Kiraz for the brainstorming discussions lasting a whole day and invaluable contributions to this work. I would like to extend my gratitude to all of my Sabancı University professors and (past and present) friends & colleagues.

Last but not least my deepest gratitude goes to my wife Burcu, my kids Meryem & Kerem, and my parents for their unflagging love, patience and support throughout my life; this dissertation is basically impossible without them.

# TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Contributions . . . . .	3
1.3	Organization . . . . .	6
<b>2</b>	<b>BACKGROUND INFORMATION</b>	<b>7</b>
2.1	Cryptographic Primitives . . . . .	7
2.1.1	Symmetric and Asymmetric Cryptosystems . . . . .	7
2.1.2	Some Computational Problems . . . . .	8
2.1.3	Hash Functions . . . . .	10
2.1.4	Elliptic Curve Cryptography . . . . .	11
2.1.5	Homomorphic Encryption . . . . .	12
2.1.6	Oblivious Transfer Protocols . . . . .	13
2.2	Basics of Secure Computation and Garbled Circuits . . . . .	14
2.2.1	Yao's Garbled Circuit . . . . .	16
2.2.2	Optimizations on Yao's Scheme . . . . .	17
2.2.3	Adversary Types . . . . .	23
<b>3</b>	<b>RELATED WORKS</b>	<b>25</b>
3.1	Universal Circuit Based PFE Solutions . . . . .	26
3.2	Special Purpose PFE Solutions . . . . .	26

<b>4</b>	<b>AN EFFICIENT 2-PARTY PRIVATE FUNCTION EVALUATION PROTOCOL BASED ON HALF GATES</b>	<b>30</b>
4.1	2-Party PFE Framework . . . . .	30
4.1.1	Context of CTH . . . . .	31
4.1.2	Mohassel and Sadeghian’s 2PFE scheme . . . . .	37
4.2	Our Efficient 2-Party PFE Scheme . . . . .	38
4.2.1	Use of 2-OEP protocol . . . . .	41
4.2.2	Our 2PC Garbling Scheme for 2PFE . . . . .	42
4.3	Security of the proposed protocol . . . . .	45
4.3.1	Code based games and security notions . . . . .	45
4.3.2	Security Proof . . . . .	47
4.4	Performance Comparison . . . . .	52
<b>5</b>	<b>HIGHLY EFFICIENT AND REUSABLE PRIVATE FUNCTION EVALUATION WITH LINEAR COMPLEXITY</b>	<b>56</b>
5.1	Preliminaries . . . . .	56
5.1.1	Decisional Diffie-Hellman Assumption . . . . .	57
5.1.2	Notations and Concept of 2PFE Framework . . . . .	58
5.2	Our PFE Scheme . . . . .	60
5.2.1	The description of our <b>lnExe</b> protocol . . . . .	60
5.2.2	Optimization with reusability feature: Our ( <b>ReExe</b> ) protocol . . . . .	67
5.2.3	Executing with Various $\text{Party}_2\text{s}$ . . . . .	69
5.3	Complexity Analysis . . . . .	71
5.3.1	Complexity of Our Scheme . . . . .	72
5.3.2	Comparison . . . . .	73
5.4	Security of Our Protocols . . . . .	78
<b>6</b>	<b>CONCLUSIONS</b>	<b>85</b>

# LIST OF TABLES

2.1	Garbling an odd gate using half gates technique [1]. . . . .	22
4.1	Party <sub>1</sub> learns one of these rows according to his selection bits. . . . .	36
4.2	Party <sub>1</sub> gets one of these rows by engaging in 1-out-of-4 OT with Party <sub>2</sub> . . . . .	37
4.3	Adapting half gates technique to our 2PFE for garbling an odd gate. Here, $\alpha_1$ , $\alpha_2$ and $\alpha_3$ define the gate type ( <i>e.g.</i> , $\alpha_1 = 0$ , $\alpha_2 = 0$ and $\alpha_3 = 1$ for a <b>NAND</b> gate, see Equation (2.2)). The token $w_c^0$ on the output wire equals $w_{Gc}^0 \oplus w_{Ec}^0 \oplus \psi_c$ . The three ciphertexts $T_{Gc}$ , $T_{Ec}$ , and $\psi_c$ are sent to Party <sub>1</sub> for each gate. . . . .	41
4.4	Analysis of communication costs for 2PFE schemes (see Section 4.1.1 for details of transfers in the OSN phases). . . . .	53
4.5	Communication cost comparison of 2PFE schemes in terms of $\lambda$ -bits. . . . .	54
5.1	Comparison of the existing 2PFE schemes 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. $M$ , $N$ , $\lambda$ , and $\rho$ denote the number of outgoing wires (i.e., equal to $n + g - m$ ), the number of incoming wires (i.e., $N = 2g$ ), the security parameter, and the computation cost ratio, respectively. . . . .	72
5.2	Comparison of the existing 2PFE schemes in terms of overall communication costs for various circuit sizes. Here we take $N = 2M$ and $\lambda = 128$ . . . . .	74

5.3 Our efficiency gain (in percentage) over existing 2PFE schemes in terms of overall communication costs with respect to the number of protocol runs. . . . . 77

# LIST OF FIGURES

4.1	(a) A circuit representation $\mathcal{C}_f$ of a function $f$ . (b) The mapping $\pi_f$ of $f$ . . . . .	32
4.2	The related switching network for the mapping $\pi_f$ in Figure 4.1. . . .	34
4.3	Components and high level procedures of our PFE protocol. The private function $f$ is only known to <b>Party</b> <sub>1</sub> . <b>Party</b> <sub>1</sub> compiles $f$ into a Boolean circuit $\mathcal{C}_f$ , and extracts the mapping $\pi_f$ and the template of private circuit $\tilde{\mathcal{C}}_f$ . <b>Party</b> <sub>1</sub> sends $\tilde{\mathcal{C}}_f$ to <b>Party</b> <sub>2</sub> . <b>Party</b> <sub>1</sub> randomly generates the vector $\mathcal{T}$ . <b>Party</b> <sub>2</sub> randomly generates the vector $W^0$ . They engage in a 2-OEP protocol where <b>Party</b> <sub>2</sub> learns $S^0$ as the output. With the knowledge of $W^0$ , $S^0$ and $\tilde{\mathcal{C}}_f$ , <b>Party</b> <sub>2</sub> garbles each gate and sends the garbled circuit to <b>Party</b> <sub>1</sub> . With the knowledge of $\pi_f$ , $\tilde{\mathcal{C}}_f$ , $\mathcal{T}$ , the garbled circuit and the garbled inputs, <b>Party</b> <sub>1</sub> evaluates the whole garbled circuit. . . . .	39
4.4	Our complete half gate based garbling scheme for 2PFE. $\mathbf{Gb}_{\text{NAND}}$ and $\mathbf{Gb}_{\text{NAND}}^*$ are the original half gate and our modified NAND garbling procedures, respectively. A ‘ <i>hat</i> ’ represents a sequence or a tuple, for instance, $\hat{F} = (F_1, F_2, \dots)$ or $\hat{e} = (e_1, e_2, \dots)$ . . . . .	43
4.5	Modification of our garbling scheme in Figure 4.4 for achieving authenticity ( <b>auth</b> ) property. . . . .	44

4.6	Components of and high level procedures of a OEP based Private Function Evaluation scheme. The topology hiding of the function $f$ where $\text{Party}_1$ is the evaluator and $\text{Party}_2$ is the garbler: (1) The private function $f$ is only known by $\text{Party}_1$ . (2) $\mathcal{C}_f$ is the Boolean circuit representation of $f$ . (3) $\pi_f$ is the circuit mapping of $f$ . (4) The OEP protocol is mutually run where $\text{Party}_2$ learns blinded strings. (5) The blinded strings learnt by $\text{Party}_2$ . (6) Yao's protocol with the blinded strings. . . . .	44
4.7	Simulation based games for privacy, obliviousness and authenticity [2]. The function $\mathcal{S}$ is a simulator, and $G$ denotes a garbling scheme. . . .	46
4.8	Part-A. The simulator for $\text{prv.sim}_{\mathcal{S}}$ security, and the hybrids 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 $\hat{e} = (e_1, e_2, \dots)$ . . . . .	48
4.9	Part-B. The simulator for $\text{prv.sim}_{\mathcal{S}}$ security, and the hybrids used in the proof. A 'hat' represents a sequence or a tuple, for instance, $\hat{F} = (F_1, F_2, \dots)$ or $\hat{e} = (e_1, e_2, \dots)$ . (Please see Figure 4.8 for the beginning of the figure.) . . . . .	49
4.10	The required modifications on Figure 4.8 in order to show $\text{auth}$ property.	51
5.1	Sketch of our $\text{InExe}$ 2PFE Protocol. $\text{ReuseTemp}_f$ and $T$ are stored (if needed) for the later PFE runs by $\text{ReExe}$ protocol. Note that in case $\text{Party}_1$ has inputs $(x_1)$ then OT protocol is required (to send the corresponding garbled $X_1$ ) which can be trivially combined with the protocol rounds for minimization of the total number of rounds. . . .	61
5.2	Our Optimized $\text{InExe}$ 2PFE Protocol via decomposition of offline/online computations . . . . .	63

5.3	Sketch of our ReExe protocol for the $k$ -th execution ( $k > 1$ ). The number of rounds is equal to 1, or 2, or 3 depending on the input size of Party <sub>1</sub> . . . . .	68
5.4	Our Optimized ReExe 2PFE Protocol that utilizes Reusable Mapping Template. . . . .	70
5.5	Comparison of cumulative communication cost via normalized bandwidth efficiency vs. number of PFE executions using a circuit $2^{10}$ gates. . . . .	76
5.6	Comparison of cumulative communication cost via normalized bandwidth efficiency vs. number of PFE executions using a circuit $2^{30}$ gates. . . . .	76



# LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
CDH	Computational Diffie-Hellman
CPU	Central Processing Unit
CTH	Circuit Topology Hiding
DDH	Decisional Diffie-Hellman
DKC	Dual-Key Cipher
DLP	Discrete Logarithm Problem
DNA	Deoxyribonucleic Acid
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
EP	Extended Permutation
FHE	Fully Homomorphic Encryption
gcd	Greatest Common Divisor
GG	Garbled Gate
GRR	Garbled Row Reduction
H	Hash Function
HE	Homomorphic Encryption

lsb	Least Significant Bit
MPC	(Secure) Multi-Party Computation
NAND	Not AND
NIST	National Institute of Standards and Technology
NOR	Not OR
NCE	Normalized Cost Efficiency
OEP	Oblivious Extended Permutation
OSN	Oblivious Evaluation of Switching Network
OT	Oblivious Transfer
PFE	Private Function Evaluation
PGE	Private Gate Evaluation
PHE	Partially (Singly) Homomorphic Encryption
PN	Permutation Network
PPT	Probabilistic Polynomial Time
SHA	Secure Hash Algorithm
SN	Switching Network
SWHE	Somewhat Homomorphic Encryption
XOR	Exclusive OR
2PC	Two-Party Computation
2PFE	Two-Party Private Function Evaluation

# LIST OF SYMBOLS

## *An Efficient 2PFE Protocol Based On Half Gates (Chapter 4)*

$x_1$	Private input of Party <sub>1</sub>
$x_2$	Private input of Party <sub>2</sub>
$X_1$	Garbled version of $x_1$
$X_2$	Garbled version of $x_2$
$f$	The private function of Party <sub>1</sub> to be evaluated
$y$	The output of function $f$ such that $y = f(x_1, x_2)$
$\mathcal{C}_f$	The Boolean circuit representation of function $f$
$n$	Number of inputs of $\mathcal{C}_f$
$m$	Number of outputs of $\mathcal{C}_f$
$g$	Number of gates of $\mathcal{C}_f$
$\lambda$	Security parameter
$G_i$	The $i$ -th gate of circuit $\mathcal{C}_f$
$ow_i$	The $i^{th}$ outgoing wire
$iw_i$	The $i^{th}$ incoming wire
$N$	Number of incoming wires (i.e., $N = 2g$ )

$M$	Number of outgoing wires (i.e., $M = n + g - m$ )
$\text{OW}$	Set of outgoing wires ( $\text{ow}_1, \dots, \text{ow}_{n+g-m}$ ) which is the union of the input wires of $\mathcal{C}_f$ and the output wires of its non-output gates
$\text{IW}$	Set of incoming wires ( $\text{iw}_1, \dots, \text{iw}_{2g}$ ) which is the input wires of each gate in the circuit (having $N = 2g$ elements)
$\pi_f$	The private mapping from $\text{OW}$ to $\text{IW}$
$\tilde{\mathcal{C}}_f$	Template of Private Circuit
$Y$	Garbled version of $y$
$\rho$	Number of possible circuit topologies
$w_i^0, w_i^1$	The $i$ -th garbled tokens for each $\text{ow}_i \in \text{OW}$ corresponding to FALSE and TRUE semantic values, respectively.
$W$	Garbled vector set for outgoing wires
$t_j^0, t_j^1$	Blinding strings for each $\text{iw}_j \in \text{IW}$ corresponding to FALSE and TRUE semantic values, respectively.
$\mathcal{T}$	Blinding vector for incoming wires
$\sigma_j$	the blinded strings for incoming wires such that $[\sigma_j = w_{\pi_f^{-1}(j)} \oplus t_j]$ for $j = 1, \dots, N$
$S$	SN's blinded output vector for incoming wires
$R$	Circuit-wise offset value
$T_G, T_E$	Garbler's and evaluator's half gates, respectively
$\hat{x}$	A 'hat' represents a sequence or a tuple, for instance, $\hat{x} = (x_1, x_2, \dots)$
$F$	The garbled version of $\tilde{\mathcal{C}}_f$

$e$	Encoding information
$d$	Decoding information
Gb	Garble procedure: takes a function $f$ and a security parameter $1^\lambda$ as input and outputs $(F; e; d)$
En	Encode procedure: takes an input $x$ and encoding information $e$ and outputs a garbled input $X$
Ev	Evaluate procedure: takes a garbled circuit $F$ and garbled input $X$ and outputs a garbled output $Y$ .
De	Decode procedure: takes a garbled output $Y$ and decoding information $d$ and outputs a plain circuit-output $y$ if the decoding successful, otherwise returns error
ev	Evaluation function: used to check the <i>correctness</i> condition such that $\text{ev}(f, x) = \text{De}(d, \text{Ev}(F, \text{En}(e, x)))$

## ***Highly Efficient and Reusable PFE with Linear Complexity (Chapter 5)***

*(The first nineteen symbols of Chapter 4 are common)*

$\text{PubInfo}_{C_f}$	Public information of the circuit $C_f$ (i.e., $(M, N, \text{OW}, \text{IW}, y)$ )
$\mathbb{G}$	A cyclic group of a large prime order $q \in O(\lambda)$
$P_i$	The $i$ -th generator of the group $\mathbb{G}$ picked for outgoing wires $\text{ow}_i$ by $\text{Party}_2$ where $i = 1 \dots M$
$\mathcal{P}$	Set of generators picked for outgoing wires (i.e., $(P_1, \dots, P_M)$ )
$\ell$	Bit length of a group element
$t_j$	The $j$ -th blinding strings where $j = 1, \dots, N$
$Q_j$	The $j$ -th group element generated for $\text{iw}_j$ by $\text{Party}_1$ such that $Q_j := t_j \cdot P_{\pi_f^{-1}(j)}$

$\mathcal{Q}$	Set of group elements for wires (i.e., $(Q_1, \dots, Q_N)$ )
ReuseTemp <sub>f</sub>	Reusable mapping template (i.e., $(\mathcal{P}, \mathcal{Q})$ )
$\alpha_0, \alpha_1$	Randomly chosen strings in $\mathbb{Z}_q^*$ for the wires with semantic values 0 and 1, respectively
$W_i^b$	The $i$ -th group element computed as $(W_i^b \leftarrow \alpha_b \cdot P_i)$ where $b = \{0, 1\}$
$\mathcal{W}^b$	The ordered set of the group elements $(W_1^b, \dots, W_M^b)$
$V_j^b$	The $j$ -th group element computed as $(V_j^b \leftarrow \alpha_b \cdot Q_j)$
$\mathcal{V}^b$	The ordered set of the group elements $(V_1^b, \dots, V_N^b)$
$Y^b$	The ordered set of output values such that $(y_1^b, \dots, y_m^b : y_i^b \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m)$ where $b = \{0, 1\}$

# Chapter 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>1</sup>. A PFE solution would

---

<sup>1</sup>Note that PFE also covers the case where the party who owns the function does not have any other private input.

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, or the implementation of the function (say  $C_f$ ) is an intellectual proprietary albeit the function  $f$  is public [3–11]. Efficient and practical PFE schemes are becoming increasingly important as many applications require protection of their valuable assets such as private database management systems [12], privacy-preserving intrusion detection system [13], privacy-preserving checking for creditworthiness [7] and privacy preserving medical applications [11]. Therefore, the task of designing efficient custom PFE protocols for special or general purposes is addressed in several papers in the literature [9, 14–21].

## 1.1 Motivation

The task of designing secure and efficient PFE protocols is 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 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 [22, 23], but, unfortunately, it is still far from being practical [24]. 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, PFE schemes are 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.

Moreover, Lipmaa *et al.* [25] and Sadeghian [26] mention this open problem: “*the various optimizations that are recently proposed for MPC [1, 27, 28] 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)*” (see [26, p. 98] and [25, p. 2]). One of the aims of this dissertation is providing an answer to this open question and come up with an efficient 2PFE protocol.

Furthermore, 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 [29]. 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.

## 1.2 Contributions

The results of this dissertation substantially improve the state-of-the-art by proposing more efficient PFE schemes in both symmetric and asymmetric cryptography categories. The major contributions of this thesis are summarized as follows:

We first focus on improving 2-party private function evaluation (2PFE) based on symmetric cryptographic primitives. In this respect, we first revisit the state-of-the-art Mohassel and Sadeghian’s PFE framework [17], then propose a more efficient protocol (secure in the presence of semi-honest adversaries) by adapting the half gates garbling optimization [1] to their 2PFE scheme. Note that in [30], Wang and Malluhi mention that “*free-XOR [27] and half gates [1] techniques cannot be used to improve the efficiency of non-universal circuit based custom PFE protocols such as Katz and Malka’s [9] and Mohassel and Sadeghian’s [17] 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. Our protocol in this category 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 [17] requires  $2N \log(N) + 1$  OTs, while our protocol requires  $2N \log(N) - N + 1$  OTs. (2) Our protocol reduces the data sizes entering 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>2</sup>, which is a significant improvement compared to [17], whose communication cost is  $(10N \log(N) + 4N + 5)\lambda$  bits. This means more than 40% saving in bandwidth size (see Table 4.4 and Table 4.5). Also, the overall computation cost is also slightly decreased while the number of rounds remains unchanged. We show that our resulting 2PFE scheme is secure in the semi-honest model.

We also propose a highly efficient 2PFE scheme for Boolean circuits based on the DDH assumption which utilizes asymmetric cryptographic primitives. 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 subsequent 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 [9] and MS13 [17] since

---

<sup>2</sup> $\lambda$  is the security parameter throughout this thesis.

they require running the whole protocol from scratch for each execution. In this respect, we present two protocols of our scheme: (1) a protocol for initial executions (**InExe**), (2) a resumption protocol for subsequent executions (**ReExe**). The former protocol is utilized in the first evaluation of the function, while the latter one is utilized in the second or later subsequent 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 **InExe** protocol. 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.

Our proposed protocols significantly enhance the state-of-the-art in terms of communication cost. Compared to MS13-OSN [17], BBKL18 [20], and GKS17 [19] protocols, our scheme asymptotically reduces the communication cost. Namely, while the asymptotic communication costs of those protocols are equal to  $O(g \log(g))$ , our scheme provides  $O(g)$  communication complexity where  $g$  is the number of gates. To illustrate the significance of this asymptotic difference, for a thousand-gate circuit, our cost reduction is about 94% over MS13-OSN, about 88% over BBKL18, and about 68% over GKS17. For a billion-gate circuit, our cost reduction is about 98% over MS13-OSN, about 96% over BBKL18, and about 89% over GKS17. The protocols of MS13-HE, KM11-1st, KM11-2nd and ours has linear asymptotic complexity. Thanks to the reusability feature, the advantage of our scheme becomes more conspicuous when the number of PFE execution is more than one. Namely, for two executions our cost reduction is about 54% over KM11-1st, 30% over KM11-2nd, and 20% over MS13-HE. For ten executions our cost reduction is about 63% over KM11-1st, 44% over KM11-2nd, and 37% over MS13-HE. The number of rounds of our **InExe** protocol is 3 and the number of rounds of our **ReExe** protocol is equal to 1, or 2, or 3 depending on the input string length of **Party**<sub>1</sub> (i.e., owner of  $f$ )<sup>3</sup>. This

---

<sup>3</sup>If **Party**<sub>1</sub> has  $x_1 = \perp$ , then the number of rounds is equal to 1. If **Party**<sub>1</sub> has a non-empty input  $x_1$  such that the OT extension is not applicable for its garbled input, then it is to 2. Otherwise, the number of rounds is equal to 3.

also reflects the improvement of ReExe protocol over the existing 2PFE protocols in terms of round complexity (see Table 5.1).

We also deal with the case that  $\text{Party}_1$  runs the 2PFE protocol for the same private function with various  $\text{Party}_2$ s separately. This is a common scenario where  $\text{Party}_1$  may run a business with many customers for her algorithm/software. Trivially, our ReExe protocol can be utilized between the same two parties in the second and subsequent evaluations after the first evaluation. Instead of running the initial execution protocol with each  $\text{Party}_2$ , we propose a more efficient mechanism for the generation of the reusable tokens by employing a threshold based system.

### 1.3 Organization

The organization of this dissertation is as follows: In Chapter 2, we give necessary background information about cryptographic primitives and secure computation & garbled circuits. In Chapter 3, we review the literature on existing PFE approaches. In Chapter 4, we introduce our (mostly) symmetric-based 2PFE scheme. This chapter provides the detailed explanation of our protocol then a simulation-based security proof of our scheme in the semi-honest model. Also, the chapter covers an analysis of our protocol in terms of communication and computation complexities and comparison with the state-of-the-art. Chapter 5 presents our highly efficient mechanism for improving asymmetric cryptography based 2PFE schemes. We describe our two new methods to achieve more efficient PFE between the two parties and in the presence of multiple  $\text{Party}_2$ s. Also, this chapter provides the complexities of our resulting protocols and compares them with the existing state-of-the-art 2PFE protocols. Finally, Chapter 6 concludes the dissertation and point out some future works.

# Chapter 2

## BACKGROUND INFORMATION

This chapter provides some background information on some general concepts and definitions. We begin this chapter by defining some cryptographic primitives that are used throughout this dissertation. Then we give a brief overview on basics of secure computation, Yao's garbled circuits and recent optimizations on Yao's scheme. Additional preliminaries and definitions, which are specific to some parts of the dissertation, appear within the related chapters.

### 2.1 Cryptographic Primitives

In this section, we give definitions of some cryptographic primitives that are utilized throughout this dissertation. Most of the definitions given in this section have become standard, and are widely used in cryptography.

#### 2.1.1 Symmetric and Asymmetric Cryptosystems

**Definition 2.1.1.** *Cryptosystem.* A cryptosystem is a quintuple  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  with the following properties:

1. Let  $\mathcal{P}$  be the plaintext space,  $\mathcal{C}$  be the ciphertext space, and  $\mathcal{K}$  be the keyspace, where  $\mathcal{P}$ ,  $\mathcal{C}$  and  $\mathcal{K}$  are finite sets.
2.  $E_k$  is an encryption function such that  $E_k : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$ , and  $D_k$  is an encryption function such that  $D_k : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P}$ , where  $k \in \mathcal{K}$ .
3. For each key  $k_e \in \mathcal{K}$ , there exists some key  $k_d \in \mathcal{K}$  such that for each plaintext  $p \in \mathcal{P}$ ,  $D_{k_d}(E_{k_e}(p)) = p$ .

A cryptosystem is said to be *symmetric*-key cryptosystem (or *private-key* cryptosystem) if either of the following holds: (1)  $k_d = k_e$  or (2)  $k_d$  can “easily” be determined from  $k_e$ . A cryptosystem is said to be *asymmetric* cryptosystem (or *public-key* cryptosystem) if  $k_d \neq k_e$  and it is “computationally infeasible” to determine the private key  $k_d$  from the corresponding public key  $k_e$ .

### 2.1.2 Some Computational Problems

We now give some standard definitions of computational hardness and some assumptions that are hitherto known as hard problems. A polynomial time Turing machine is one which halts within  $p(|x|)$  steps on any input string  $x$  with length  $|x|$  where  $p$  denotes some polynomials. A probabilistic Turing machine is allowed to make random choices in its execution such that on each step it chooses the next configuration randomly from the possible ones [31–33].

**Definition 2.1.2.** (*Probabilistic Polynomial Time (PPT) Turing Machine.*) A Turing machine  $\mathcal{M}$  is said to be a probabilistic polynomial time (PPT) Turing machine if it takes input  $x$  together with a string of random bits  $r$  and  $\exists c \in \mathbb{N}$  such that  $\mathcal{M}(x, r)$  always halts in  $x^c$  steps.

**Definition 2.1.3.** (*Negligible Function.*) A function  $\epsilon(x) : \mathbb{N} \mapsto \mathbb{R}$  is negligible if and only if  $\forall c \in \mathbb{N}$ ,  $\exists x_0 \in \mathbb{N}$  such that  $\forall x \geq x_0$

$$\epsilon(x) < \frac{1}{x^c}.$$

Let  $\mathcal{A}$  be a PPT algorithm and  $\epsilon(\cdot)$  is a negligible function. A problem is said to be “easy” if it can be solved by a PPT  $\mathcal{A}$  with respect to the size of the input.

Let  $(\mathbb{G}, \cdot)$  be a finite cyclic group  $\mathbb{G} = \langle g \rangle$  of order  $|\mathbb{G}|$  and  $g$  is a generator. For this given group the definitions of Discrete Logarithm Problem (DLP), Computational Diffie-Hellman (CDH) Problem and Decisional Diffie-Hellman (DDH) Problem are as follows.

**Definition 2.1.4.** (*Discrete Logarithm Problem (DLP).*) *The DLP states that for any PPT  $\mathcal{A}$ , there exists a negligible function  $\epsilon(\cdot)$  such that given  $g^x \in \mathbb{G}$ , the probability of finding  $x$  is*

$$\Pr[x \leftarrow \mathcal{A}(\langle g \rangle, g, g^x)] \leq \epsilon(|\mathbb{G}|).$$

**Definition 2.1.5.** (*Computational Diffie-Hellman (CDH) Problem.*) *The CDH problem states that for any PPT  $\mathcal{A}$ , there exists a negligible function  $\epsilon(\cdot)$  such that given  $g^x, g^y \in \mathbb{G}$  the probability of finding  $g^{xy}$  is*

$$\Pr[g^{xy} \leftarrow \mathcal{A}(\langle g \rangle, g, g^x, g^y)] \leq \epsilon(|\mathbb{G}|).$$

**Definition 2.1.6.** (*Decisional Diffie-Hellman (DDH) Problem.*) *The DDH problem states that for any PPT  $\mathcal{A}$ , there exists a negligible function  $\epsilon(\cdot)$  such that given  $g^x, g^y \in \mathbb{G}$  and  $\chi$ , the probability of distinguishing  $g^{xy}$  from a randomly chosen element  $g^r \in \mathbb{G}$  is*

$$\left| \Pr[\beta \leftarrow \mathcal{A}(\langle g \rangle, g, g^x, g^y, \chi)] - \frac{1}{2} \right| \leq \epsilon(|\mathbb{G}|), \text{ where } \chi = \begin{cases} g^{xy}, & \text{if } \beta = 0 \\ g^r, & \text{otherwise.} \end{cases}$$

### 2.1.3 Hash Functions

A hash function is a deterministic mapping [34, 35] defined as below.

**Definition 2.1.7.** (*Hash Function.*) A function  $H : \{0, 1\}^* \mapsto \{0, 1\}^\ell$ , mapping arbitrary-length bit strings to a fixed length  $\ell$ -bit strings is called a hash function, where  $\ell \in \mathbb{Z}$  and  $\ell \geq 0$ . Function  $H$  is called a cryptographic hash function that satisfies the following properties:

- **Computability:** For any given input  $x \in \{0, 1\}^*$ ,  $y = H(x)$  is computed in a polynomially bounded time.
- **One-wayness:** Given an  $\ell$ -bit string  $y$ , for any PPT  $\mathcal{A}$ , there exist a negligible function  $\epsilon(\cdot)$  such that

$$\Pr[y \in \{0, 1\}^\ell; x \leftarrow \mathcal{A}(1^\ell, H, y) : H(x) = y] \leq \epsilon(\ell).$$

*This property is also known as “pre-image resistance”.*

- **2nd pre-image resistance:** Given a bit string  $x$ , for any PPT  $\mathcal{A}$ , there exist a negligible function  $\epsilon(\cdot)$  such that

$$\Pr[x_1 \leftarrow \{0, 1\}^*; y_1 = H(x_1); x_2 \leftarrow \mathcal{A}(1^\ell, H, y_1) : x_1 \neq x_2 \wedge H(x_1) = H(x_2)] \leq \epsilon(\ell).$$

*This property is also known as “weak collision resistance”.*

- **Collision resistance:** For any PPT  $\mathcal{A}$ , there exist a negligible function  $\epsilon(\cdot)$  such that

$$\Pr[(x_1, x_2) \leftarrow \mathcal{A}(1^\ell, H) : x_1 \neq x_2 \wedge H(x_1) = H(x_2)] \leq \epsilon(\ell).$$

*This property is also known as “strong collision resistance”.*



In general, collision resistance (strong collision resistance) implies 2nd pre-image resistance (weak collision resistance) but collision resistance need not imply one-wayness [32, 34, 36, 37]. We treat cryptographic hash functions as *random oracles* as they satisfy the following definition.

**Definition 2.1.8.** (*Random Oracles.*) A function  $H : \{0, 1\}^* \mapsto \{0, 1\}^\ell$  is said to be a random oracle if given any PPT  $\mathcal{A}$ , there exist a negligible function  $\epsilon(\cdot)$  such that

$$\left| \Pr[\beta \leftarrow \mathcal{A}(y)] - \frac{1}{2} \right| \leq \epsilon(\ell), \text{ where } y = \begin{cases} H(x), & \text{if } \beta = 0 \\ r \in \{0, 1\}^\ell, & \text{otherwise.} \end{cases}$$

Namely, in the random oracle model, a cryptographic hash function  $H$  viewed as a random oracle that responds to every query with a random response chosen uniformly from  $\{0, 1\}^\ell$  [38, 39].

## 2.1.4 Elliptic Curve Cryptography

Let  $\mathbb{F}_p$  be a finite field with  $p > 3$  a large prime. Also let  $E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b \text{ where } a, b \in \mathbb{F}_p \text{ with } 4a^3 + 27b^2 \neq 0\} \cup \{\mathcal{O}\}$ , where  $\mathcal{O}$  denotes the point at infinity. In general, for security purposes, the order of  $E(\mathbb{F}_p)$  has a large prime factor and a few other small factors, called cofactors. The order of  $E(\mathbb{F}_p)$  must be  $kq$  where  $q$  is a large prime, and  $k$  is the cofactor. Let  $\mathbb{G}$  be a cyclic subgroup of large prime order  $q$  of  $E(\mathbb{F}_p)$ . The security of the system is based on the intractability of the discrete logarithm problem (DLP) in the subgroup  $\mathbb{G}$ .

A base point (generator) of the group  $\mathbb{G}$  can be found by first finding a random element  $x_0 \in \mathbb{F}_p$  such that  $y_0^2 = x_0^3 + ax_0 + b$  for some  $y_0 \in \mathbb{F}_p$ , then multiplying it by the cofactor  $k$  as  $P := k \cdot (x_0, y_0)$ . Thanks to the Lagrange theorem, if  $P \neq \mathcal{O}$ , then it is a base point of order  $q$ . For the other generators of the group, just pick a random element  $r_i \in \mathbb{Z}_q^*$ , then  $P_i := r_i \cdot P$  is also another base point of the group  $\mathbb{G}$  (due to the fact that  $\gcd(r_i, q) = 1, \forall r_i \in \mathbb{Z}_q^*$ ) [40–44].

Throughout this dissertation, points on an elliptic curve are represented by capital letters while scalars are represented by lower-case letters.

## 2.1.5 Homomorphic Encryption

*Homomorphic Encryption* is a form of cryptosystem as defined below.

**Definition 2.1.9.** (*Homomorphic Encryption.*) *The encryption algorithm  $E$  is homomorphic if given any two encryptions  $E(p_1)$  and  $E(p_2)$ , one can obtain  $E(p_1 \star p_2)$  without decrypting the ciphertexts  $E(p_1)$  and  $E(p_2)$  for some operation “ $\star$ ” and  $\forall p_1, p_2 \in \mathcal{P}$ .*

In general, the operation  $\star$  is either addition or multiplication because these operations are functionally complete sets over finite sets. Homomorphic encryption systems are useful cryptographic tools since it allows operations on encrypted data as if it had been performed on the plaintexts without the need for the decryption key. Such cryptosystems have natural applications in privacy-preserving, secure computations. The homomorphic encryption schemes can be addressed in three categories with respect to the number of applicable operations on the encrypted message: (i) Partially (singly) HE (PHE), (ii) somewhat HE (SWHE) and (iii) fully HE (FHE) [45]. In PHE, only one type of operation is allowed without a bound on the number of operation calls. In literature, there are many cryptosystems that have PHE property or especially proposed to be so [46–58]. SWHE allows both types of operations but with a limited number of times. The bound on the number of operation is due to the fact that the noise grows much faster with the number of operations. There are several works on SWHE, some of the important ones are [59–63]. FHE allows all types of operations with an unlimited number of times by handling the noise using the *bootstrapping technique*. The first reasonable FHE scheme was introduced by Craig Gentry in 2009 [22, 23]. Although this was a breakthrough, several works and implementations hitherto demonstrated that FHE still needs significant improvement to be able to used in practice [64–71].

## 2.1.6 Oblivious Transfer Protocols

Oblivious Transfer (OT) protocol was primarily introduced by Rabin [72] and later Even *et al.* [73] presented a 1-out-of-2 OT protocol. A 1-out-of-2 OT protocol takes place between two participants: a *sender*  $S$  and a *receiver*  $R$ , where  $S$ 's input is  $(m_0, m_1)$  and  $R$ 's input is  $b \in \{0, 1\}$ . The OT must guarantee that after protocol executions  $S$  receives nothing about the selection bit, and  $R$  receives only  $(m_b)$  corresponds to his input and nothing about  $(m_{1-b})$ . Crépeau [74] later showed that Rabin's OT essentially implies 1-out-of-2 OT. In another words, he showed that using Rabin's OT one can realize a 1-out-of-2 OT in polynomial number of steps.

We note that 1-out-of-2 OT can also be generalized to  $k$ -out-of- $n$  OT protocol where  $S$  has a set of values  $\{x_1, \dots, x_m\}$ ,  $R$  has  $k$  selection indices. At the 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 which allows the evaluator to obtain garbled wire tokens corresponding to his/her private inputs.

**OT extension:** OT extension is a way of obtaining many OTs from a few numbers of OT runs and cheap symmetric cryptographic operations. Ishai *et al.* constructed the first OT extension method [75], which reduces a given large number of required OTs to a fixed size security parameter (say  $n$ ). This is crucial in MPC implementations especially when the evaluator's input size is too much.

A protocol for reducing  $m$  OTs to  $n$  OTs is as follows. Sender  $S$ 's inputs:  $(x_1^0, x_1^1), \dots, (x_m^0, x_m^1)$  and receiver  $R$ 's input:  $\sigma = \sigma_1, \dots, \sigma_m$ . The sender  $S$  samples a random string  $s \in \{0, 1\}^n$ ; denote  $s = s_1, \dots, s_n$ . The receiver  $R$  samples  $n$  random strings  $T_1, \dots, T_n \in \{0, 1\}^m$ . For  $i = 1, \dots, n$ , Now  $S$  and  $R$  run a new sub-OT protocol as  $R$  plays the sender and inputs the pair  $(T_i, T_i \oplus \sigma)$  and  $S$  plays the receiver and inputs  $s_i$ . Denote the output of  $S$  by  $Q_i$  ( $Q_i = T_i$  if  $s_i = 0$ , and

$Q_i = T_i \oplus \sigma$  if  $s_i = 1$ ).

Let  $Q$  be the  $m \times n$  matrix  $[Q_1 | \dots | Q_n]$ ;  $Q(i) = i^{\text{th}}$  row. Let  $T$  be the  $m \times n$  matrix  $[T_1 | \dots | T_n]$ ;  $T(i) = i^{\text{th}}$  row. For  $i = 1, \dots, m$ :

- S sends  $y_i^0 = x_i^0 \oplus H(i, Q(i))$  and  $y_i^1 = x_i^1 \oplus H(i, Q(i) \oplus s)$ .
- R outputs  $z_i = y_i^{\sigma_i} \oplus H(i, T(i))$ .

Later, several OT extension schemes based on [75] are proposed for improving the efficiency [76, 77].

## 2.2 Basics of Secure Computation and Garbled Circuits

This section provides background information about secure computation and the garbled circuit scheme for secure computation originally proposed by Yao and some primitives for formal security analysis.

Secure computation protocols allow two or more mutually (possibly distrustful) parties to collaborate and compute a public functionality using their private inputs. Secure computation got a lot of attention in recent years due to its advantages for cloud computing and secure outsourcing. Consider the following real-life problems.

- Alice wants to investigate her DNA because of her suspicious about an inherent genetic disease. She is aware of a database (e.g. a cloud service) which contains DNA sequences about numerous genetic diseases. Once Alice gets a sample of her DNA sequence, she can make a query to the database, who will then declare Alice the possible diagnosis. On the other hand, in case Alice is concerned about her personal privacy, the above naive procedure is not applicable because it does not prevent Alice's private information both the query (DNA information) and the result (diagnosis) [78]. The database query problem can also mandate that the server does not learn not only the user query but

also the answer to the query. Besides, the service may also need data privacy due to accountability concerns. For instance, in case the service is charging for answering each query, than he wants to make sure that no information other than the answer to a single query is leaked at each transaction.

- The number of orbits around the Earth is nearly 7,000 spacecraft, orbital debris larger than 10 centimeters are routinely tracked and their number exceeds 21,000. It is reasonable that competitor countries do not want to leak the position information of their vital strategic satellite orbits. Besides, space satellites are a huge investment and the owners would like to keep their satellites alive in the space as long as possible. Satellites are able to approximate their positions on the space. These data can be analyzed to predict collisions and hopefully react to the more critical results. Once the satellite pairs with a sufficiently high collision risk have been found, the satellite operators should exchange more detailed information and determine if a collision is imminent and decide if the trajectory of either object should be modified [79].

The common focus of the above-mentioned illustrations is the following: The parties would like to execute a specific function on their confidential inputs, and learn the output result, but neither party is permitting to reveal its own input. The problem is how to handle such cooperative computation problems without revealing the privacy of the party's inputs and eliminate the need of a trusted third party. Secure multi-party computation (MPC) is a strong candidate approach as a solution to these problems. In order to that parties can obtain the output of a desired function by engaging in a protocol where they exchange some messages. The ultimate aim is that nothing is revealed aside from the output of the protocol as the value of the function.

Other examples of such computations include real-life applications such as: voting over the Internet [80–83], electronic bidding [84, 85], financial data analysis [86], privacy preserving data mining [87, 88] data sharing & analytics [89–91], blockchain solutions [92–97], etc. For more reading on applications of MPC, we refer to [98–102].

In fact, there is no bound for the fields where MPC could be applied, and it can be adopted in any relevant cases.

A secure two-party computation protocol allows two parties to compute a common function using their private inputs without leaking any information except the output. The concept is appeared in the 1980-s by the seminal work of Andrew Yao, but the original have been far too inefficient for practical use. The very classical garbled circuit construction methods require four ciphertexts per gate, although a quite large effort has been put into reducing this cost. The two-party MPC is an important special case, which received a lot of targeted attention [98], and because two-party protocols are often different from the generic n-party case (in terms of protocol efficiency etc.), we use the abbreviation 2PC to emphasize this special case as needed.

### 2.2.1 Yao's Garbled Circuit

In 1980s Andrew Yao has shown that secure two-party protocols can be constructed for any computable function [103,104]. In Yao's protocol, the function is represented as a Boolean circuit and it is quite efficient in terms of number of rounds, which is constant. The original protocol is secure in the semi-honest adversary model.

In a nutshell, Yao's garbled circuit protocol allows two parties (garbler and evaluator) having inputs  $x_1$  and  $x_2$  to evaluate a function  $f(x_1, x_2)$  without revealing any information about their private inputs beyond the function output. The basic concept is that the garbler computes an *encrypted* form of the circuit  $C_f$ ; then the evaluator obviously obtains the output of  $C_f$  without retrieving any private intermediate values.

Beginning with the Boolean circuit  $C_f$  (in which both parties agreed upon in advance), the garbler associates two garbled tokens  $X_i^0$  and  $X_i^1$  for each wire  $i$  of the circuit ( $X_i^0$  corresponds to the semantic value 0 and  $X_i^1$  to 1). Then, for each two-fan-in and one-fan-out gate  $g$  of the circuit with input wires  $i, j$  and output wire

$y$ , the garbler computes the following four ciphertexts for all inputs  $b_i, b_j \in \{0, 1\}$ .

$$Enc_{X_i^{b_i}, X_j^{b_j}}^y (X_y^{g(b_i, b_j)}) \quad (2.1)$$

This results in four random ordered ciphertexts that yield a *garbled gate*. In the end, the collection of garbled gates constitutes the *garbled circuit* which is sent to the evaluator.

In order to perform the garbled circuit evaluation, the evaluator needs the garbled tokens (keys) corresponding to each party’s input wires. The garbler can simply send (in plaintext form) the keys that correspond to her own inputs. For the evaluator’s inputs, the parties should run an oblivious transfer (OT) protocol. In addition, the garbler sends a mapping that reveals the resulting output-wire tokens to the semantic output bits.

### 2.2.2 Optimizations on Yao’s Scheme

In the past, academicians had a various prediction regarding the applicability of Yao’s scheme. In 1997, Goldwasser [105] states that: “*The field of multi-party computations is today where public-key cryptography was ten years ago, namely an extremely powerful tool and rich theory whose real-life usage is at this time only beginning but will become in the future an integral part of our computing reality*”. However, Goldreich [106] points out that using the solutions derived by general results for the special case of multi-party computation could be impractical; special solutions should be developed and tailored for special cases for efficiency reasons.

The past few years have seen much progress in constructing secure and efficient secure multi-party schemes using garbled circuits. With the recent improvements, the garbled circuit approach is now believed to be a feasible solution for real-life secure computation problems.

Recently, several important optimizations have been proposed that improves either the garbled circuit construction, or the computation of both the garbler and the evaluator, or the bandwidth efficiency. Some of the major optimizations are point and permute [107], free-XOR [27], garbled row reduction [84, 108], pipelining [109], dual-key cipher [2], miniLEGO [110], fleXOR [28], and half gates technique [1]. All these optimizations mostly consider the semi-honest adversary model. With the recent improvements, Yao's protocol has now very impressive results in terms of complexity and communication bandwidth requirements. We now give a brief summary of some of the seminal ones.

### Point and permute

The simple version of Yao's method basically decrypts all ciphertexts which demand on four decryptions per gate to evaluate the circuit. In [107], an elegant method is introduced which reduces the circuit evaluator's work from four decryptions to one. In this method for each wire  $i$ , garbler chooses  $w_i^0$ ,  $w_i^1$  and a *signal bit*  $\sigma_i$ . The basic intuition is that if  $\sigma_i$  equals 0, then write the ciphertexts that use  $w_i^0$  first; otherwise, write them second. The order of the ciphertexts for general  $\sigma_i$  and  $\sigma_j$  is as follows:

$$\begin{aligned}
c_0 &= E_{w_i^{\sigma_i}} \left( E_{w_j^{\sigma_j}} \left( w_k^{g(\sigma_i, \sigma_j)} \parallel \sigma_k \oplus g(\sigma_i, \sigma_j) \right) \right) \\
c_1 &= E_{w_i^{\sigma_i}} \left( E_{w_j^{\bar{\sigma}_j}} \left( w_k^{g(\sigma_i, \bar{\sigma}_j)} \parallel \sigma_k \oplus g(\sigma_i, \bar{\sigma}_j) \right) \right) \\
c_2 &= E_{w_i^{\bar{\sigma}_i}} \left( E_{w_j^{\sigma_j}} \left( w_k^{g(\bar{\sigma}_i, \sigma_j)} \parallel \sigma_k \oplus g(\bar{\sigma}_i, \sigma_j) \right) \right) \\
c_3 &= E_{w_i^{\bar{\sigma}_i}} \left( E_{w_j^{\bar{\sigma}_j}} \left( w_k^{g(\bar{\sigma}_i, \bar{\sigma}_j)} \parallel \sigma_k \oplus g(\bar{\sigma}_i, \bar{\sigma}_j) \right) \right)
\end{aligned}$$

The evaluator uses these keys  $w_i^{\sigma_i} \parallel \phi_i$  and  $w_j^{\phi_j} \parallel \phi_j$  to decrypt the ciphertext at that position  $\phi_i, \phi_j$ . By doing this evaluator will recover  $w_k^{g(b_i, b_j)} \parallel \phi_k$  where  $\phi_k = \sigma_k \oplus g(b_i, b_j)$  as desired.



## Free-XOR

Kolesnikov and Schneider [27] present an influential approach that removes the need of garbling XOR gates (so XOR gates become free, incurring no communication or cryptographic operations). They proposed picking a *global random* value  $R$  and a single random token  $w_i^0$  for wire  $i$ , and setting the token for the complement one as  $w_i^1 = w_i^0 \oplus R$ . If  $k$  is the output wire of an XOR gate, then  $w_k^0 = w_i^0 \oplus w_j^0$  and  $w_k^1 = w_k^0 \oplus R$ . On the both garbler and evaluator side the XOR operation is simple. Consider an XOR gate with input wires  $i, j$  and output wire  $k$  and given input garbled wire values  $w_i$  and  $w_j$ . We want to compute  $w_k = w_i \oplus w_j$ . Let  $w_i = w_i^\alpha$  and  $w_j = w_j^\beta$ . If  $\alpha = \beta = 0$  then  $w_k = w_i^0 \oplus w_j^0 = w_k^0 = w_i^0 \oplus R \oplus w_j^0 \oplus R = w_k^0$ . Non-XOR gates (such as AND, OR etc.) are computed as usual (with  $w_k^1 = w_k^0 \oplus R$ ).

Free-XOR method remarkably reduces the complexity of the garbled circuits in terms of both computation and communication and become a seminal work that took a big step towards making MPC practical.

On the other hand, the security of Free-XOR method is questioned by Choi *et al.* [111]. Kolesnikov and Schneider proved (somehow) security of their approach in the random oracle model, and claimed that correlation robustness is sufficient for their scheme. However, Choi *et al.* [111] showed that the free-XOR technique is not secure based on correlation robustness alone and some form of circular security is also needed. This work also demonstrates that correlation robustness is strictly weaker than circular-correlation robustness that means weaker than also random oracle model.

## Garbled row reduction

From the end of the 90s, the focus mostly turns to reducing the bandwidth overhead since it seems to be one of the most important bottlenecks for secure computation protocols. Naor *et al.* [84] introduced two types of optimizations for row reductions in a garbled scheme to reduce bandwidth consumption. These optimizations are later formally described by Pinkas *et al.* [108]. Naor *et al.* [84] presented an optimization

for reducing the standard 4-ciphertext garbled gates to three ciphertexts. In this optimization, one of the ciphertexts is fixed to an all-zeros bit string. Since one of the rows is set to always consist of all-zeros, then it does not actually need to be included in the garbled table. Result of decryption of this all-zeros row gives one of the tokens.

Later Pinkas et al. in [108] proposed a technique to reduce the size of a garbled table from four to three ciphertexts, thus saving 25% of network bandwidth. This method is also known as the *GRR3* method as it requires to send three ciphertexts per gate in the communication channel. The method is as follows:

- Let  $i$  and  $j$  be the input wires and let  $k$  be the output wire
  - Set  $(w_k^{g(\sigma_i, \sigma_j)} || \sigma_k \oplus g(\sigma_i, \sigma_j)) =: H(w_i^{\sigma_i} || w_j^{\sigma_j})$
  - When using the free-XOR technique, set  $w_k^{1-b} = w_k^b \oplus R$  otherwise, choose it at random.
  - Construct ciphertexts  $c_2, c_3, c_4$  as usual
- When computing the gate:
  - If both signal bits equal 0 (i.e.,  $\alpha \oplus \sigma_j = \beta \oplus \sigma_j = 0$ ), then don't decrypt; just derive  $w_k^{g(\sigma_i, \sigma_j)}$  and  $\sigma_k \oplus g(\sigma_i, \sigma_j)$  by computing  $H(w_i^{\sigma_i} || w_j^{\sigma_j})$
  - Otherwise decrypt one of  $c_2, c_3, c_4$  as usual

In [108] another garbled row reduction variant called *GRR2* is proposed for reducing the bandwidth size to two ciphertexts per gate. *GRR2* involves computing polynomial interpolation and a modified version of secret sharing [112]. Therefore, *GRR2* is more costly than the standard PRF or hash function garbling. The performance experiments in [108] also show that *GRR2* is about three times slower than the fastest experiment.

In general, *GRR* is a technique for reducing a standard garbled gate bandwidth from a size of 4 ciphertexts down to either 3 or 2. However, Free-XOR is only

compatible with *GRR3*, but not with the more compressive *GRR2* variant. The underlying reason is that in the *GRR2* method, both output wire labels (for 0 and 1) of a gate as fixed pseudorandom functions of the input wire labels. Therefore, it is not always achievable to guarantee that the output wire labels are of the form  $(C, C \oplus R)$  for the global  $R$  value of Free-XOR method.

## **FleXOR**

In [28] Kolesnikov, Mohassel, and Rosulek introduced a generalization of free-XOR technique called the flexible XOR, shortly *fleXOR*. In *fleXOR* technique, an XOR gate can be garbled using 0, 1, or 2 ciphertexts, depending on the combinatorial structure of the Boolean circuit and the circuit is compliant with *GRR2* applied to the AND gates. For circuits with many AND gates, this method results in smaller circuits than with free-XOR. In other cases, free-XOR could be more preferable. Therefore, the actual benefit of this method is strictly depending on the structure of the Boolean circuit.

Considering the security of fleXOR, the circularity assumption can be removed using this technique at some additional cost while the correlation robustness/related key assumption remains.

## **Half gates technique**

In [1], Zahur, Rosulek, and Evans propose an elegant and efficient garbling scheme called *half gates* technique. Their garbling technique is currently known as the most efficient optimization in terms of communication complexity compared to any prior scheme. The idea behind the approach is to divide an AND gate into two half AND gates for which one party knows one of the inputs.

This technique remains compatible with free-XOR [27] while also reducing the ciphertext requirement for each odd gate<sup>1</sup> to two. Here, we briefly describe the

---

<sup>1</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.

Table 2.1: Garbling an odd gate using half gates technique [1].

<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>

garbling procedure of odd gates using the half gates technique and refer the reader to [1] for further details and its security proof.

Any odd gate type can be written in the form of Equation (2.2) 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 [1]. Let  $v_i$  denote the one-bit truth value on the  $i^{th}$  wire in a circuit.

$$f_{G_{\text{odd}}}(v_a, v_b) \rightarrow (\alpha_1 \oplus v_a) \wedge (\alpha_2 \oplus v_b) \oplus \alpha_3 \quad (2.2)$$

The garbler garbles an odd gate by following the steps for both half gates in Table 2.1. The tokens for FALSE and TRUE on the  $i^{th}$  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  $\text{lsb}(w_i^0) \neq \text{lsb}(w_i^1)$ .  $w_{Gc}^b$  and  $w_{Ec}^b$  denote the tokens for the garbler and the evaluator half gate outputs for truth value  $\mathbf{b}$ , respectively.  $T_{Gc}$  and  $T_{Ec}$  denote the  $\lambda$ -bit strings needing to be sent for the garbler and evaluator half gates, respectively.

Let  $w_i$  be a token on  $i^{th}$  wire obtained by the evaluator who does not know its corresponding truth value  $v_i$ . For the  $i$ 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>2</sup>, having the security parameter  $\lambda$ .

<sup>2</sup>Circular correlation robustness for naturally derived keys is the security requirement for a

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.

Half gate method presents the best known garbled circuit scheme in terms of both communication and computation complexity comparing any prior scheme.

### 2.2.3 Adversary Types

In what follows, we briefly describe the types and intuition for the capabilities of the semi-honest, covert, and malicious adversaries. For a more detailed discussion, we refer to [88, 113] and for formal definitions [114].

#### Malicious adversary

A *malicious adversary* is a kind of active attacker that can arbitrarily deviate from specification of the protocol and utilizes any effective strategy to retrieve some additional knowledge about the other parties private data and/or manipulate the outcome of the computation. Since this is the strongest type of adversary model, in most cases, protection against such attacks is excessive and expensive to achieve [115]. In literature, there exist many proposals to provide security malicious model such as [116–122].

#### Covert adversary

A *covert adversary* is also a kind of active attacker that can arbitrarily deviate from specification of the protocol. Covert adversary is very similar to malicious adversary but it differs in the sense that the attacker can only be caught with a given probability (e.g., 1/2) called as the deterrence factor [115, 123, 124].

---

suitable hash function used in half gates garbling. We refer the reader to [1] for its details.

## **Semi-honest adversary**

A *semi-honest adversary* is a type of passive attacker that executes verbatim the prescribed steps of the protocol but can record all the intermediate transactions and make analysis in order to retrieve some additional knowledge about the other parties private data. This class of attacker also known as *honest-but-curious adversary* [100, 114]. In fact, several MPC protocols and designs consider the semi-honest model since this model is highly relevant for numerous real-world applications [84, 109, 125–128].

# Chapter 3

## RELATED WORKS

In this chapter, we give an overview of the existing private function evaluation protocols, especially for two parties. First proposed by Andrew Yao [103, 104], 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 has been promising progress over the original Yao’s protocol [1, 27–29, 84, 108, 129, 130]. 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. The 2PFE concept is first appeared in [131, 132]. So far, there are basically two main approaches that PFE solutions are built upon.

Several proposals in literature have aimed to design efficient *special*-purpose and *general*-purpose PFE protocols. In what follows, we explore the fundamental approaches on designing PFE solutions which can be classified as general purpose (universal circuit based) and special purpose PFE protocols. The special purpose protocols can be divided into two sub-categories (i) (mostly)<sup>1</sup> symmetric-based, (ii) asymmetric-based PFE solutions. We first begin with reviewing the general purpose

---

<sup>1</sup>The only asymmetric cryptographic structure is due to the OT operations of underlying 2PC, therefore it can be considered as symmetric-based (see [19, 26]).

universal circuit based PFE solutions. Next, we explore existing the special purpose PFE solutions.

### 3.1 Universal Circuit Based PFE Solutions

The general-purpose PFE solutions reduce the *universal circuit* [133] based approach that works with any MPC protocol. The idea is that if the regular secure computation techniques can be applied on a universal circuit, then a PFE scheme can be obtained. 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  $\mathcal{C}_f$  of the private function  $f$ , and inputs of parties  $x_1, \dots, x_n$  (*i.e.*,  $\mathcal{F}_{U_g}(\mathcal{C}_f, x_1, \dots, x_n)$ ), 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 [14, 15, 18, 19, 25]. The early universal circuit based schemes result in massive circuit sizes [14, 15, 133, 134], which was the root cause of their inefficiency. By the recent works [18] and [19] the universal circuits approach becomes more practical but the computation cost is still worse than the special purpose OSN (symmetric) based protocols of [17, 26] and the communication cost is worse than the asymmetric-based protocols of [9, 17, 21].

### 3.2 Special Purpose PFE Solutions

The second approach falls into designing special purpose PFE protocols which avoids the use of universal circuits. Following this line of work, several PFE schemes have been proposed [9, 16, 17, 20, 21, 26, 135]. An early attempt on this category is Paus, Sadeghi, and Schneider's work [16]. 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. A remarkable work embracing this approach is *singly homomorphic encryption* based 2PFE scheme of Katz and Malka (KM11) applied on Boolean circuits [9]. This work utilizes a singly homomorphic scheme (e.g., ElGamal [50] or Paillier [55]) for the generation of the two random tokens<sup>2</sup> on each wire, later utilized in the 2PC stage. They first propose a basic version of their protocol in [9, Sect. 3.1] (we call KM11-1st) and for the efficiency concerns they propose a more efficient variant in [9, Sect. 3.2] (we call KM11-2nd). Both schemes have only three rounds and provide  $O(g)$  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 [17], Mohassel and Sadeghian come up with a framework for PFE that includes several schemes for different settings. They have 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 [17, 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<sup>3</sup>. For Boolean circuits, they propose two types of protocols: one is based on partially HE (we call MS13-HE) and the other one is based on *oblivious evaluation of switching networks* (we call MS13-OSN). The MS13-OSN protocol of [17] is (mostly) based on symmetric cryptographic primitives since the only asymmetric cryptographic structure is due to the OT operations of underlying 2PC. Their proposals are essentially secure in the semi-honest model and have later been extended to the malicious model by [135].

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

---

<sup>2</sup>Throughout this dissertation, 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.

<sup>3</sup>We can intuitively say that as the latency between parties increases, so does the cost of each additional communication round (we refer to [136] that backs up this discussion). A similar analysis on trade-offs between Boolean and arithmetic circuit based protocols have also been addressed in [137, p. 527].

complexities of [9]. On the other hand, MS13-HE provides linear communication and computation complexities and slightly outperforms KM11-2nd. We remark that to the best of our knowledge, a reusability feature cannot be adapted<sup>4</sup> to protocols proposed in [9] and [17].

The existing schemes based on asymmetric cryptographic primitives such as [9] and partially HE based protocol of [17] are promising in terms of linear communication complexity. However, for some applications, protocols primarily based on symmetric cryptography could be favorable.

Considering OSN based 2PFE scheme of [17], 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 **Party**<sub>1</sub>) detaches the interconnections of the gates to obtain single gates, and keeps the topological mapping of the circuit private. Second, **Party**<sub>1</sub> and the other party (say **Party**<sub>2</sub>) engage in an *oblivious evaluation of switching network* (OSN)<sup>5</sup> protocol which consists of  $O(g \log(g))$  *oblivious transfer* (OT) operations (throughout this dissertation,  $g$  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 [103, 138] where **Party**<sub>1</sub> and **Party**<sub>2</sub> 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 [17].

---

<sup>4</sup>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.

<sup>5</sup>The OSN mechanism is introduced in [17] to achieve a solution for the oblivious 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 [30], Wang and Malluhi attempt to improve the 2PFE 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%.

## Chapter 4

# AN EFFICIENT 2-PARTY PRIVATE FUNCTION EVALUATION PROTOCOL BASED ON HALF GATES

In this chapter, we mainly focus on improving *2-party private function evaluation* (2PFE) based on (mostly) symmetric cryptographic primitives. This chapter is based on our work published in *The Computer Journal* [20]. In Section 4.1, we first give an introduction on 2PFE framework and scheme of [17] in detail. We formally present our 2PFE scheme in Section 4.2. Section 4.3 provides a simulation-based security proof of our 2PFE scheme in the semi-honest model. In Section 4.4, we analyze our protocol in terms of communication and computation complexities and compare it with 2PFE scheme in [17].

### 4.1 2-Party PFE Framework

In [17], 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., 2PFE). In order to achieve a secure 2PFE, Mohassel and Sadeghian show that hiding (i) the parties' private inputs, (ii) the topology of the circuit representation  $\mathcal{C}_f$ , 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 thesis, the party who knows the private function is denoted by  $\text{Party}_1$ , plays the evaluator role in 2PC; whereas the other party is denoted by  $\text{Party}_2$  plays the garbler role in 2PC. In a nutshell, in CTH,  $\text{Party}_1$  extracts the topological mapping  $\pi_f$  (kept private) from the circuit representation  $\mathcal{C}_f$  and converts the whole circuit into a collection of single gates. Then  $\text{Party}_1$  and  $\text{Party}_2$  engage in an *oblivious evaluation of switching network* (OSN) protocol where  $\text{Party}_2$  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 2PFE scheme proposed in [17].

#### 4.1.1 Context of CTH

Let  $n$  and  $m$  denote the number of inputs and outputs of  $\mathcal{C}_f$ , 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  $\mathcal{C}_f$  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  $\mathcal{C}_f$ . 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 4.1 shows an example circuit  $\mathcal{C}_f$  and its mapping  $\pi_f$ .

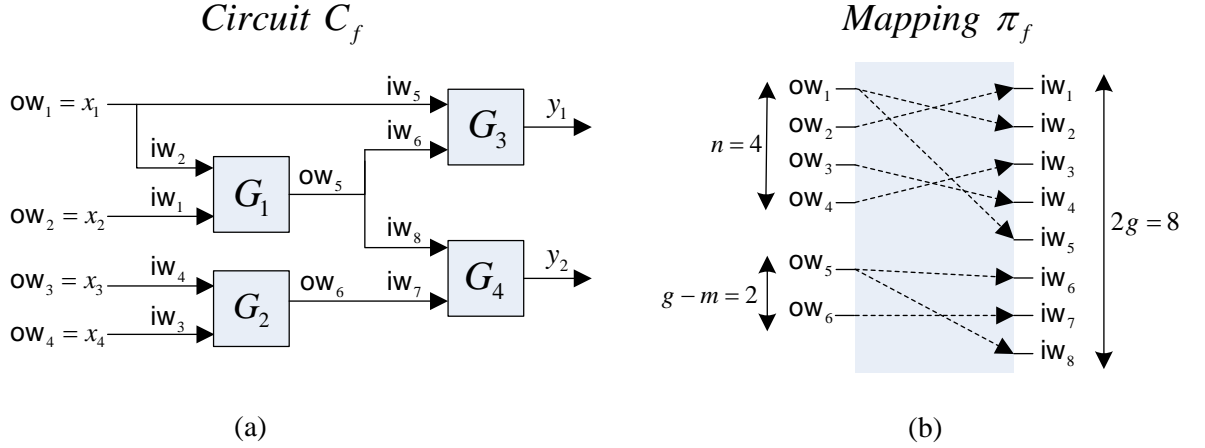


Figure 4.1: (a) A circuit representation  $\mathcal{C}_f$  of a function  $f$ . (b) The mapping  $\pi_f$  of  $f$ .

From the inclusion-exclusion principle, we obtain Equation (4.1) that gives the number of possible mappings for the given  $M$  and  $N$  values.

$$\rho = \sum_{i=0}^M (-1)^i \binom{M}{i} (M-i)^N \quad (4.1)$$

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 4.1.1** (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 extended permutation (2-OEP) functionality is defined as follows:

**Definition 4.1.2** (2-OEP functionality). *The first party  $\text{Party}_1$ '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  $\text{Party}_2$ 's inputs are a vector for outgoing wires  $W := [w_i \leftarrow \{0, 1\}^\lambda]$  for  $i = 1, \dots, M$ . In the end,  $\text{Party}_2$  learns  $S := [\sigma_j = w_{\pi_f^{-1}(j)} \oplus t_j]$  for  $j = 1, \dots, N$  while  $\text{Party}_1$  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 [17] 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 [17]. 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 4.1.3** (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  $i$ .*

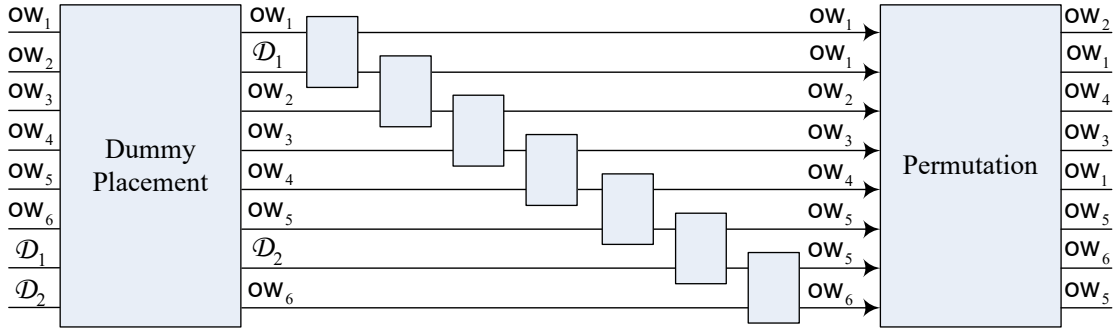


Figure 4.2: The related switching network for the mapping  $\pi_f$  in Figure 4.1.

A *permutation network* PN is a special type of SN whose mapping is a permutation of its inputs. In contrast to SNs, PNs composed of 1-switches. Waksman proposes an efficient PN construction in [139]. Mainly, this work suggests that a PN with  $N = 2^n$  inputs can be constructed with  $N \log(N) - N + 1$  switches. In [17], the authors propose the construction of an extended permutation by combining SNs and PNs. However, extended permutations differ from SNs 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 4.2).

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  $2(N \log(N) - N + 1) + N - 1 = 2N \log(N) - N + 1$  [17]. The topology of the whole switching network is the same for all  $N$  input EPs, and the selection bits specify the input values appearing on the outputs.

**Oblivious evaluation of switching network (OSN) construction** 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, **Party<sub>1</sub>** produces the selection bits of the switching network using  $\pi$ , and has a blinding vector  $\mathcal{T}$ . **Party<sub>2</sub>** has an input vector for outgoing wires  $W$ . In the end, **Party<sub>2</sub>** learns the switching network’s blinded output vector for incoming wires  $S$ , and **Party<sub>1</sub>** 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 **Party<sub>2</sub>** 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. **Party<sub>1</sub>** 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 **Party<sub>1</sub>** to obtain his output shares according to his selection bits. There are four possibilities for **Party<sub>1</sub>**’s output shares depending on his selection bits  $s_{0u}$  and  $s_{1u}$  (see Table 4.1).

Table 4.1: **Party<sub>1</sub>** 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)	$w_2 \oplus r_c$	$w_2 \oplus r_d$

**Party<sub>2</sub>** prepares a table with four rows using  $r_a, r_b, r_c, r_d$  (see Table 4.2). **Party<sub>1</sub>** and **Party<sub>2</sub>** engage in a 1-out-of-4 OT in which **Party<sub>2</sub>** inputs the four rows that she has prepared, and **Party<sub>1</sub>** inputs his selection bits for the switch  $u$ . At the end, **Party<sub>1</sub>** learns one of the rows as the output in the table. Assume that **Party<sub>1</sub>**'s selection bits are (1,0). This means **Party<sub>1</sub>** retrieves the third row, *i.e.*,  $(r_b \oplus r_c, r_a \oplus r_d)$ . According to his selection bits, **Party<sub>1</sub>** 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, **Party<sub>2</sub>** sets a uniformly random  $\lambda$ -bit string to each wire in the switching network. **Party<sub>2</sub>** 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). **Party<sub>2</sub>** prepares tables for each switch in the switching network similar to Table 4.1 and Table 4.2. However, both tables for each switch in this scenario have two rows since each switch, in fact, has two possible outputs<sup>1</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. **Party<sub>2</sub>** needs to send her blinded inputs to **Party<sub>1</sub>**, which can be done during her turn in OT extension in order not to increase the round complexity unnecessarily. Once **Party<sub>1</sub>** gets **Party<sub>2</sub>**'s blinded inputs which are also his input shares and the outputs of all OTs, he evaluates the entire switching network in

---

<sup>1</sup>For the 1-switches in dummy placement and permutation components, the first and second rows of Table 4.1 and Table 4.2, and for 2-switches in replacement components, the second and third rows of Table 4.1 and Table 4.2 are sufficient.

Table 4.2:  $\text{Party}_1$  gets one of these rows by engaging in 1-out-of-4 OT with  $\text{Party}_2$ .

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

topological order, obtaining his output shares.  $\text{Party}_1$  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  $\text{Party}_2$ .  $\text{Party}_2$  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.

#### 4.1.2 Mohassel and Sadeghian's 2PFE scheme

Here we provide an outline of Mohassel and Sadeghian's 2PFE construction and refer the reader to their work for detailed information and its security proof [17]. Their protocol is as follows.  $\text{Party}_2$  first randomly generates tokens  $w_i^0, w_i^1 \leftarrow \{0, 1\}^\lambda$  for each  $ow_i \in \text{OW}$  corresponding to FALSE and TRUE, respectively.  $\text{Party}_1$  also generates random blinding strings  $t_j^0, t_j^1 \leftarrow \{0, 1\}^\lambda$  for each  $iw_j \in \text{IW}$ . And then  $\text{Party}_1$  and  $\text{Party}_2$  engage in OSN slightly modified from their 2-OEP protocol, where at the end,  $\text{Party}_2$  learns  $[\sigma_j^0 = w_{\pi_f^{-1}(j)}^0 \oplus t_j^{b_j}]$  and  $[\sigma_j^1 = w_{\pi_f^{-1}(j)}^1 \oplus t_j^{\bar{b}_j}]$ .  $\text{Party}_2$  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.  $\text{Party}_2$  sends the garbled gates and her garbled input tokens to  $\text{Party}_1$ .  $\text{Party}_1$  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  $\text{Party}_1$  evaluates the whole garbled circuit, and obtains the tokens of output bits of  $f(x)$ . In [17], a gate hiding mechanism is not provided for 2PFE 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 [17]. All of the OTs in the protocol can be combined for just one invocation of OT extension. The overall computation cost<sup>2</sup> of [17] is about  $6N \log(N) + 2N + 12$  symmetric-key cryptographic operations.

## 4.2 Our Efficient 2-Party PFE Scheme

In what follows, we describe our scheme in detail (see also Figure 4.3). In the preparation stage, **Party<sub>1</sub>** compiles the function into a Boolean circuit  $\mathcal{C}_f$  consisting of only NAND gates<sup>3</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*  $\tilde{\mathcal{C}}_f$  defined as follows:

**Definition 4.2.1** (Template of Private Circuit ( $\tilde{\mathcal{C}}_f$ )). *A template of private circuit  $\tilde{\mathcal{C}}_f$  is some information about a circuit  $\mathcal{C}_f$  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. The steps of our complete 2-party PFE protocol is provided below:

**Party<sub>1</sub>’s Input:** A bit string  $x_1$  and a function  $f$ .

**Party<sub>2</sub>’s Input:** A bit string  $x_2$ .

---

<sup>2</sup>In [18], the computation cost of [17] is also computed. We note that there is a minor typo in [18, p. 723] i.e., the computation complexity of [17] 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>3</sup>Any functional-complete gate can be used to rule out the need for a gate hiding mechanism as in [17].

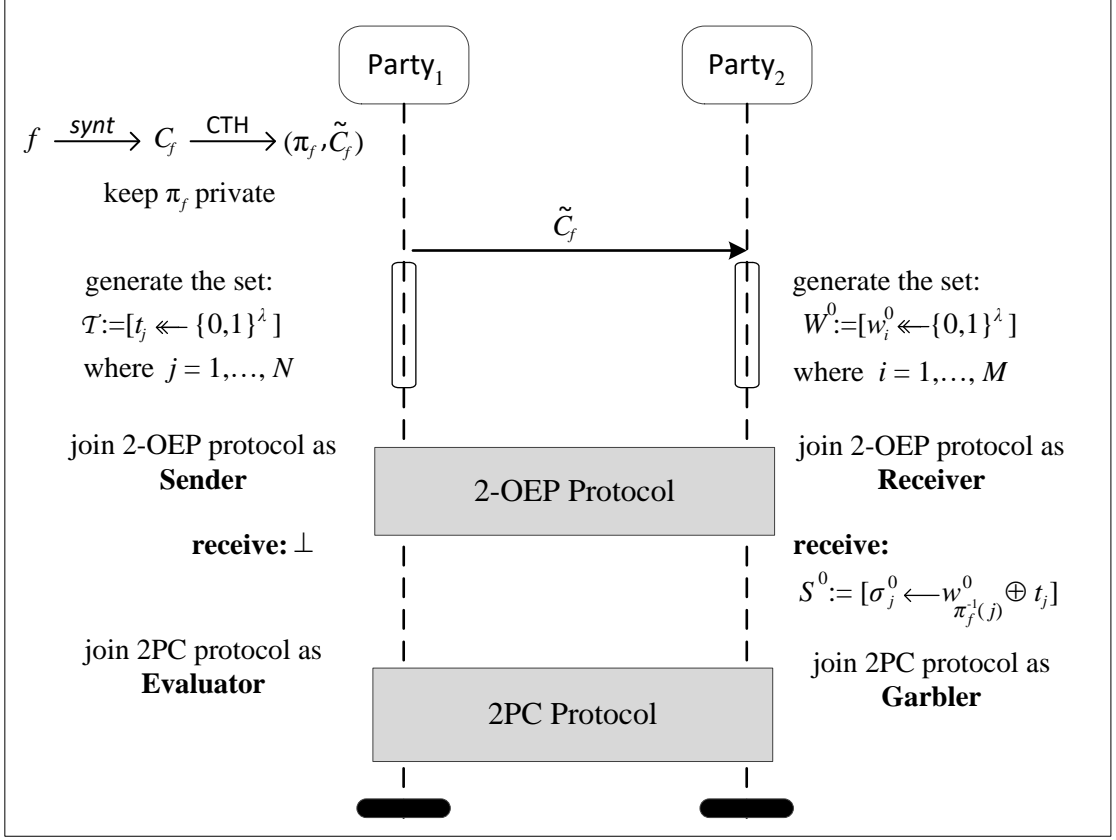


Figure 4.3: Components and high level procedures of our PFE protocol. The private function  $f$  is only known to Party<sub>1</sub>. Party<sub>1</sub> compiles  $f$  into a Boolean circuit  $C_f$ , and extracts the mapping  $\pi_f$  and the template of private circuit  $\tilde{C}_f$ . Party<sub>1</sub> sends  $\tilde{C}_f$  to Party<sub>2</sub>. Party<sub>1</sub> randomly generates the vector  $\mathcal{T}$ . Party<sub>2</sub> randomly generates the vector  $W^0$ . They engage in a 2-OEP protocol where Party<sub>2</sub> learns  $S^0$  as the output. With the knowledge of  $W^0$ ,  $S^0$  and  $\tilde{C}_f$ , Party<sub>2</sub> garbles each gate and sends the garbled circuit to Party<sub>1</sub>. With the knowledge of  $\pi_f$ ,  $\tilde{C}_f$ ,  $\mathcal{T}$ , the garbled circuit and the garbled inputs, Party<sub>1</sub> evaluates the whole garbled circuit.

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

**Preparation:**

1. Party<sub>1</sub> 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. Party<sub>1</sub> sends  $\tilde{C}_f$  to Party<sub>2</sub>.

3. **Party<sub>2</sub>** randomly generates an  $\lambda$ -bit token  $w_i^0 \leftarrow \{0, 1\}^\lambda$  for **FALSE** on each  $\text{ow}_i \in \text{OW}$ . This yields a total of  $M = n + g - o$  pairs. Moreover, **Party<sub>2</sub>** sets a vector  $W^0 := [w_i^0]$  for  $i = 1, \dots, M$ .
4. **Party<sub>1</sub>** generates an  $\lambda$ -bit blinding string  $t_j \leftarrow \{0, 1\}^\lambda$  for each  $\text{iw}_j \in \text{IW}$ . He sets those values to a blinding vector  $\mathcal{T} := [t_j]$  for  $j = 1, \dots, 2g$ .

### 2-OEP Protocol:

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

### 2PC Protocol (**Party<sub>2</sub>** plays the garbler, and **Party<sub>1</sub>** plays the evaluator):

6. **Garbling:** **Party<sub>2</sub>** generates a secret  $\lambda$ -bit offset  $R \leftarrow \{0, 1\}^{\lambda-1}$ . **Party<sub>2</sub>** sets the token for **TRUE** on each  $\text{ow}_i$  as  $w_i^1 \leftarrow w_i^0 \oplus R$ , and the blinded for **TRUE** on each  $\text{iw}_j$  as  $\sigma_j^1 \leftarrow \sigma_j^0 \oplus R$ . Moreover, **Party<sub>2</sub>** 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  $\tilde{\mathcal{C}}_f$ , **Party<sub>2</sub>** garbles each odd gate using the **Gb** procedure in Figure 4.4, resulting in three ciphertexts per non-output gate and two ciphertexts per output gate. **Party<sub>2</sub>** sends the garbled circuit  $\hat{F}$  and the tokens  $\hat{X}_2$  for her own inputs  $x_2$  to **Party<sub>1</sub>**. **Party<sub>1</sub>** gets tokens  $\hat{X}_1$  for his own input bits  $x_1$  from **Party<sub>2</sub>** 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, **Party<sub>2</sub>** needs to pick  $R$  and compute the tokens for **TRUE** on **Party<sub>1</sub>**'s input wires before 2-OEP protocol.)
7. **Evaluating:** With the knowledge of  $\pi_f$ ,  $\mathcal{T}$ ,  $\hat{F}$  and the garbled input  $\hat{X} = (\hat{X}_1, \hat{X}_2)$ , **Party<sub>1</sub>** 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  $\sigma_j$ . **Party<sub>1</sub>** evaluates each garbled gate using

Table 4.3: Adapting half gates technique to our 2PFE 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 (2.2)). The token  $w_c^0$  on the output wire equals  $w_{Gc}^0 \oplus w_{Ec}^0 \oplus \psi_c$ . The three ciphertexts  $T_{Gc}$ ,  $T_{Ec}$ , and  $\psi_c$  are sent to Party<sub>1</sub> for each gate.

<b>Garbler half gate</b> ( $p_b$ known to the garbler)	<b>Evaluator half gate</b> ( $p_b \oplus v_b$ known to the evaluator)
<u>Defines the half gate:</u>	<u>Defines the half gate:</u>
$f_G(v_a, p_b) := (\alpha_1 \oplus v_a)(\alpha_2 \oplus p_b) \oplus \alpha_3$	$f_E(v_a, v_b \oplus p_b) := (\alpha_1 \oplus v_a)(p_b \oplus v_b)$
<u>Computes:</u>	<u>Computes:</u>
$T_{Gc} \leftarrow H(\sigma_a^0) \oplus H(\sigma_a^1) \oplus (p_b \oplus \alpha_2)R$	$T_{Ec} \leftarrow H(\sigma_b^0) \oplus H(\sigma_b^1) \oplus \sigma_a^{\alpha_1}$
$w_{Gc}^0 \leftarrow H(\sigma_a^{p_a}) \oplus f_G(p_a, p_b)R$	$w_{Ec}^0 \leftarrow H(\sigma_b^{p_b})$
<u>Defines the third ciphertext:</u>	
$\psi_c := w_{Gc}^0 \oplus w_{Ec}^0 \oplus w_c^0$	
<u>Party<sub>2</sub> sends <math>T_{Gc}</math>, <math>T_{Ec}</math>, and <math>\psi_c</math>.</u>	

the Ev procedure in Figure 4.4. At the end, Party<sub>1</sub> obtains the tokens for  $f(x_1, x_2)$ .

### 4.2.1 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  $\text{ow}_i \in \text{OW}$ , respectively, and  $R$  be the global free-XOR offset [27] throughout the circuit. Party<sub>2</sub> sets  $w_i^0 \leftarrow \{0, 1\}^\lambda$  for each  $\text{ow}_i$ . The blinding string on the  $j$ th incoming wire  $\text{iw}_j \in \text{IW}$  is denoted as  $t_j$ . Party<sub>1</sub> sets  $t_j \leftarrow \{0, 1\}^\lambda$  for each  $\text{iw}_j$ . Party<sub>1</sub> and Party<sub>2</sub> engage in a 2-OEP protocol where Party<sub>1</sub>'s inputs are  $\pi_f$  and a blinding vector for incoming wires  $\mathcal{T} := [t_j]$  for  $j = 1, \dots, N$ , and Party<sub>2</sub>'s inputs is a token vector for FALSE on outgoing wires  $W^0 := [w_i^0]$  for  $i = 1, \dots, M$ . At the end, Party<sub>2</sub> learns the vector of blinded strings for FALSE  $S^0 := [\sigma_j^0 = w_{\pi_f^{-1}(j)} \oplus t_j]$  for  $j = 1, \dots, N$ , while Party<sub>1</sub> learns  $\perp$ .

Since our protocol allows all wires in the circuit to have the same offset  $R$ , unlike [17], Party<sub>1</sub> needs only a single blinding string  $t_j$  for each wire, and Party<sub>2</sub> 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 [17], 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).

### 4.2.2 Our 2PC Garbling Scheme for 2PFE

This section presents our garbling scheme based on half gates technique [1]. Similar to half gates technique, **Party**<sub>2</sub> 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  $\text{lsb}(\sigma_j^0) \neq \text{lsb}(\sigma_j^1)$ . **Party**<sub>2</sub> follows the steps in Table 4.3 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 the  $i$ th outgoing wire and the blinded string on the  $j$ th incoming wire obtained by **Party**<sub>1</sub> 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 **Party**<sub>2</sub>. 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  $\hat{e} = (e_1, e_2, \dots)$ .

In accordance with the framework<sup>4</sup> of [2], Figure 4.4 depicts our complete garbling scheme, composed of the following procedures. The garble procedure **Gb** takes

---

<sup>4</sup>Bellare, Hoang, and Rogaway introduce the notion of a garbling scheme as a cryptographic primitive. They also describe the procedures and security requirements of garbling schemes. We refer the reader to [2, 140] for details concerning definitions and introduction to the formal concepts of garbling schemes.



```

proc Gb( $1^\lambda, \tilde{C}_f, S^0, W^0$ ) :
   $R \leftarrow \{0, 1\}^{\lambda-1} 1$ 
  for  $i w_j \in \tilde{C}_f$  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 $_{\text{NAND}}^*$ ( $\sigma_a^0, \sigma_b^0, w^0$ ):
   $p_a \leftarrow \text{lsb}(\sigma_a^0); p_b \leftarrow \text{lsb}(\sigma_b^0)$ 
   $k \leftarrow \text{NextIndex}(); k' \leftarrow \text{NextIndex}()$ 
   $T_G \leftarrow H(\sigma_a^0, k) \oplus H(\sigma_a^1, k) \oplus p_b R$ 
   $w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G \oplus R$ 
   $T_E \leftarrow H(\sigma_b^0, k') \oplus H(\sigma_b^1, k') \oplus \sigma_a^0$ 
   $w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_b (T_E \oplus \sigma_a^0)$ 
   $\psi \leftarrow w_G^0 \oplus w_E^0 \oplus w^0$ 
  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  $\hat{y}$ 

```

Figure 4.4: Our complete half gate based garbling scheme for 2PFE.  $\text{Gb}_{\text{NAND}}$  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  $\hat{e} = (e_1, e_2, \dots)$ .

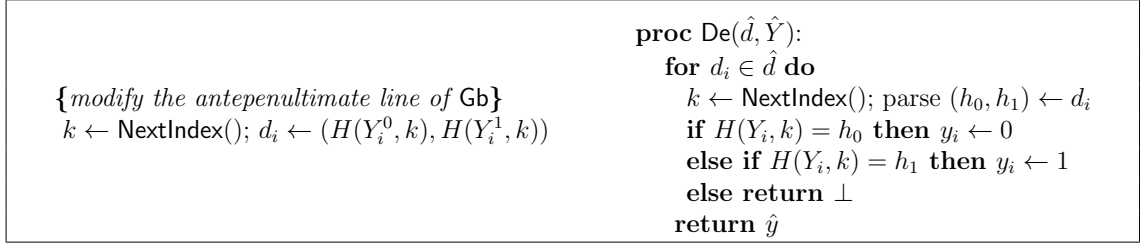


Figure 4.5: Modification of our garbling scheme in Figure 4.4 for achieving authenticity (auth) property.

$1^\lambda$ ,  $\tilde{\mathcal{C}}_f$ ,  $S^0$  and  $W^0$  as input, and outputs  $(\hat{F}, \hat{e}, \hat{d})$  where  $\hat{F}$  is the garbled version of  $\tilde{\mathcal{C}}_f$ ,  $\hat{e}$  is the encoding information, and  $\hat{d}$  is decoding information. **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  $(T_G, T_E, Y^0)$ . **En** is the encode algorithm that takes the plaintext input  $\hat{x}$  of the circuit and  $e$  as input, and outputs a garbled input  $\hat{X}$ . **Ev** is the evaluate procedure that takes the inputs  $\hat{F}$ ,  $\hat{X}$ ,  $\pi_f$  and  $\mathcal{T}$ , and outputs garbled output  $\hat{Y}$ . **De** is the decode algorithm that takes  $\hat{Y}$  and  $d$  as input, and outputs the plaintext output  $\hat{y}$  of the circuit. Finally, **ev** is an algorithm that is not needed for garbling but used for checking the *correctness* condition such that  $\text{De}(\hat{d}, \text{Ev}(\hat{F}, \text{En}(\hat{e}, \hat{x}))) = \text{ev}(f, \hat{x})$ . In Figure 4.6, we extend the garbling model of [2, 140] to be complied with our PFE scheme.

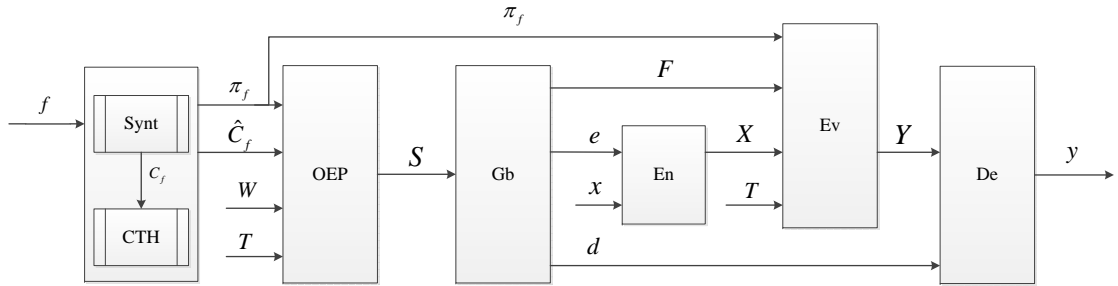


Figure 4.6: Components of and high level procedures of a OEP based Private Function Evaluation scheme. The topology hiding of the function  $f$  where  $\text{Party}_1$  is the evaluator and  $\text{Party}_2$  is the garbler: (1) The private function  $f$  is only known by  $\text{Party}_1$ . (2)  $\mathcal{C}_f$  is the Boolean circuit representation of  $f$ . (3)  $\pi_f$  is the circuit mapping of  $f$ . (4) The OEP protocol is mutually run where  $\text{Party}_2$  learns blinded strings. (5) The blinded strings learnt by  $\text{Party}_2$ . (6) Yao's protocol with the blinded strings.

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 2PFE, 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 **Party**<sub>1</sub> gets his own garbled inputs by means of OT. This can also be done in an earlier stage together with other OTs in the 2-OEP protocol (if OSN construction is used) in order not to increase round complexity. For this setting, **Party**<sub>2</sub> needs to pick  $R$  and compute the tokens for TRUE on **Party**<sub>1</sub>'s input wires before 2-OEP protocol. This setting is compatible with our protocol as well.

### 4.3 Security of the proposed protocol

In this section, we start by revisiting the code based games of [2] and security notions of Choi *et al.* [111] and Zahur *et al.* [1] as preliminaries. We then provide simulation-based security proof of our proposed protocol.

#### 4.3.1 Code based games and security notions

Bellare, Hoang, and Rogaway introduce the notion of a garbling scheme as a cryptographic primitive. We refer the reader to [2, 140] for details concerning definitions and a thorough introduction to the concepts of descriptive set theory.

Our work uses the **prv.sim**<sub>S</sub> (privacy), **obv.sim**<sub>S</sub> (obliviousness) and **auth**<sub>S</sub> (authenticity) security definitions of [2] depicted in Figure 4.7. Considering the **prv.sim** and **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, the adversary is allowed to make a single call to the **Garble** procedure.

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

Figure 4.7: Simulation based games for privacy, obliviousness and authenticity [2]. The function  $\mathcal{S}$  is a simulator, and  $G$  denotes a garbling scheme.

For further information about the simulation-based games and related security properties, we refer the reader to [2]. The advantages of the corresponding adversary classes are as follows:

$$\begin{aligned}
\text{Adv}_{G,\Phi,\mathcal{S}}^{\text{prv.sim}}(\mathcal{A}, \lambda) &:= \left| \Pr[\text{prv.sim}_{G,\Phi,\mathcal{S}}^{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \\
\text{Adv}_{G,\Phi,\mathcal{S}}^{\text{obv.sim}}(\mathcal{A}, \lambda) &:= \left| \Pr[\text{obv.sim}_{G,\Phi,\mathcal{S}}^{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \\
\text{Adv}_G^{\text{auth}}(\mathcal{A}, \lambda) &:= \Pr[\text{auth}_G^{\mathcal{A}}(1^\lambda) = 1]
\end{aligned} \tag{4.2}$$

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 [1].

- $\text{Circ}_R(x, j, b) = H(x \oplus R, j) \oplus bR$  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>5</sup> in order to prevent the adversary from trivially obtaining  $R$  [111]. Furthermore, we give the following definition for *natural queries*.

<sup>5</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  $b$  [111].

**Definition 4.3.1.** [1] 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.

### 4.3.2 Security Proof

Our security proof is based on the security proofs provided in [9] and [1].

**Theorem 4.3.1.** *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,*

```

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

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

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

```

Figure 4.8: Part-A. The simulator for  $\text{prv.sim}_{\mathcal{S}}$  security, and the hybrids 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  $\hat{e} = (e_1, e_2, \dots)$ .

<pre> <b>proc</b> <math>\mathcal{G}_3(1^\lambda, \tilde{\mathcal{C}}_f, \pi_f, \mathcal{T}, \hat{x})</math>:   <math>R \leftarrow \{0, 1\}^{\lambda-1}</math>   <b>for</b> <math>ow_i \in OW(\tilde{\mathcal{C}}_f)</math> <b>do</b>     <math>w_i^0 \leftarrow \{0, 1\}^\lambda, w_i^1 \leftarrow w_i^0 \oplus R</math>   <b>for</b> <math>iw_j \in IW(\tilde{\mathcal{C}}_f)</math> <b>do</b>     <math>\sigma_j^0 \leftarrow w_{\pi_f^{-1}(j)} \oplus t_j, \sigma_j^1 \leftarrow \sigma_j^0 \oplus R</math>   <b>for</b> <math>ow_i \in Inputs(\tilde{\mathcal{C}}_f)</math> <b>do</b>     <math>X_i \leftarrow w_i^{x_i}</math>   <b>for</b> each gate <math>\tilde{G}_i \in \tilde{\mathcal{C}}_f</math> <b>do</b>     <math>\{a, b\} \leftarrow GateInputs(\tilde{G}_i)</math>     <b>if</b> <math>\tilde{G}_i</math> is a non-output gate <b>then</b>       <math>(T_{G_i}, T_{E_i}, \psi_i) \leftarrow Sim_{NAND_3}^*(\sigma_a^0, \sigma_b^0, w_i^0)</math>       <math>F_i^{non-out} \leftarrow (T_{G_i}, T_{E_i}, \psi_i)</math>     <b>else</b>       <math>(T_{G_i}, T_{E_i}, Y_i^0) \leftarrow Sim_{NAND_3}(\sigma_a^0, \sigma_b^0)</math>       <math>F_i^{out} \leftarrow (T_{G_i}, T_{E_i})</math>       <math>Y_i^1 \leftarrow Y_i^0 \oplus R, d_i \leftarrow lsb(Y_i^0)</math>     <b>end if</b>   <b>return</b> <math>(\hat{F}, \hat{X}, \hat{d})</math> </pre>	<pre> <b>private proc</b> <math>Sim_{NAND_3}^*(\sigma_a^0, \sigma_b^0, w^0)</math>:   // <math>Sim_{NAND_3}(\sigma_a^0, \sigma_b^0)</math>:   <math>p_a \leftarrow lsb(\sigma_a^0); p_b \leftarrow lsb(\sigma_b^0)</math>   <math>k \leftarrow NextIndex(); k' \leftarrow NextIndex()</math>   <math>T_G \leftarrow H(\sigma_a^0, k) \oplus H(\sigma_a^1, k) \oplus p_b R</math>   <math>w_G^0 \leftarrow H(\sigma_a^0, k) \oplus p_a T_G \oplus R</math>   <math>T_E \leftarrow H(\sigma_b^0, k') \oplus H(\sigma_b^1, k') \oplus \sigma_a^0</math>   <math>w_E^0 \leftarrow H(\sigma_b^0, k') \oplus p_b (T_E \oplus \sigma_a^0)</math>   <math>\psi \leftarrow w_G^0 \oplus w_E^0 \oplus w^0</math>   // <math>Y^0 \leftarrow w_G^0 \oplus w_E^0</math>   <b>return</b> <math>(T_G, T_E, \psi)</math> // <math>(T_G, T_E, Y^0)</math> </pre>
--	---

Figure 4.9: Part-B. The simulator for  $\text{prv.sim}_S$  security, and the hybrids used in the proof. A ‘hat’ represents a sequence or a tuple, for instance,  $\hat{F} = (F_1, F_2, \dots)$  or  $\hat{e} = (e_1, e_2, \dots)$ . (Please see Figure 4.8 for the beginning of the figure.)

- the OT scheme for acquisition of  $\text{Party}_1$ ’s garbled input by  $\text{Party}_2$  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 the corruption of either party, separately. First, consider the case that  $\text{Party}_1$  is corrupted. Since the ideal 2-OEP functionality outputs  $\perp$  for  $\text{Party}_1$ , and the transcripts received by  $\text{Party}_1$  during OT reveals nothing other than  $\text{Party}_1$ ’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  $\text{Party}_2$ ’s input to  $\text{Party}_1$ . For any probabilistic polynomial time adversary  $\mathcal{A}_1$ , controlling  $\text{Party}_1$  in the real world, we construct a simulation game based on  $\text{prv.sim}$  game from [2] 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  $\text{prv.sim}$  game of Figure 4.7). Note that  $\mathcal{S}$  denotes the simulation function, and  $\text{Gb}$  denotes the

actual garbling (Figure 4.8 and 4.9 show the procedures 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  $\text{Party}_1$ . Finally, in the  $\text{Finalize}(\beta')$  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 4.8 and 4.8).

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  $\mathcal{O}$  changed from  $\text{Rand}$  to  $\text{Circ}_R$ . 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 4.8. Here, the variable  $R$  in  $\mathcal{G}_2$  refers to the  $R$  of the oracle  $\text{Circ}_R$ . 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  $W_i^{\bar{v}_i} \leftarrow W_i^{v_i} \oplus R$ . The algebraic relationships among variables remain unchanged. The oracle calls are also expanded in  $\text{SimAnd}_3$  to correspond to  $\mathcal{O} = \text{Circ}_R$ .

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  $\text{prv.sim}_{\mathcal{S}}$  game. Therefore, the advantage of  $\mathcal{A}_1$  in the  $\text{prv.sim}$  game

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



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

$k \leftarrow \text{NextIndex}(); r \leftarrow \{0, 1\}^\lambda$   
**if**  $y_i = 0$   
    **then**  $d_i \leftarrow (H(Y_i^0, k), r)$   
    **else**  $d_i \leftarrow (r, H(Y_i^0, k))$   
**end if**  
**return**  $(\hat{F}, \hat{X}, \hat{d})$

Figure 4.10: The required modifications on Figure 4.8 in order to show auth property.

is negligible. Hence, our scheme satisfies the security notion of  $\text{prv.sim}_{\mathcal{S}}$  and  $\text{obv.sim}_{\mathcal{S}}$ <sup>6</sup>. This proves that our scheme is secure against the corrupted  $\text{Party}_1$ .

Second, consider the case that  $\text{Party}_2$  is corrupted. For any probabilistic polynomial-time adversary  $\mathcal{A}_2$ , controlling  $\text{Party}_2$  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  $\text{Party}_2$ 's input, and  $\tilde{C}_f$  as follows.

1.  $\mathcal{S}'$  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.  $\mathcal{S}'$  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,  $\text{Party}_2$  receives only the message  $S^0$  in Round 2 (apart from the exchanged messages during the OT protocol for  $\text{Party}_1$ 's garbled input). However, the transcripts received by  $\text{Party}_2$  during OT do not leak any information to  $\text{Party}_2$  because of  $\mathcal{F}_{OT}$  in the OT-hybrid model. Due to one-time pad security, in  $\text{Party}_2$ '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. □

---

<sup>6</sup>The proof for  $\text{obv.sim}_{\mathcal{S}}$  differs from that of  $\text{prv.sim}_{\mathcal{S}}$  only in that in  $\text{obv.sim}_{\mathcal{S}}$ , the simulator neither computes  $\hat{d}$ , nor receives  $\hat{y}$ . So providing proof for  $\text{prv.sim}_{\mathcal{S}}$  also implies proof for  $\text{obc.sim}_{\mathcal{S}}$ .

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 [1], our garbling scheme in Figure 4.4 can be modified as in Figure 4.5 to achieve authenticity property (i.e., the antepenultimate line of **Gb** in Figure 4.4 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 4.4 can be modified as  $\text{De}(\hat{d}, \hat{Y}')$  procedure in Figure 4.5).

**Theorem 4.3.2.** *Our modified scheme (see Figure 4.4 and Figure 4.5) satisfies the security notion of **auth** with any  $H$  that has correlation robustness for naturally derived keys.*

*Proof Sketch.* We execute the simulator  $\mathcal{S}$  (in Figures 4.8 and 4.9 with the modifications in Figure 4.10), 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 4.5) 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.  $\square$

## 4.4 Performance Comparison

One of the primary objectives of the recent research on PFE is minimizing the communication cost. This is due to the fact that historical developments in hardware technology show us computing power advances faster than communication channels. This is even more likely to be so in the near future, *i.e.*, the main bottleneck for many secure computation applications will not be the CPU load but be the bandwidth constraints [28, 108]. Therefore we are first interested in the problem of minimizing the communication complexity.

Table 4.4: Analysis of communication costs for 2PFE schemes (see Section 4.1.1 for details of transfers in the OSN phases).

			MS'13 [17]		Our Protocol	
			Num. of Strings	Str. Length (bits)	Num. of Strings	Str. Length (bits)
OSN	Before OT Ext.	Party <sub>2</sub> → Party <sub>1</sub>	$N$	$2\lambda$	$N$	$\lambda$
	During OT Ext.	Party <sub>1</sub> → Party <sub>2</sub>	$\lambda$	$2N \log(N) + 1$	$\lambda$	$2N \log(N) - N + 1$
		Party <sub>2</sub> → Party <sub>1</sub>	$4N \log(N) - N + 2$	$2\lambda$	$4N \log(N) - 2N + 2$	$\lambda$
	After OT Ext.	Party <sub>1</sub> → Party <sub>2</sub>	$N$	$2\lambda$	$N$	$\lambda$
2PC	Garbled Circ.	Party <sub>2</sub> → Party <sub>1</sub>	$2N$	$\lambda$	$1.5N$	$\lambda$
TOTAL (bits)			$(10N \log(N) + 4N + 5)\lambda$		$(6N \log(N) + 0.5N + 3)\lambda$	

In this section, we evaluate the performance of our protocol and compare it with Mohassel and Sadeghian’s 2-party PFE scheme [17]. 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 [17], and that the OSN phases in both protocols are optimized with the General OT extension scheme of Asharov, Lindell, Schneider, and Zohner [77]. Similar results can be obtained by using other Ishai *et al.* based OT extension schemes [75, 76] as well.

Regarding the OSN phase, the total number of OTs in our 2PFE protocol is  $2N \log(N) - N + 1$ , while it is  $2N \log(N) + 1$  in [17] (see Section 4.1.1). Moreover, our protocol requires only one of the tokens on a wire entering the OSN phase, so the size of the rows in Table 4.2 which enter each OT is reduced by a factor of two [17], further resulting in a significant decrease in communication cost. Regarding the 2PC phase, 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 [17].

Table 4.4 shows the number of strings and their corresponding lengths sent in each turn in both schemes (see also Section 4.1.1 for details of transfers in the OSN phases). We omit the OTs for Party<sub>1</sub>’s garbled input, the transfers for decoding the garbled output, and the base OTs in the OT extension scheme [77]. The strings sent by Party<sub>2</sub> during the OT extension in [17], in fact, consists of  $4N \log(N) - 2N + 2$  of  $\lambda$ -bit stings and  $2N$   $\lambda$ -bit strings. The data sent by Party<sub>2</sub> before OT extension can also be sent during Party<sub>2</sub>’s turn in OT extension for saving in the number of

Table 4.5: Communication cost comparison of 2PFE schemes in terms of  $\lambda$ -bits.

Num. of Gates	MS'13 [17]			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%

rounds. Table 4.5 reflects the communication cost reduction achieved by our 2PFE protocol for the circuits with different number of gates.

Recently, in [30], Wang and Malluhi have attempted to improve the 2PFE scheme in [17] by removing only one ciphertext from each garbled gate (in 2PC phase) while retaining the cost of OSN phase unchanged. However, the influence of 2PC phase in [17] on overall communication cost is quite low (see Table 4.5). 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 [17]. More precisely, in the OSN phase, our scheme eliminates  $N$  oblivious transfer (OT) operations. This results in a decrease of  $2N$  symmetric encryptions performed by  $\text{Party}_2$  ( $\text{Party}_1$ '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  $\text{Party}_1$  ( $\text{Party}_2$ '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 2PFE scheme

in [17]. 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 [17].

## Chapter 5

# HIGHLY EFFICIENT AND REUSABLE PRIVATE FUNCTION EVALUATION WITH LINEAR COMPLEXITY

In this chapter, we propose a novel and highly efficient *two-party private function evaluation* (2PFE) scheme for Boolean circuits based on the DDH assumption. 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 subsequent executions for the same function and remarkably reduces the cost for the cases the same function is evaluated more than once.

In Section 5.1, we give a preliminary background that is used throughout this chapter. Section 5.2 presents the descriptions of our **InExe** and **ReExe** protocols, and a method for the case where **Party**<sub>1</sub> would like to execute 2PFE with various **Party**<sub>2</sub>s separately. Section 5.3 provides the complexities of our resulting protocols, and compare them with the existing state-of-the-art 2PFE protocols. In Section 5.4, we formally prove the security of our protocols in the semi-honest model.

### 5.1 Preliminaries

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

### 5.1.1 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 [141] providing a natural generalization of the DDH assumption for  $m > 2$  generators.

**Lemma 5.1.1** ([141]). *Under the DDH assumption on  $\mathbb{G}$ , for any positive integer  $m$ ,*

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

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

There exist certain elliptic curve groups where the DDH assumption holds. We refer the reader to [142, 143]. 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 [144] while the current best algorithms known for solving the elliptic curve DDH problem requires exponential time resulting in 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  $\mathbb{F}_p$  [145].

### 5.1.2 Notations and Concept of 2PFE Framework

In a two-party private function evaluation (2PFE) scheme,  $\text{Party}_1$  has a function input  $f$  (compiled into a boolean circuit  $C_f$ ) and optionally a private input bit string  $x_1$ , whereas  $\text{Party}_2$  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 [9, 17] conform to a generic 2PFE framework (formalized by [17]) 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.

In a nutshell, the 2PFE framework is as follows. Before starting the 2PFE protocol,  $\text{Party}_1$  compiles the function into a boolean circuit  $C_f$  consisting of only one type of gates (e.g., NAND gates). During the protocol execution,  $\text{Party}_1$  and  $\text{Party}_2$  first engage in a mapping evaluation protocol where  $\text{Party}_2$  obviously obtains the tokens on gate inputs, and then they mutually run a 2PC protocol where  $\text{Party}_2$  garbles each gate separately using those tokens, and  $\text{Party}_1$  evaluates the garbled circuit. As a result,  $\text{Party}_1$  obtains the garbled tokens that map to the corresponding outputs of the function (i.e.,  $y = f(x_1, x_2)$ ).

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  $\text{OW} = (\text{ow}_1, \dots, \text{ow}_{n+g-m})$  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  $\text{OW}$  is  $M = n + g - m$ . Similarly, let  $\text{IW} = (\text{iw}_1, \dots, \text{iw}_{2g})$  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  $\text{IW}$  is  $N = 2g$ . Throughout this paper,  $M$  and  $N$  denote the numbers of outgoing and incoming wires, respectively. Let  $\pi_f$  be a mapping such that  $j \leftarrow \pi_f(i)$  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  $C_f$ .

We define the public information of the circuit  $C_f$  as  $\text{PubInfo}_{C_f}$  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 [9, 17, 20] 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  $\text{Party}_1$  ( $q$ ). Both parties generate  $\text{PubInfo}_{C_f}$  at the beginning of the protocol execution (without an additional round of communication). Namely, each party runs the following deterministic procedure to obtain  $\text{PubInfo}_{C_f}$  on public input  $(g, n, m, q)$ :

- Set  $N := 2g$ ,  $M := n + g - m$ .
- For  $i = 1, \dots, g$ , set  $\text{iw}_{2i-1}$  and  $\text{iw}_{2i}$  as the incoming wires of the gate  $G_i$ .
- For  $i = 1, \dots, g - m$ , set  $\text{ow}_i$  as the outgoing wire of the gate  $G_i$ .
- For  $i = 1, \dots, q$ , set  $\text{ow}_{g-m+i}$  as the outgoing wire corresponding to  $\text{Party}_1$ 's  $i$ -th input bit.
- For  $i = 1, \dots, n - q$ , set  $\text{ow}_{g-m+q+i}$  as the outgoing wire corresponding to  $\text{Party}_2$ 's  $i$ -th input bit.
- For  $i = 1, \dots, m$ , set the output wire  $y_i$  as the output of  $G_{g-m+i}$ .
- Return  $\text{PubInfo}_{C_f} := (M, N, \text{OW}, \text{IW}, y)$ .

Next,  $\text{Party}_1$  generates  $\pi_f$  (i.e., the connection between incoming and outgoing wire indices) using the following randomized procedure on input  $(C_f, \text{OW}, \text{IW})$ .

- Randomly permute the indices  $1, \dots, g - m$ , and assign it to an ordered set  $A$ .
- For  $i = 1, \dots, g - m$ , assign  $G_{A[i]}$  to the  $i$ -th non-output gate in topological order.
- For  $i = 1, \dots, m$ , assign  $G_{g-m+i}$  to  $i$ -th output gate.

- Extract  $\pi_f$  from  $C_f$  according to the connections between **ows** and **iws**.
- Return  $\pi_f$ .

We next define as *Reusable Mapping Template* in which the efficiency of our scheme mostly due to the reusability of this template.<sup>1</sup>

**Definition 5.1.1** (Reusable Mapping Template). *Let  $\pi_f^{-1}(j)$  be the inverse mapping of  $\pi_f$  that denotes the index of the outgoing wire connected to  $\text{iw}_j$ . A Reusable Mapping Template is a set  $\text{ReuseTemp}_f := (\mathcal{P}, \mathcal{Q})$  such that  $\mathcal{P} := (P_1, \dots, P_M)$  where  $P_i$  is a generator of the group picked for  $\text{ow}_i$  by  $\text{Party}_2$  and  $\mathcal{Q} := (Q_1, \dots, Q_N)$  where  $Q_j := t_j \cdot P_{\pi_f^{-1}(j)}$  is a group element generated for  $\text{iw}_j$  by  $\text{Party}_1$  for  $t_j \in_R \mathbb{Z}_q^*$ ,  $i = 1, \dots, M$ , and  $j = 1, \dots, N$ .*

## 5.2 Our PFE Scheme

In this section, we first present our protocol for initial executions **InExe** which is optimized by offline/online decomposition (Figure 5.1). We next introduce our efficient resumption protocol for subsequent executions **ReExe** (Figure 5.3). We then propose an efficient method for executions with multiple  $\text{Party}_2$ s.

### 5.2.1 The description of our **InExe** protocol

We now introduce our efficient **InExe** scheme that is optimized by carrying out some of the computations in the off-line 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 give the full protocol steps of our optimized initial execution **InExe** protocol with a precomputation phase in Figure 5.2. Also,

---

<sup>1</sup>Although KM11 [9] also involves homomorphic encryption for token generation, it requires all protocol steps to be repeated in each subsequent executions.

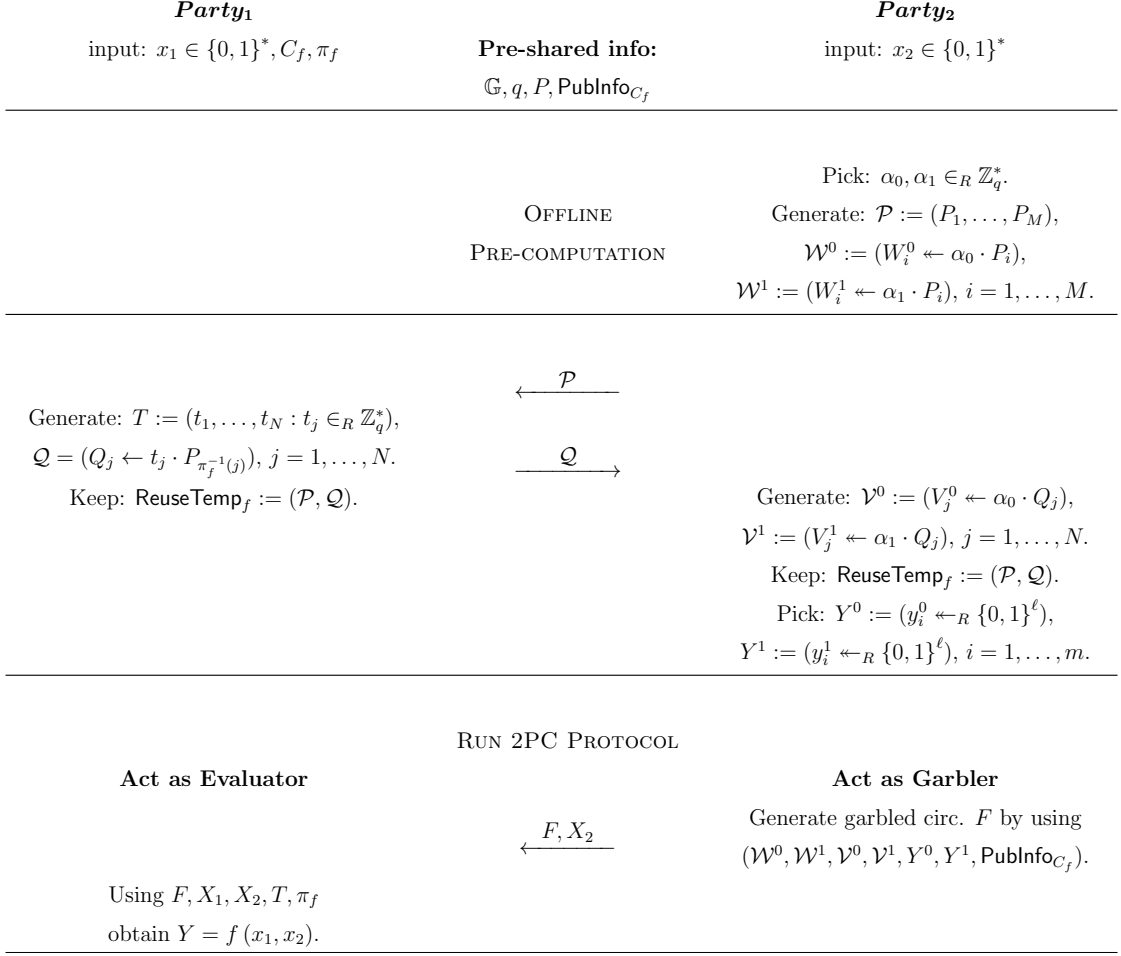


Figure 5.1: Sketch of our InExe 2PFE Protocol.  $\text{ReuseTemp}_f$  and  $T$  are stored (if needed) for the later PFE runs by ReExe protocol. Note that in case Party<sub>1</sub> has inputs  $(x_1)$  then OT protocol is required (to send the corresponding garbled  $X_1$ ) which can be trivially combined with the protocol rounds for minimization of the total number of rounds.

Figure 5.1 depicts the protocol steps of our InExe protocol. 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<sub>2</sub>.

In accordance with the generic 2PFE framework, the description of our InExe is as follows.

**Inputs:** 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  $P$  and the

### Our InExe Protocol: The procedures for the initial execution

**Party<sub>1</sub>'s Input:**  $x_1 \in \{0, 1\}^*$ , a Boolean circuit  $C_f$  consisting of NAND gates (compiled from the function  $f$ ) and a mapping  $\pi_f$  (extracted from  $C_f$ ).

**Party<sub>2</sub>'s Input:**  $x_2 \in \{0, 1\}^*$ .

**Pre-shared Information:** A group  $\mathbb{G}$  of prime order  $q$  with a generator  $P$  and  $\text{PubInfo}_{C_f}$ .

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

#### Precomputation phase

1. Party<sub>2</sub> 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  $\mathcal{W}^0 := (W_1^0, \dots, W_M^0 : W_i^0 \leftarrow \alpha_0 \cdot P_i, i = 1, \dots, M)$  for FALSEs and  $\mathcal{W}^1 := (W_1^1, \dots, W_M^1 : W_i^1 \leftarrow \alpha_1 \cdot P_i, i = 1, \dots, M)$  for TRUEs, where  $P_i$  is the  $i$ -th element in  $\mathcal{P}$  and each  $W_i^b$  is a token for  $\text{ow}_i \in \text{OW}$ ,  $b \in \{0, 1\}$ . Party<sub>2</sub> stores  $\mathcal{P}$ ,  $\mathcal{W}^0$ ,  $\mathcal{W}^1$ ,  $\alpha_0$ , and  $\alpha_1$ .

#### Online phase

##### Round 1:

2. Party<sub>2</sub> sends  $\mathcal{P}$  to Party<sub>1</sub>.

##### Round 2:

3. Party<sub>1</sub> generates the blinding set  $T := (t_1, \dots, t_N : t_j \in_R \mathbb{Z}_q^*, j = 1, \dots, N)$ , computes the set  $\mathcal{Q} = (Q_1, \dots, Q_N : Q_j \leftarrow t_j \cdot P_{\pi_f^{-1}(j)}, j = 1, \dots, N)$ . For the later PFE runs with the same function (if needed), Party<sub>1</sub> stores  $\text{ReuseTemp}_f := (P_1, \dots, P_M, Q_1, \dots, Q_N)$  (see Figure 5.4 for the protocol of subsequent executions (ReExe)). Party<sub>1</sub> sends  $\mathcal{Q}$  to Party<sub>2</sub>.

##### Round 3:

4. Party<sub>2</sub> prepares the group element sets  $\mathcal{V}^0 := (V_1^0, \dots, V_N^0 : V_j^0 \leftarrow \alpha_0 \cdot Q_j, j = 1, \dots, N)$  for FALSEs and  $\mathcal{V}^1 := (V_1^1, \dots, V_N^1 : V_j^1 \leftarrow \alpha_1 \cdot Q_j, j = 1, \dots, N)$  for TRUEs for  $\text{iw}_j \in \text{IW}$ . For the later PFE runs with the same function (if needed), Party<sub>2</sub> stores  $\text{ReuseTemp}_f = (P_1, \dots, P_M, Q_1, \dots, Q_N)$ . Next, Party<sub>2</sub> picks two random token sets for the output wires  $Y^0 := (y_1^0, \dots, y_m^0 : y_i^0 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m)$  and  $Y^1 := (y_1^1, \dots, y_m^1 : y_i^1 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m)$ .

5. The 2PC protocol now starts from this stage where **Party**<sub>2</sub> becomes the garbler and **Party**<sub>1</sub> becomes the evaluator. Using  $\mathcal{W}^0, \mathcal{W}^1, \mathcal{V}^0, \mathcal{V}^1, Y^0, Y^1$ , and  $\text{PubInfo}_{C_f}$ , **Party**<sub>2</sub> prepares the garbled circuit  $F$  by garbling each gate as follows. **Party**<sub>2</sub> prepares the following four ciphertexts to garble a non-output NAND gate  $G_a$  whose incoming wires  $\text{iw}_i$  and  $\text{iw}_j$ , and outgoing wire is  $\text{ow}_z$ :  $\text{Enc}_{V_i^0, V_j^0}(\overline{W_z^1}), \text{Enc}_{V_i^0, V_j^1}(\overline{W_z^1}), \text{Enc}_{V_i^1, V_j^0}(\overline{W_z^1}), \text{Enc}_{V_i^1, V_j^1}(\overline{W_z^0})$ . Similarly, **Party**<sub>2</sub> prepares the following four ciphertexts to garble an output NAND gate  $G_b$  whose incoming wires  $\text{iw}_i$  and  $\text{iw}_j$ , and output wire index is  $z$ :  $\text{Enc}_{V_i^0, V_j^0}(y_z^1), \text{Enc}_{V_i^0, V_j^1}(y_z^1), \text{Enc}_{V_i^1, V_j^0}(y_z^1), \text{Enc}_{V_i^1, V_j^1}(y_z^0)$ . Each garbled gate  $GG_a$  is then composed of four  $\ell$ -bit ciphertexts and two  $\log_2(\tau)$ -bit indices,  $I_a^1$  and  $I_a^2$  (see Section 5.2.1 for garbling details). **Party**<sub>2</sub> sends  $F$  and the garbled input  $X_2$  for its own input  $x_2$  to **Party**<sub>1</sub>. **Party**<sub>1</sub> also obtains the garbled input  $X_1$  for its own input  $x_1$  from **Party**<sub>2</sub> using parallel 1-out-of-2 OTs (or a more efficient OT extension scheme).<sup>a</sup>

6. Using  $F$ , the garbled input  $X = (X_1, X_2)$ ,  $T$ , and  $\pi_f$ , **Party**<sub>1</sub> evaluates the whole garbled circuit in topological order. If an outgoing wire  $\text{ow}_d$  is mapped to an incoming wire  $\text{iw}_e$ , 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  $(\text{iw}_i, \text{iw}_j)$  are computed. To evaluate each  $GG_a$ , **Party**<sub>1</sub> first computes  $H(V_i, V_j, \text{gateID})$ , and then XORs the ciphertext in the  $GG_a$  pointed by  $I_a^1$ -th and  $I_a^2$ -th bits of  $[H(V_i, V_j, \text{gateID})]_\tau$ . In the end, **Party**<sub>1</sub> obtains the token set  $Y = (y_1, \dots, y_m)$  for the output bits  $f(x_1, x_2)$ .

<sup>a</sup>Note that the OT protocol rounds can be combined with the former protocol rounds for minimization of the overall rounds.

Figure 5.2: Our Optimized InExe 2PFE Protocol via decomposition of offline/online computations

$\text{PubInfo}_{C_f}$  on inputs  $(g, n, m, q)$ . Each party has the following inputs: (1)  $\text{Party}_1$  holds a boolean circuit  $C_f$  consisting of only one type of gates (e.g., NAND gates) and the corresponding mapping  $\pi_f$ , and (possibly but not necessarily) his input  $x_1$  (2)  $\text{Party}_2$  holds his inputs  $x_2$ .

**Offline pre-computation phase:**  $\text{Party}_2$  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  $\mathcal{W}^0 := (W_1^0, \dots, W_M^0 : W_i^0 \leftarrow \alpha_0 \cdot P_i, i = 1, \dots, M)$  for FALSEs and  $\mathcal{W}^1 := (W_1^1, \dots, W_M^1 : W_i^1 \leftarrow \alpha_1 \cdot P_i, i = 1, \dots, M)$  for TRUEs, where  $P_i$  is the  $i$ -th element in  $\mathcal{P}$  and each  $W_i^b$  is a token for  $\text{ow}_i \in \text{OW}$ ,  $b \in \{0, 1\}$ .  $\text{Party}_2$  stores  $\mathcal{P}$ ,  $\mathcal{W}^0$ ,  $\mathcal{W}^1$ ,  $\alpha_0$ , and  $\alpha_1$ .

**Online phase:** Online phase consists of three rounds as follows.

**Round 1:**  $\text{Party}_2$  sends  $\mathcal{P}$  to  $\text{Party}_1$ .

**Round 2:**  $\text{Party}_1$  generates the blinding set  $T := (t_1, \dots, t_N : t_j \in_R \mathbb{Z}_q^*, j = 1, \dots, N)$ , computes the set  $\mathcal{Q} = (Q_1, \dots, Q_N : Q_j \leftarrow t_j \cdot P_{\pi_f^{-1}(j)}, j = 1, \dots, N)$ , where  $\pi_f^{-1}(j)$  denotes the index of the outgoing wire connected to  $\text{iw}_j$ .  $\text{Party}_1$  sends  $\mathcal{Q}$  to  $\text{Party}_2$ . Now, both parties have the knowledge of the set  $\text{ReuseTemp}_f := (\mathcal{P}, \mathcal{Q})$ .

For the later PFE runs with the same function (if needed),  $\text{Party}_1$  stores  $\text{ReuseTemp}_f$  (see Figure 5.4 for the protocol of subsequent executions (ReExe)).

$\text{Party}_2$  prepares the group element sets corresponding to  $\text{iw}_j \in \text{IW}$ . The set  $\mathcal{V}^0$  is for FALSE,  $\mathcal{V}^1$  is for TRUE semantic values.

$$\mathcal{V}^0 := (V_1^0, \dots, V_N^0 : V_j^0 \leftarrow \alpha_0 \cdot Q_j, j = 1, \dots, N),$$

$$\mathcal{V}^1 := (V_1^1, \dots, V_N^1 : V_j^1 \leftarrow \alpha_1 \cdot Q_j, j = 1, \dots, N).$$

Next,  $\text{Party}_2$  picks the following two randomly chosen ordered sets for output wires of the circuit

$$Y^0 := (y_1^0, \dots, y_m^0 : y_i^0 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m),$$

$$Y^1 := (y_1^1, \dots, y_m^1 : y_i^1 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m),$$

where  $\ell$  is the bit length of a group element (i.e.,  $\ell = \lceil \log_2(q) \rceil$ ). For the later PFE runs with the same function (if needed), **Party**<sub>2</sub> stores **ReuseTemp**<sub>*f*</sub>.

**Round 3:** Now, both parties then engage in a 2PC protocol where **Party**<sub>2</sub> and **Party**<sub>1</sub> play the garbler and evaluator roles, respectively. **Party**<sub>2</sub> garbles the whole circuit by using  $\mathcal{W}^0, \mathcal{W}^1, \mathcal{V}^0, \mathcal{V}^1, Y^0, Y^1$ , and **PubInfo**<sub>*C<sub>f</sub>*</sub>. Note that in contrast to the usual garbling in [9,17], in our garbling phase, **Party**<sub>2</sub> 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 [2] using a pseudorandom function as

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

where  $P_1$  and  $P_2$  are two group elements used as keys,  $\mathbf{m}$  is the  $\ell$ -bit plaintext, **gateID** is the index number of the gate,  $H : \mathbb{G} \times \mathbb{G} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\ell+\tau}$  is a hash-function (which we model as a random oracle),  $\tau$  is an integer such that  $\tau > 2 \log_2(4g)$  for preventing collisions in the  $\tau$  rightmost bits of hashes, and  $[H(X)]_\ell$  denotes the truncated hash value (of the message  $X$ ) which is cropped to the  $\ell$  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  $\tau$  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. [29]. Note that the encryption scheme **Enc** is based on the encryption scheme in [146] and differs from it only by the utilization of group elements as keys.

Let  $G_a$  be a non-output NAND gate for some  $a \in \{1, \dots, g\}$ . Let also  $\text{iw}_i, \text{iw}_j$  be the incoming wires and  $\text{ow}_z$  be the outgoing wire of  $G_a$  where  $i, j \in \{1, \dots, M\}$  and  $z \in \{1, \dots, N\}$ . To garble  $G_a$ , **Party**<sub>2</sub> prepares the following four ciphertexts

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

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

where  $\overline{W_z^1}$  and  $\overline{W_z^0}$  are the  $\ell$ -bit string representations of the group elements. Similarly, let  $G_b$  be an output NAND gate for some  $b \in \{1, \dots, g\}$ . Let also  $\text{iw}_i, \text{iw}_j$  be the incoming wires and  $z$  be the output wire index of  $G_b$  where  $i, j \in \{1, \dots, M\}$  and  $z \in \{1, \dots, m\}$ . To garble  $G_b$ ,  $\text{Party}_2$  prepares the following four ciphertexts

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

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

For the point and permute optimization [29], for each gate  $G_a$  in the circuit,  $\text{Party}_2$  picks random indices  $I_a^1, I_a^2 \in \{1, \dots, \tau\}$  such that

$$\begin{aligned} & \{(\mathsf{X}[I_a^1], \mathsf{X}[I_a^2]), (\mathsf{Y}[I_a^1], \mathsf{Y}[I_a^2]), (\mathsf{Z}[I_a^1], \mathsf{Z}[I_a^2]), (\mathsf{T}[I_a^1], \mathsf{T}[I_a^2])\} = \\ & \{(0, 0), (0, 1), (1, 0), (1, 1)\} \end{aligned}$$

where  $\mathsf{X} = [H(V_i^0, V_j^0, \text{gateID})]_\tau$ ,  $\mathsf{Y} = [H(V_i^0, V_j^1, \text{gateID})]_\tau$ ,  $\mathsf{Z} = [H(V_i^1, V_j^0, \text{gateID})]_\tau$ ,  $\mathsf{T} = [H(V_i^1, V_j^1, \text{gateID})]_\tau$ , and  $\mathsf{S}[I_a^i]$  denotes the  $I_a^i$ -th bit of the bit string  $\mathsf{S}$ . We denote each garbled gate  $GG_a$ , which is then composed of four  $\ell$ -bit ciphertexts,  $\text{ct}_a^1$ ,  $\text{ct}_a^2$ ,  $\text{ct}_a^3$ , and  $\text{ct}_a^4$ , 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  $\mathsf{X}$ ,  $\mathsf{Y}$ ,  $\mathsf{Z}$ , and  $\mathsf{T}$  values. For example, let  $\mathsf{X} = 011001 \dots 1$ ,  $\mathsf{Y} = 101111 \dots 0$ ,  $\mathsf{Z} = 110001 \dots 0$ , and  $\mathsf{T} = 010111 \dots 1$ . 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  $GG_a = (\text{ct}_a^1, \text{ct}_a^4, \text{ct}_a^3, \text{ct}_a^2, (I_a^1, I_a^2))$ . A trivial method for finding such a pair  $(I_a^1, I_a^2)$  could be as follows. First,  $\text{Party}_2$  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 a 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.  $\text{Party}_2$  garbles all the gates of the circuit in the above-mentioned way, and obtains the garbled circuit  $F$ .  $\text{Party}_2$  then sends  $F$  and its garbled input  $X_2$



(i.e., the  $W_i$  group elements for outgoing wires corresponding to  $x_2$ ) to  $\text{Party}_1$ . As usual,  $\text{Party}_1$  gets its own garbled input  $X_1$  (i.e., the  $W_i$  group elements for outgoing wires corresponding to  $x_1$ ) from  $\text{Party}_2$  using oblivious transfers (OT) (or one invocation of the OT extension schemes [75–77]). Note that this does not increase the round complexity of our overall protocol, since the exchange messages needed for OT rounds can be accompanied to the protocol rounds (i.e., the first round of OT is sent with  $\mathcal{P}$  message and the second one with  $\mathcal{Q}$  and the third one with  $F, X_2$ ).

Using  $F$ , the garbled input  $X = (X_1, X_2)$ ,  $T$ , and  $\pi_f$ ,  $\text{Party}_1$  evaluates the whole garbled circuit in topological order. If an outgoing wire  $\text{ow}_d$  is mapped to an incoming wire  $\text{iw}_e$ , 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 ( $\text{iw}_i, \text{iw}_j$ ) are computed. To evaluate each  $GG_a$ ,  $\text{Party}_1$  first computes  $H(V_i, V_j, \text{gateID})$ , and then XORs the ciphertext in the  $GG_a$  pointed by  $I_a^1$ -th and  $I_a^2$ -th bits of the  $H(V_i, V_j, \text{gateID})_\tau$ . In the end,  $\text{Party}_1$  obtains the token set  $Y = (y_1, \dots, y_m)$  for the output bits of the function  $y = f(x_1, x_2)$ .

### 5.2.2 Optimization with reusability feature: Our (ReExe) 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 a long-term business relationship instead of a single deal. Note that such a cost reduction is not available in the protocols of KM11 [9] and MS13 [17] since they require all token generation and 2PC procedures repeated in all executions. However, our scheme involves  $\text{ReuseTemp}_f$  that is reusable for the generation of tokens on incoming and outgoing wires. The reusability of  $\text{ReuseTemp}_f$

incurs a massive reduction in protocol overhead since a large part of costs in existing 2PFE protocols [9, 17] results from the generation of these tokens.

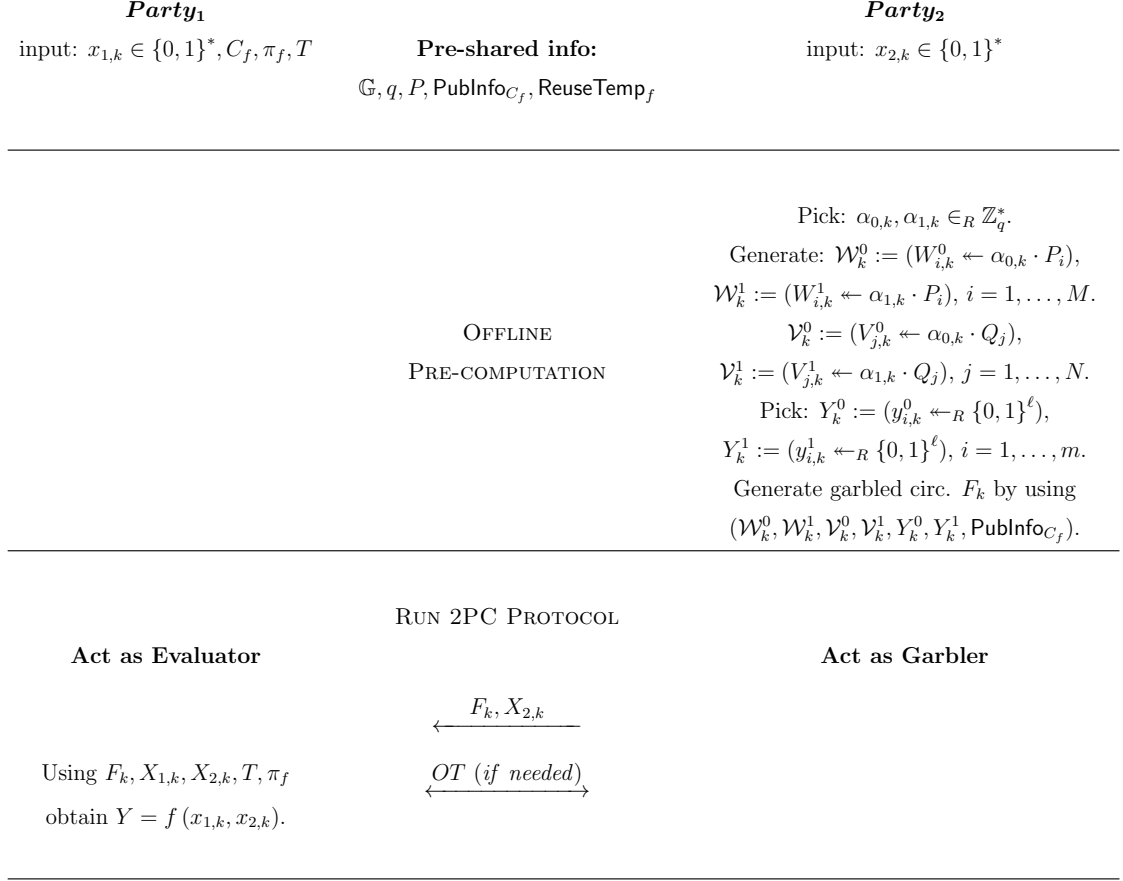


Figure 5.3: Sketch of our ReExe protocol for the  $k$ -th execution ( $k > 1$ ). The number of rounds is equal to 1, or 2, or 3 depending on the input size of Party<sub>1</sub>.

Our ReExe protocol optimizes the baseline InExe scheme presented by utilizing the Reusable Mapping Template  $\text{ReuseTemp}_f$  when the same private function is evaluated more than once.

Figure 5.3 depicts the sketch of our optimized ReExe protocol and Figure 5.4 give the detailed protocol steps. In ReExe protocol most of the calculations are performed in the offline pre-computation phase. For the  $k$ -th evaluation, Party<sub>2</sub> picks  $\alpha_{0,k}, \alpha_{1,k} \in_R \mathbb{Z}_q^*$  values then prepares the sets  $\mathcal{W}_k^0, \mathcal{W}_k^1, \mathcal{V}_k^0, \mathcal{V}_k^1, Y_k^0$  and  $Y_k^1$ . Then using  $\mathcal{W}_k^0, \mathcal{W}_k^1, \mathcal{V}_k^0, \mathcal{V}_k^1, Y_k^0, Y_k^1$  and  $\text{PubInfo}_{C_f}$ , Party<sub>2</sub> prepares the garbled circuit

$F$  as in the `lnExe` protocol. The online phase then includes only the 2PC stage that also runs the same way as in Section 5.2.1. During the evaluation procedure of the 2PC stage,  $\text{Party}_1$  always use the same  $T$  in all protocol runs.

The number of rounds is equal to 1, or 2, or 3 depending on the input string length of  $\text{Party}_1$ . Namely, if  $x_1 = \perp$ , then the number of rounds is equal to 1 (i.e., no rounds needed for OT). If  $\text{Party}_1$ 's input bits are not many, it is more efficient to use separate OTs for  $\text{Party}_1$ 's input tokens in parallel instead of an OT extension scheme. There exists OT schemes with 2 rounds (e.g., [147] and [148]). Hence, this choice results in a PFE scheme with overall 2 rounds (i.e., one round is sent accompanied by  $F_k, X_{2,k}$  message and the other OT round is sent from  $\text{Party}_1$  to  $\text{Party}_2$ ). If  $\text{Party}_1$ '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 [148]) and an additional round. Similarly, this choice results in a PFE scheme with overall 3 rounds.

### 5.2.3 Executing with Various $\text{Party}_2$ s

In the previous section, we have 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  $\text{Party}_1$  would like to run the 2PFE protocol for the same private function with various  $\text{Party}_2$ s separately. This is a relevant scenario where  $\text{Party}_1$  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  $\text{Party}_2$ s is suitable for.

First of all, we recall that the execution of our second protocol in Figure 5.3 requires the preknowledge of  $\text{ReuseTemp}_f := (\mathcal{P}, \mathcal{Q})$  by  $\text{Party}_2$  and the set  $T$  by

### Our ReExe Protocol: The procedures for the $k$ -th execution ( $k > 1$ )

**Party<sub>1</sub>'s Input:**  $x_{1,k} \in \{0, 1\}^*$ , a Boolean circuit  $C_f$  consisting of NAND gates (compiled from the function  $f$ ), a mapping  $\pi_f$  (extracted from  $C_f$ ), and the blinding set  $T := (t_1, \dots, t_N : t_j \in_R \mathbb{Z}_q^*, j = 1, \dots, N)$ .

**Party<sub>2</sub>'s Input:**  $x_{2,k} \in \{0, 1\}^*$ .

**Pre-shared Information:** A cyclic group  $\mathbb{G}$  of prime order  $q$  with a generator  $P$ ,  $\text{PubInfo}_{C_f}$ , and  $\text{ReuseTemp}_f^a$ .

**Output:**  $f(x_{1,k}, x_{2,k})$ .

---

#### Precomputation phase of the $k$ -th execution:

1. Party<sub>2</sub> picks  $\alpha_{0,k}, \alpha_{1,k} \in_R \mathbb{Z}_q^*$  and prepares the group element sets  $\mathcal{W}_k^0 := (W_{1,k}^0, \dots, W_{M,k}^0 : W_{i,k}^0 \leftarrow \alpha_{0,k} \cdot P_i, i = 1, \dots, M)$  for FALSEs and  $\mathcal{W}_k^1 := (W_{1,k}^1, \dots, W_{M,k}^1 : W_{i,k}^1 \leftarrow \alpha_{1,k} \cdot P_i, i = 1, \dots, M)$  for TRUEs for  $\text{ow}_i \in \text{OW}$ , and  $\mathcal{V}_k^0 := (V_{1,k}^0, \dots, V_{N,k}^0 : V_{j,k}^0 \leftarrow \alpha_{0,k} \cdot Q_j, j = 1, \dots, N)$  for FALSEs and  $\mathcal{V}_k^1 := (V_{1,k}^1, \dots, V_{N,k}^1 : V_{j,k}^1 \leftarrow \alpha_{1,k} \cdot Q_j, j = 1, \dots, N)$  for TRUEs for  $\text{iw}_j \in \text{IW}$ . Next, Party<sub>2</sub> generates two random token sets for output wires of the circuit  $Y_k^0 := (y_{1,k}^0, \dots, y_{o,k}^0 : y_{i,k}^0 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m)$  and  $Y_k^1 := (y_{1,k}^1, \dots, y_{o,k}^1 : y_{i,k}^1 \leftarrow_R \{0, 1\}^\ell, i = 1, \dots, m)$ .

2. The 2PC protocol now starts from this stage where Party<sub>2</sub> becomes the garbler and Party<sub>1</sub> 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  $\text{PubInfo}_{C_f}$ , Party<sub>2</sub> prepares the garbled circuit  $F_k$  by garbling each gate as follows. Party<sub>2</sub> prepares the following four ciphertexts to garble a non-output NAND gate  $G_a$  whose incoming wires are  $\text{iw}_i$  and  $\text{iw}_j$ , and outgoing wire is  $\text{ow}_z$ :  $\text{Enc}_{V_{i,k}^0, V_{j,k}^0}(\overline{W_{z,k}^1})$ ,  $\text{Enc}_{V_{i,k}^0, V_{j,k}^1}(\overline{W_{z,k}^1})$ ,  $\text{Enc}_{V_{i,k}^1, V_{j,k}^0}(\overline{W_{z,k}^1})$ ,  $\text{Enc}_{V_{i,k}^1, V_{j,k}^1}(\overline{W_{z,k}^1})$ . Similarly, Party<sub>2</sub> also prepares the following four ciphertexts to garble an output NAND gate  $G_b$  whose incoming wires are  $\text{iw}_i$  and  $\text{iw}_j$ , and output wire index is  $z$ :  $\text{Enc}_{V_{i,k}^0, V_{j,k}^0}(y_{z,k}^1)$ ,  $\text{Enc}_{V_{i,k}^0, V_{j,k}^1}(y_{z,k}^1)$ ,  $\text{Enc}_{V_{i,k}^1, V_{j,k}^0}(y_{z,k}^1)$ ,  $\text{Enc}_{V_{i,k}^1, V_{j,k}^1}(y_{z,k}^1)$ . Each garbled gate  $GG_{a,k}$  is then composed of four  $\ell$ -bit ciphertexts and two  $\log_2(\tau)$ -bit bit indices,  $I_{a,k}^1$  and  $I_{a,k}^2$  (see Section 5.2.1 for garbling details). Party<sub>2</sub> 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<sub>2</sub> sends  $F_k$  and the garbled input  $X_{2,k}$  for its own input  $x_{2,k}$  to Party<sub>1</sub>.

4. Party<sub>1</sub> gets the garbled input  $X_{1,k}$  for its own input  $x_{1,k}$  from Party<sub>2</sub> 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  $\pi_f$ , Party<sub>1</sub> evaluates the whole garbled circuit in topological order. If an outgoing wire  $\text{ow}_d$  is mapped to an incoming wire  $\text{iw}_e$ , 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 ( $\text{iw}_i, \text{iw}_j$ ) are computed. To evaluate each  $GG_{a,k}$ , Party<sub>1</sub> first computes  $H(V_{i,k}, V_{j,k}, \text{gateID})$ , 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  $[H(V_{i,k}, V_{j,k}, \text{gateID})]_\tau$ . At the end, Party<sub>1</sub> obtains the token set  $Y_k = (y_{1,k}, \dots, y_{o,k})$  for the output bits  $f(x_{1,k}, x_{2,k})$ .

<sup>a</sup> $\text{ReuseTemp}_f$  is already computed in  $\text{InExe}$  as in Figure 5.2.

Figure 5.4: Our Optimized ReExe 2PFE Protocol that utilizes Reusable Mapping Template

Party<sub>1</sub>. Trivially, once `ReuseTempf` and  $T$  are produced during `InExe` with any Party<sub>2</sub> as in our first protocol in Figure 5.1, then they can be stored, and our second protocol can be made use of in the subsequent executions with the same Party<sub>2</sub>. We are here interested in a more efficient mechanism running with various Party<sub>2</sub>s 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<sub>1</sub> 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<sub>1</sub>. In order to do so, we utilize a distributed system<sup>2</sup> 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<sub>1</sub>. Next, Party<sub>1</sub> computes the sets  $T$  and `ReuseTempf`. It then publishes `PubInfoCf` and `ReuseTempf` so that any prospective  $k$ -th party Party<sub>2, $k$</sub>  can utilize them in a 2PFE protocol run. This offline stage is dealt with only once, and its outputs (i.e.,  $T$  and `ReuseTempf`) are used in the subsequent executions. Note that the flow of re-executions for all Party<sub>2, $k$</sub> s is exactly the same as our `ReExe` protocol. We would like to stress that the costs of any execution in our new mechanism with a distributed system do not differ from the `ReExe` protocol.

### 5.3 Complexity Analysis

In this section, we first present the costs of our `InExe` and `ReExe` protocols in terms of communication, online computation, and round complexities. We then compare these protocols with the existing Boolean circuit based 2PFE schemes.  $M$ ,  $N$ ,  $\lambda$ , and  $\rho$  denote the number of outgoing wires (i.e., equal to  $n + g - m$ ), the number of incoming wires (i.e.,  $N = 2g$ ), the security parameter, and the computation cost

---

<sup>2</sup>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.

ratio, respectively. For KM11 and MS13-HE, it is assumed that the elliptic curve ElGamal is used for the singly homomorphic encryption schemes (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  $\lambda$ -bit security. We ignore the small communication cost of bit indices in garbled gates ( $2 \times \log_2(\tau)$  bits for each garbled gate) used for the point and permute optimization.

Table 5.1: Comparison of the existing 2PFE schemes 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.  $M$ ,  $N$ ,  $\lambda$ , and  $\rho$  denote the number of outgoing wires (i.e., equal to  $n + g - m$ ), the number of incoming wires (i.e.,  $N = 2g$ ), the security parameter, and the computation cost ratio, respectively.

	Communication	Online Comp.	Offline Comp.	Rounds
KM11-1st [9, Sec.3.1]	$(4M + 10N)\lambda$	$(\rho + 2.5)N$	$4(M + N)\rho$	3
KM11-2nd [9, Sec.3.2]	$(2M + 7N)\lambda$	$(\rho + 2.5)N$	$2(M + N)\rho$	3
MS13-OSN [17]	$(10N \log_2 N + 4N + 5)\lambda$	$6N \log_2 N + 2.5N + 3$	$O(\lambda)$	6
MS13-HE [17]	$(2M + 6N)\lambda$	$(\rho + 2.5)N$	$2(M + N)\rho$	3
GKS17 [19]	$(2N \log_2 N)\lambda$	$0.7N \log_2 N$	$2N \log_2 N$	3
BBKL18 [20]	$(6N \log_2 N + 0.5N + 3)\lambda$	$6N \log_2 N + N + 3$	$O(\lambda)$	6
<b>Our InExe Protocol</b>	$(2M + 6N)\lambda$	$(4\rho + 2.5)N$	$(3M - 1)\rho$	3
<b>Our ReExe Protocol</b>	$4N\lambda$	$(\rho + 0.5)N$	$2(M + N)\rho + 2$	1 / 2 / 3

### 5.3.1 Complexity of Our Scheme

**Communication cost:** Considering our InExe protocol, the overall communication overhead is  $(2M + 6N)\lambda$  bits, composed of (i) the set  $\mathcal{P}$  ( $M$  of  $2\lambda$ -bit strings) is sent by  $\text{Party}_2$  in Round 1, (ii) the set  $\mathcal{Q}$  ( $N$  of  $2\lambda$ -bit strings) is sent by  $\text{Party}_1$  in Round 2, (iii) the garbled circuit ( $2N$  of  $2\lambda$ -bit strings) is sent by  $\text{Party}_2$  in Round 3, where  $M$  is the number of outgoing wires and  $N$  is the number of incoming wires ( $N = 2g$ ). Considering our ReExe protocol, the use of  $\text{ReuseTemp}_f$  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.

**Computation cost:** In terms of online computation complexity, InExe protocol requires  $4N$  elliptic curve point multiplications, composed of (i)  $N$  operations by  $\text{Party}_1$  in Round 2, (ii)  $2N$  operations by  $\text{Party}_2$  in Round 3, (iii)  $N$  operations by  $\text{Party}_1$  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  $\text{Party}_2$  for garbling and  $0.5N$  operations by  $\text{Party}_1$  for evaluating). ReExe protocol reduces the online computation costs to  $N$  elliptic curve point multiplications and  $0.5N$  symmetric-key operations (carried out only by  $\text{Party}_1$ ). Note that Beaver’s OT pre-computation technique [147] can be used for decomposing OT’s for  $\text{Party}_1$ ’s input bits into online/offline stages. This eliminates online public-key operations of OT by carrying out them offline.

**Number of rounds:** Our InExe protocol has 3 rounds. The number of rounds of our ReExe protocol is equal to 1, or 2, or 3 depending on the input string length of  $\text{Party}_1$ . Namely, if  $\text{Party}_1$  has  $x_1 = \perp$ , then the number of rounds is equal to 1 (i.e., no rounds needed for OT). If  $\text{Party}_1$ ’s input bits are not many, it is more efficient to use separate OTs for  $\text{Party}_1$ ’s input tokens in parallel instead of an OT extension scheme. There exist OT schemes with 2 rounds (e.g., [147] and [148]). Hence, this choice results in a PFE scheme with overall 2 rounds. If  $\text{Party}_1$ ’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 [148]) and an additional round. Similarly, this choice results in a PFE scheme with overall 3 rounds.

### 5.3.2 Comparison

We now compare our 2PFE protocols with the state-of-the-art constant-round 2PFE protocols. In our scheme, we utilize an EC cyclic group where the DDH assumption holds for state-of-the-art efficiency. For [9], we take into account both protocols: (1) their “ $\mathcal{C}$ -PFE protocol” (see [9, Sect. 3.1], what we call KM11-1st) and (2) their

Table 5.2: Comparison of the existing 2PFE schemes in terms of overall communication costs for various circuit sizes. Here we take  $N = 2M$  and  $\lambda = 128$ .

	Number of Gates				
	$2^{10}$	$2^{15}$	$2^{20}$	$2^{25}$	$2^{30}$
KM11-1st [9, Sec.3.1]	0.38 MB	12.00 MB	0.38 GB	12.00 GB	384.00 GB
KM11-2nd [9, Sec.3.2]	0.25 MB	8.00 MB	0.25 GB	8.00 GB	256.00 GB
MS13-OSN [17]	3.56 MB	164.00 MB	6.69 GB	264.00 GB	10,048.00 GB
MS13-HE [17]	0.22 MB	7.00 MB	0.22 GB	7.00 GB	224.00 GB
GKS17 [19]	0.68 MB	32.00 MB	1.31 GB	52.00 GB	1,984.00 GB
BBKL18 [20]	1.89 MB	90.50 MB	3.77 GB	151.00 GB	5,776.00 GB
<b>Our InExe Protocol</b>	0.22 MB	7.00 MB	0.22 GB	7.00 GB	224.00 GB
<b>Our ReExe Protocol</b>	0.13 MB	4.00 MB	0.13 GB	4.00 GB	128.00 GB

“A More Efficient Variant” (see [9, Sect. 3.2], what we call KM11-2nd). For a fair comparison, we assume that the point and permute optimization [29] is directly applied to the MS13 and KM11 protocols during the 2PC phase<sup>3</sup>. Regarding the HE based schemes, for a fair comparison, we assume that EC-ElGamal is used. Also, considering KM11 and our protocols, we assume that each element of  $\mathbb{G}$  has a length  $\ell = 2\lambda$  bits for a  $\lambda$ -bit security.

Table 5.1 compares the existing 2PFE schemes in terms of overall communication cost, online/offline computation costs, and the number of rounds. We also provide Table 5.2 that depicts a comparison in terms of overall communication costs for various circuit sizes. In general, MS13-OSN, GKS17, BBKL18 performs  $O(N \log N)$ , whereas MS13-HE, KM11 and our protocols achieve linear complexity<sup>4</sup>. Note that although the complexity of MS13-HE is same as our InExe protocol, for the later executions our ReExe protocol enjoys a significant cost reduction due to the reusability feature, which is not possible for MS13-HE and KM11 protocols. For all circuit sizes, the communication costs of ReExe protocol are significantly lower than that

<sup>3</sup>In [17] and [9], for the 2PC phases, the authors do not suggest any optimization. However, a point and permute optimization is available for both schemes.

<sup>4</sup>Note that  $M \leq N$ , therefore  $O(M + N) = O(N)$



of existing 2PFE protocols.

The advantage of our scheme becomes more pronounced when the number of executions is more than one. To demonstrate this, we define the normalized cost efficiency ( $NCE$ ) function that takes a protocol ( $\text{Prot}_i$ ), a circuit  $C_f$  and the number of executions ( $k$ ), then outputs an efficiency ratio wrt our scheme. The normalized cost efficiency is calculated via dividing the cumulative communication cost of our protocol by that of  $\text{Prot}_i$ . i.e.,

$$NCE(\text{Prot}_i, C_f, k) = \frac{f_c(\text{InExe}, C_f) + (k - 1)f_c(\text{ReExe}, C_f)}{kf_c(\text{Prot}_i, C_f)},$$

where  $f_c$  is the cost function that outputs the communication cost value for a given protocol and  $C_f$ .

Figure 5.5 and Figure 5.6 depict the normalized cost efficiency comparison of the protocols for circuits with  $2^{10}$  and  $2^{30}$  gates, respectively. Also, without loss of generality, we take  $N = 2M$ . Considering MS13-HE, although it performs the same efficiency in the initial execution, after the second execution, its efficiency is about 0.8 (meaning that our protocol saves about 20% bandwidth as compared to MS13-HE), and after ten executions it is about 0.63 (we achieve 37% saving). Moreover, for two executions our cost reduction is about 54% over KM11-1st, 30% over KM11-2nd. For ten executions our cost reduction is about 63% over KM11-1st, 44% over KM11-2nd.

Figure 5.5 and Figure 5.6 shows how the normalized cost efficiency changes with respect to circuit size. For the protocols that have linear complexity, the normalized cost efficiency does not change as the number of gates increases. However, for the protocols with  $O(N \log N)$  complexity, their normalized efficiency dramatically decreases. For instance, after two executions, our cost reduction is about 74% and 91% over GKS17; about 91% and 97% over BBKL18; and about 95% and 98% over MS13-OSN for a thousand and a billion gate circuits, respectively. Table 5.3 depicts our efficiency gain (in percentage) over existing 2PFE schemes in terms of overall communication costs with respect to the number of protocol runs, in detail.

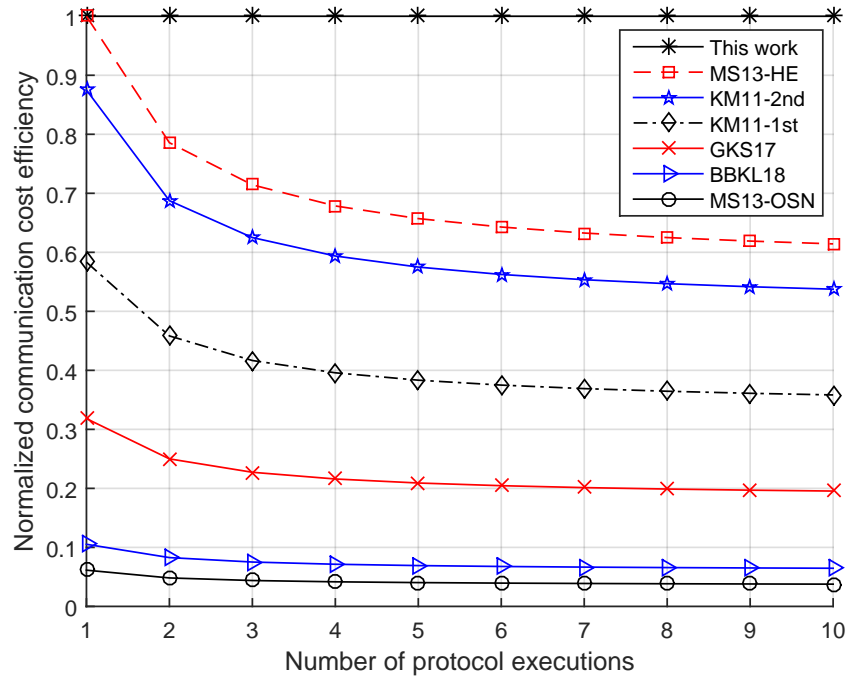


Figure 5.5: Comparison of cumulative communication cost via normalized bandwidth efficiency vs. number of PFE executions using a circuit  $2^{10}$  gates.

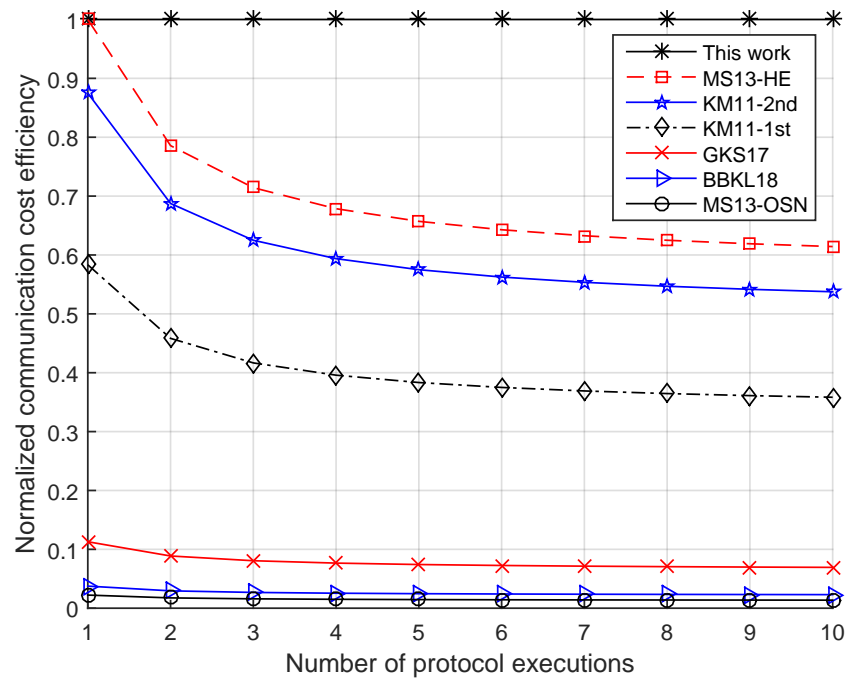


Figure 5.6: Comparison of cumulative communication cost via normalized bandwidth efficiency vs. number of PFE executions using a circuit  $2^{30}$  gates.

Table 5.3: Our efficiency gain (in percentage) over existing 2PFE schemes in terms of overall communication costs with respect to the number of protocol runs.

Existing Protocols	No. of Gates	Our Efficiency Gain (%)									
		1st run	2nd run	3rd run	4th run	5th run	6th run	7th run	8th run	9th run	10th run
MS13-HE [17]		0	20.45	27.27	30.68	32.73	34.09	35.06	35.80	36.36	36.82
KM11-2nd [9]	Any	12.00	30.00	36.00	39.00	40.80	42.00	42.86	43.50	44.00	44.40
KM11-1st [9]		42.11	53.95	57.89	59.87	61.05	61.84	62.41	62.83	63.16	63.42
GKS17 [19]		67.65	74.26	76.47	77.57	78.24	78.68	78.99	79.23	79.41	79.56
BBKL18 [20]	$2^{10}$	88.36	90.74	91.53	91.93	92.17	92.33	92.44	92.53	92.59	92.65
MS13-OSN [17]		93.82	95.08	95.51	95.72	95.84	95.93	95.99	96.03	96.07	96.10
GKS17 [19]		88.71	91.13	91.94	92.34	92.58	92.74	92.86	92.94	93.01	93.06
BBKL18 [20]	$2^{30}$	96.12	96.95	97.23	97.37	97.45	97.51	97.55	97.58	97.60	97.62
MS13-OSN [17]		97.77	98.25	98.41	98.49	98.54	98.57	98.59	98.61	98.62	98.63

For the computation costs, in order to compare symmetric-key and asymmetric-key based operations, we define the computation cost ratio  $\rho$  as the cost of an elliptic curve point multiplication divided by the cost of a symmetric-key operation for the same security level. Note that the value of  $\rho$  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 [149], 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  $\rho$  is roughly 130.

Among all protocols, our ReExe protocol performs the best result in terms of round complexity. Namely, the number of rounds in ReExe is equal to 1 if  $\text{Party}_1$  has  $x_1 = \perp$ , or 2 if  $\text{Party}_1$  has a non-empty input  $x_1$  in such that the OT extension is not applicable to its garbled input, or 3 otherwise. Note that the arithmetic circuit based protocol of [17] provides  $O(g)$  round complexity (see [17, p. 570]).

## 5.4 Security of Our Protocols

In this section, we give simulation-based security proofs of our InExe protocol in Figure 5.1, ReExe protocol in Figure 5.3, and our mechanism with various  $\text{Party}_2$ s in Sect. 5.2.3 in accordance with the security proof of [9].

**Theorem 5.4.1.** *If the following three conditions hold then the 2PFE protocol proposed in Figure 5.1 is secure against semi-honest adversaries: (1) the DDH assumption is hard in the cyclic group  $\mathbb{G}$ , (2) the hash-function  $H : \mathbb{G} \times \mathbb{G} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\ell+\tau}$  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  $\text{Party}_1$  is corrupted. For any probabilistic polynomial time adversary  $\mathcal{A}_1$ , controlling  $\text{Party}_1$  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  $\text{Party}_1$ 's inputs,  $f$  and  $x_1$ , the function output token set  $Y = (y_1, \dots, y_m)$ , the pre-shared group parameters, and  $\text{PubInfo}_{C_f}$  as follows.

1.  $\mathcal{S}_1$  generates the generator set  $\tilde{\mathcal{P}} := (\tilde{P}_1, \dots, \tilde{P}_M)$ .  $\mathcal{S}_1$  also prepares the group element sets  $\tilde{\mathcal{W}}^0 := (\tilde{W}_1^0, \dots, \tilde{W}_M^0 : \tilde{W}_i^0 \leftarrow \tilde{\alpha}_{0,i} \cdot P, \tilde{\alpha}_{0,i} \in_R \mathbb{Z}_q^*, i = 1, \dots, M)$  and  $\tilde{\mathcal{W}}^1 := (\tilde{W}_1^1, \dots, \tilde{W}_M^1 : \tilde{W}_i^1 \leftarrow \tilde{\alpha}_{1,i} \cdot P, \tilde{\alpha}_{1,i} \in_R \mathbb{Z}_q^*, i = 1, \dots, M)$ .  $\mathcal{S}_1$  gives  $\tilde{\mathcal{P}}$  to  $\mathcal{A}_1$ .
2.  $\mathcal{S}_1$  receives the blinding set  $T := (t_1, \dots, t_N : t_j \in_R \mathbb{Z}_q^*, j = 1, \dots, N)$  from  $\mathcal{A}_1$ , and prepares the sets  $\tilde{\mathcal{V}}^0 := (\tilde{V}_1^0, \dots, \tilde{V}_N^0 : \tilde{V}_j^0 \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j)}^0, j = 1, \dots, N)$  and  $\tilde{\mathcal{V}}^1 := (\tilde{V}_1^1, \dots, \tilde{V}_N^1 : \tilde{V}_j^1 \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j)}^1, j = 1, \dots, N)$ .
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  $\text{iw}_i$  and  $\text{iw}_j$ , outgoing wire is  $\text{ow}_z$ ,  $\mathcal{S}_1$  prepares the following four ciphertexts:  $\tilde{\text{ct}}_a^1 = \text{Enc}_{\tilde{V}_i^0, \tilde{V}_j^0}(\overline{\tilde{W}_z^0})$ ,  $\tilde{\text{ct}}_a^2 = \text{Enc}_{\tilde{V}_i^0, \tilde{V}_j^1}(\overline{\tilde{W}_z^0})$ ,  $\tilde{\text{ct}}_a^3 = \text{Enc}_{\tilde{V}_i^1, \tilde{V}_j^0}(\overline{\tilde{W}_z^0})$ ,  $\tilde{\text{ct}}_a^4 = \text{Enc}_{\tilde{V}_i^1, \tilde{V}_j^1}(\overline{\tilde{W}_z^0})$ . To garble an output NAND gate whose incoming wires are  $\text{iw}_i$  and  $\text{iw}_j$ , and output wire is  $z$ ,  $\mathcal{S}_1$  prepares the four ciphertexts:  $\tilde{\text{ct}}_b^1 = \text{Enc}_{\tilde{V}_i^0, \tilde{V}_j^0}(y_z)$ ,  $\tilde{\text{ct}}_b^2 = \text{Enc}_{\tilde{V}_i^0, \tilde{V}_j^1}(y_z)$ ,  $\tilde{\text{ct}}_b^3 = \text{Enc}_{\tilde{V}_i^1, \tilde{V}_j^0}(y_z)$ ,  $\tilde{\text{ct}}_b^4 = \text{Enc}_{\tilde{V}_i^1, \tilde{V}_j^1}(y_z)$ . For each garbled gate  $\tilde{G}G_a$ ,  $\mathcal{S}_1$  then permutes  $\tilde{\text{ct}}_a^2$ ,  $\tilde{\text{ct}}_a^3$ ,  $\tilde{\text{ct}}_a^4$ , and picks  $\tilde{I}_a^1, \tilde{I}_a^2 \in_R \{1, \dots, \tau\}$ , and places  $\tilde{\text{ct}}_a^1$  in the order pointed by  $\tilde{I}_a^1$ -th and  $\tilde{I}_a^2$ -th bits of  $[H(\tilde{V}_i^0, \tilde{V}_j^0, \text{gateID})]_\tau$  among the other three ciphertexts. Each garbled gate  $\tilde{G}G_a$  is then composed of four  $\ell$ -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 FALSEs 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  $\text{Party}_1$  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<sub>1</sub> obtains the group elements  $\mathcal{W} = (W_1, \dots, W_M)$  while  $\mathcal{A}_1$  obtaining the group elements  $\tilde{\mathcal{W}} = (\tilde{W}_1^0, \dots, \tilde{W}_M^0)$ . We now show the computational indistinguishability of  $(\mathcal{P}, \mathcal{W})$  and  $(\tilde{\mathcal{P}}, \tilde{\mathcal{W}})$  by utilizing Lemma 5.1.1, which ultimately ties the security of our protocol to the DDH assumption. More concretely, we need to show

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

and

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

where  $\mathbf{b}_i \in \{0, 1\}$  is the semantic value on  $\text{ow}_i$  and  $\tilde{P}_i = \tilde{r}_i \cdot P$ . For the sake of a simpler representation, we replace  $\alpha_{\mathbf{b}_i} r_i$  with  $r_{M+i}$ , and  $\tilde{\alpha}_{0,i}$  with  $\tilde{r}_{M+i}$  for  $i = 1, \dots, M$ . Note that  $(r_1, \dots, r_{2M})$  is not identically distributed to  $(\tilde{r}_1, \dots, \tilde{r}_{2M})$ , while it is only sufficient to show that

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

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

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

Thanks to Lemma 5.1.1 and the underlying DDH assumption, we have both

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

and

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

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

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

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  $\mathcal{V} = (V_1, \dots, V_N)$  is the set of tokens obtained by **Party**<sub>1</sub> and  $\tilde{\mathcal{V}} = (\tilde{V}_1^0, \dots, \tilde{V}_N^0)$  is the set of tokens obtained by  $\mathcal{A}_1$ . In contrast to [9], 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 a 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  $\tilde{G}G_a$ , 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 as follows:

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

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

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

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

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

Since in **Party**<sub>1</sub>'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**<sub>2</sub> is corrupted. For any probabilistic polynomial-time adversary  $\mathcal{A}_2$ , controlling **Party**<sub>2</sub> 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**<sub>2</sub>'s input, and the pre-shared group parameters, and  $\text{PubInfo}_{C_f}$  as follows.

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

In the real execution of our protocol, **Party**<sub>2</sub> receives only the message  $\mathcal{Q} := (Q_1, \dots, Q_N : Q_j \leftarrow t_j \cdot P_{\pi_f^{-1}(j)}, j = 1, \dots, N)$  in Round 2 (apart from the exchanged messages during the OT protocol for **Party**<sub>1</sub>'s garbled input). However, the transcripts received by **Party**<sub>2</sub> during the OT do not leak any information to **Party**<sub>2</sub> because of the ideal execution of  $\mathcal{F}_{OT}$  in the OT-hybrid model. Due to the DDH assumption, in **Party**<sub>2</sub>'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 **InExe** protocol. □

**Theorem 5.4.2.** *If the 2PFE protocol proposed in Figure 5.1 is secure against semi-honest adversaries (i.e., the three conditions in Theorem 5.4.1 are satisfied), then the 2PFE protocol proposed in Figure 5.3 is also secure against semi-honest adversaries.*

*Sketch.* The main difference of the **ReExe** protocol from the first one is the utilization of  $\text{ReuseTemp}_f$ . 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  $\text{ReuseTemp}_f$  in the  $k$ -th execution gives **Party**<sub>1</sub> no advantage in deducing **Party**<sub>2</sub>'s inputs.

We now show that in **Party**<sub>1</sub>'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  $\tilde{\mathcal{W}}_1 :=$



$(\tilde{W}_{1,1}, \dots, \tilde{W}_{M,1} : \tilde{W}_{i,1} = \tilde{q}_{i,1} \cdot P, \tilde{q}_{i,1} \in_R \mathbb{Z}_q^*, i = 1, \dots, M), \tilde{\mathcal{V}}_1 := (\tilde{V}_{1,1}, \dots, \tilde{V}_{N,1} : \tilde{V}_{j,1} \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j),1}, j = 1, \dots, N), \tilde{\mathcal{W}}_2 := (\tilde{W}_{1,2}, \dots, \tilde{W}_{M,2} : \tilde{W}_{i,2} = \tilde{q}_{i,2} \cdot P, \tilde{q}_{i,2} \in_R \mathbb{Z}_q^*, i = 1, \dots, M),$  and  $\tilde{\mathcal{V}}_2 := (\tilde{V}_{1,2}, \dots, \tilde{V}_{N,2} : \tilde{V}_{j,2} \leftarrow t_j \cdot \tilde{W}_{\pi_f^{-1}(j),2}, j = 1, \dots, N).$

More concretely, we have

$$\begin{aligned}
& \{(\overline{(\overline{1, k})}), \dots, (\overline{(\overline{M, k})}), t_1 \cdot (\overline{(\overline{\pi_f^{-1}(1), k})}), \dots, t_N \cdot (\overline{(\overline{\pi_f^{-1}(N), k})}), \\
& (\overline{(\overline{1, k+1})}), \dots, (\overline{(\overline{M, k+1})}), t_1 \cdot (\overline{(\overline{\pi_f^{-1}(1), k+1})}), \dots, t_N \cdot (\overline{(\overline{\pi_f^{-1}(N), k+1})})\} \\
& \approx_c \{(\tilde{q}_{1,1} \cdot P, \dots, \tilde{q}_{M,1} \cdot P, t_1 \cdot (\tilde{q}_{\pi_f^{-1}(1),1} \cdot P), \\
& \dots, t_N \cdot (\tilde{q}_{\pi_f^{-1}(N),1} \cdot P), \tilde{q}_{1,2} \cdot P, \dots, \tilde{q}_{M,2} \cdot P, \\
& t_1 \cdot (\tilde{q}_{\pi_f^{-1}(1),2} \cdot P), \dots, t_N \cdot (\tilde{q}_{\pi_f^{-1}(N),2} \cdot P))\}
\end{aligned}$$

where  $(\overline{(i, j)})$  is the abbreviation for  $\alpha_{\mathbf{b}_{i,j}} \cdot P_i$ , and  $\mathbf{b}_{i,k} \in \{0, 1\}$  is the semantic bit value of  $\mathbf{ow}_i$  in the  $k$ -th execution. The proof of their indistinguishability relies on the same flow as the proof of Theorem 5.4.1, which depends on Lemma 5.1.1 and ultimately on the DDH assumption.

□

**Theorem 5.4.3.** *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 5.3 is secure against semi-honest adversaries; then our mechanism with various Party<sub>2</sub>s in Sect. 5.2.3 is also secure against semi-honest adversaries.*

*Sketch.* First, Party<sub>1</sub>'s view in the 2PFE mechanism is equivalent to the one in the protocol in Figure 5.3. Observe that the generator set is generated by the distributed system and the tokens (that are used in the 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 5.3. Therefore, the 2PFE mechanism prevents Party<sub>1</sub> from deducing any information about Party<sub>2,k</sub>'s input. Second, Party<sub>2,k</sub>s cannot obtain any information about Party<sub>1</sub>'s input in none of the executions since the OT outputs are only obtained

by  $\text{Party}_1$  due the  $\mathcal{F}_{OT}$  functionality in the OT-hybrid model. Also, due to Theorem 5.4.1, no one can obtain information about  $\pi_f$  from the  $\text{ReuseTemp}_f$ . Moreover, any  $\text{Party}_{2,k}$  has a negligible advantage on distinguishing the exchanged messages in an evaluation between  $\text{Party}_1$  and  $\text{Party}_{2,l}$  from a random string due to the underlying DDH assumption for  $l \neq k$ . More concretely, the tokens (that are used in the preparation of  $\text{Party}_{2,l}$ 's garbled input  $X_{2,l}$  and the garbled circuit  $F_l$ ) are computed by multiplying the elements of the  $\text{ReuseTemp}_f$  with the private values  $\alpha_{0,l}$  or  $\alpha_{1,l}$  of  $\text{Party}_{2,l}$ .  $\square$

# Chapter 6

## CONCLUSIONS

In this dissertation, we studied the problem of the private function evaluation. 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.

One of the primary objectives of the recent research on MPC, and specifically PFE, is minimizing the communication cost. This is due to the fact that historical developments in hardware technology show us computing power advances faster than communication channels. This is even more likely to be so in the near future, *i.e.*, the main bottleneck for many secure computation applications will not be the CPU load but be the bandwidth constraints [28, 108]. Motivated by this, we are mainly interested in reducing the communication complexity of the 2PFE protocols.

We first proposed an efficient and secure protocol for 2PFE based on (mostly) symmetric cryptography primitives. In this respect, we proposed an efficient protocol by adapting the state-of-the-art half gates garbling optimization [1] to Mohassel and Sadeghian's 2PFE scheme [17] 2PFE scheme. Our optimization achieves a remarkable advantage over [17] in both OSN and 2PC phases in terms of communication complexity. In particular, in the 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.

Next, we have proposed a secure and highly efficient 2PFE scheme for Boolean circuits based on the DDH assumption. Our scheme consists of two protocols: (1) a protocol for initial executions (**InExe**), (2) a resumption protocol (**ReExe**) for subsequent executions. The latter protocol is more efficient due to the fact that it benefits from the reusable tokens generated already in the former one. 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 a long-term business relationship instead of a single deal. Note that such a cost reduction is not available in the protocols of KM11 [9] and MS13 [17] since they require all token generation and 2PC procedures repeated in all executions. However, our scheme involves **ReuseTemp<sub>f</sub>** that is reusable for the generation of tokens on incoming and outgoing wires. The reusability of **ReuseTemp<sub>f</sub>** incurs a massive reduction in protocol overhead since a large part of costs in existing 2PFE protocols [9, 17] results from the generation of these tokens. Our protocols achieve linear communication and computation complexities and a constant number of rounds which is at most three. To the best of our knowledge, this is the first and most efficient 2PFE scheme that enjoys a reusability feature.

Comparing the existing protocols, our scheme asymptotically reduces the communication cost compared to MS13-OSN [17], BBKL18 [20], and GKS17 [19] protocols (i.e., from  $O(g \log(g))$  to  $O(g)$  where  $g$  is the number of gates). For instance, for a billion-gate circuit, our cost reduction is about 98% over MS13-OSN, about 96% over BBKL18, and about 89% over GKS17. Comparing with the protocols that have linear complexity, for ten executions (regardless of the number of gates) our

cost reduction is about 63% over KM11-1st, 44% over KM11-2nd, and 37% over MS13-HE.

We also propose a solution for the case that  $\text{Party}_1$  runs the 2PFE protocol for the same private function with various  $\text{Party}_2$ s separately. This is a common scenario where  $\text{Party}_1$  may run a business with many customers for her algorithm/software. Instead of running InExe protocol with each  $\text{Party}_2$ , we have proposed a more efficient mechanism for the generation of the reusable tokens by utilizing a threshold based system.

After all, we hope that our work sheds light on the future researches and leads to more practical PFE constructions. In accordance with this goal, we conclude with the following open questions:

1. Although the 2-OEP protocol in [17], 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 symmetric cryptography based 2-OEP protocol that has linear asymptotic complexity while also being efficient in small circuit sizes?*
2. Our and existing 2PFE protocols permit 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 2PFE schemes permitting the use of various gates in logic circuit representations?*

# BIBLIOGRAPHY

- [1] S. Zahur, M. Rosulek, and D. Evans, “Two halves make a whole: Reducing data transfer in garbled circuits using half gates,” in *Proceedings of the Advances in Cryptology - EUROCRYPT 2015*, (Sofia, Bulgaria), pp. 220–250, Springer, Berlin Heidelberg, 26 - 30 April 2015.
- [2] M. Bellare, V. T. Hoang, and P. Rogaway, “Foundations of garbled circuits,” in *Proceedings of the ACM Conference on Computer and Communications Security (CCS’12)*, (Raleigh, North Carolina, USA), pp. 784–796, ACM, NY, USA, 16-18 October 2012.
- [3] C. Cachin, J. Camenisch, J. Kilian, and J. Müller, “One-round secure computation and secure autonomous mobile agents,” in *Proceedings of the 27th International Colloquium on Automata, Languages and Programming, ICALP ’00*, (London, UK, UK), pp. 512–523, Springer-Verlag, 2000.
- [4] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright, “Selective private function evaluation with applications to private statistics,” in *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC ’01*, (New York, NY, USA), pp. 293–304, ACM, 2001.
- [5] Y.-C. Chang and C.-J. Lu, “Oblivious polynomial evaluation and oblivious neural learning,” in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT ’01*, (Berlin, Heidelberg), pp. 369–384, Springer-Verlag, 2001.
- [6] K. Frikken, M. Atallah, and J. Li, “Hidden access control policies with hidden credentials,” in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES ’04*, (New York, NY, USA), pp. 27–27, ACM, 2004.
- [7] K. Frikken, M. Atallah, and C. Zhang, “Privacy-preserving credit checking,” in *Proceedings of the 6th ACM Conference on Electronic Commerce, (EC’05)*, (Vancouver, BC, Canada), pp. 147–154, ACM, New York, 05-08 June 2005.

- [8] K. B. Frikken, J. Li, and M. J. Atallah, “Trust negotiation with hidden credentials, hidden policies, and policy cycles,” in *In Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, pp. 157–172, 2006.
- [9] J. Katz and L. Malka, “Constant-round private function evaluation with linear complexity,” in *Proceedings of the Advances in Cryptology – ASIACRYPT 2011*, (Seoul, South Korea), pp. 556–571, Springer, Berlin, Heidelberg, 4-8 December 2011.
- [10] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, “Privacy-preserving remote diagnostics,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS ’07*, (New York, NY, USA), pp. 498–507, ACM, 2007.
- [11] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. 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)*, (Saint-Malo, France), pp. 424–439, Springer-Verlag, Berlin, Heidelberg, 21-23 September 2009.
- [12] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin, “Blind Seer: A scalable private DBMS,” in *Proceedings of IEEE Symposium on Security and Privacy (SP’14)*, (San Jose, CA), pp. 359–374, IEEE Computer Society, Washington, DC, 18-21 May 2014.
- [13] S. Niksefat, B. Sadeghiyan, P. Mohassel, and S. S. Sadeghian, “ZIDS: A privacy-preserving intrusion detection system using secure two-party computation protocols,” *Comp. J.*, vol. 57, no. 4, pp. 494–509, 2014.
- [14] V. Kolesnikov and T. Schneider, “A practical universal circuit construction and secure evaluation of private functions,” in *Proceedings of the Financial Cryptography and Data Security (FC’08)*, (Cozumel, Mexico), pp. 83–97, Springer-Verlag Berlin Heidelberg, 28-31 January 2008.
- [15] A.-R. Sadeghi and T. 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)*, (Seoul, Korea), pp. 336–353, Springer-Verlag, Berlin, Heidelberg, 3-5 December 2009.
- [16] A. Paus, A.-R. Sadeghi, and T. Schneider, “Practical secure evaluation of semi-private functions,” in *Proceedings of the Applied Cryptography and Network Security (ACNS’09)*, (Paris-Rocquencourt, France), pp. 89–106, Springer-Verlag Berlin, Heidelberg, 2-5 June 2009.

- [17] P. Mohassel and S. Sadeghian, “How to hide circuits in mpc an efficient framework for private function evaluation,” in *Proceedings of the Advances in Cryptology – EUROCRYPT 2013*, (Athens, Greece), pp. 557–574, Springer, Berlin, Heidelberg, 26-30 May 2013.
- [18] Á. Kiss and T. 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*, (Berlin, Heidelberg), pp. 699–728, Springer Berlin Heidelberg, 2016.
- [19] D. Günther, Á. Kiss, and T. Schneider, “More efficient universal circuit constructions,” in *Proceedings of the Advances in Cryptology – ASIACRYPT 2017*, (Hong Kong, China), pp. 443–470, Springer-Verlag Berlin, Heidelberg, 3-7 December 2017.
- [20] M. A. Bingöl, O. Biçer, M. S. Kiraz, and A. Levi, “An efficient 2-party private function evaluation protocol based on half gates,” *The Computer Journal*, no. bxy136, 2018. Available at <http://dx.doi.org/10.1093/comjnl/bxy136>.
- [21] O. Bicer, M. A. Bingöl, M. S. Kiraz, and A. Levi, “Highly efficient and reusable private function evaluation with linear complexity.” Cryptology ePrint Archive, Report 2018/515, 2018. Available at <https://eprint.iacr.org/2018/515>.
- [22] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC ’09, (New York, NY, USA), pp. 169–178, ACM, 2009.
- [23] C. Gentry, *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, Stanford, CA, USA, Available at <http://crypto.stanford.edu/craig>, 2009.
- [24] S. Halevi and V. 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*, (Berlin, Heidelberg), pp. 641–670, Springer Berlin Heidelberg, 2015.
- [25] H. Lipmaa, P. Mohassel, and S. Sadeghian, “Valiant’s universal circuit: Improvements, implementation, and applications.” Cryptology ePrint Archive, Report 2016/017, 2016. <http://eprint.iacr.org/2016/017>.
- [26] S. Sadeghian, *New Techniques for Private Function Evaluation*. PhD thesis, University of Calgary, 2015. <https://prism.ucalgary.ca/handle/11023/2657>.



- [27] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications,” in *International Colloquium on Automata, Languages, and Programming (ICALP’08)*, (Reykjavik, Iceland), pp. 486–498, Springer Berlin, Heidelberg, 06-13 July 2008.
- [28] V. Kolesnikov, P. Mohassel, and M. Rosulek, “FleXOR: Flexible garbling for XOR gates that beats free-XOR,” in *Advances in Cryptology CRYPTO 2014* (J. Garay and R. Gennaro, eds.), vol. 8617 of *Lecture Notes in Computer Science*, pp. 440–457, Springer Berlin Heidelberg, 2014.
- [29] 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, pp. 503–513, ACM, 1990.
- [30] Y. Wang and Q. m. Malluhi, “Reducing garbled circuit size while preserving circuit gate privacy.” Cryptology ePrint Archive, Report 2017/041, 2017. <http://eprint.iacr.org/2017/041>.
- [31] J. T. Gill, III, “Computational complexity of probabilistic turing machines,” in *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC ’74, (New York, NY, USA), pp. 91–95, ACM, 1974.
- [32] B. Schoenmakers, “Lecture notes cryptographic protocols.” Department of Mathematics and Computer Science, Technical University of Eindhoven, version 1.32, February 2018. <https://www.win.tue.nl/~berry/2WC13/LectureNotes.pdf>.
- [33] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd ed., 2014.
- [34] R. C. Merkle, “One way hash functions and DES,” in *Proceedings on Advances in Cryptology*, CRYPTO ’89, (New York, NY, USA), pp. 428–446, Springer-Verlag New York, Inc., 1989.
- [35] M. Naor and M. Yung, “Universal one-way hash functions and their cryptographic applications,” in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC ’89, (New York, NY, USA), pp. 33–43, ACM, 1989.
- [36] I. B. Damgård, “A design principle for hash functions,” in *Advances in Cryptology — CRYPTO’ 89 Proceedings* (G. Brassard, ed.), (New York, NY), pp. 416–427, Springer New York, 1990.
- [37] X. Lai and J. L. Massey, “Hash functions based on block ciphers,” in *Advances in Cryptology — EUROCRYPT’ 92* (R. A. Rueppel, ed.), (Berlin, Heidelberg), pp. 55–70, Springer Berlin Heidelberg, 1992.

- [38] R. Canetti, O. Goldreich, and S. Halevi, “The random oracle methodology, revisited,” *J. ACM*, vol. 51, pp. 557–594, July 2004.
- [39] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS '93*, (New York, NY, USA), pp. 62–73, ACM, 1993.
- [40] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, pp. 203–209, Jan. 1987.
- [41] V. S. Miller, “Use of elliptic curves in cryptography,” in *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*, (Berlin, Heidelberg), pp. 417–426, Springer-Verlag, 1986.
- [42] H. W. Lenstra Jr., “Factoring integers with elliptic curves,” *Annals of Mathematics*, vol. 126, no. 3, pp. 649–6733, 1987.
- [43] J. Silverman, *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics, Springer New York, 2009.
- [44] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd ed., 2012.
- [45] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Comput. Surv.*, vol. 51, pp. 79:1–79:35, July 2018.
- [46] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.
- [47] R. Rivest, L. Adleman, and M. Dertouzos, “On data banks and privacy homomorphisms,” *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–177, 1978.
- [48] S. Goldwasser and S. Micali, “Probabilistic encryption & how to play mental poker keeping secret all partial information,” in *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82*, (New York, NY, USA), pp. 365–377, ACM, 1982.
- [49] E. F. Brickell and Y. Yacobi, “On privacy homomorphisms,” in *Proceedings of the 6th Annual International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'87*, (Berlin, Heidelberg), pp. 117–125, Springer-Verlag, 1988.

- [50] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Trans. Inf. Theor.*, vol. 31, pp. 469–472, September 1985.
- [51] T. El Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *Proceedings of the Advances in Cryptology – CRYPTO 1984*, (Santa Barbara, California, USA), pp. 10–18, Springer-Verlag Berlin, Heidelberg, 19-22 August 1984.
- [52] J. B. Clarkson, “Dense probabilistic encryption,” in *In Proceedings of the Workshop on Selected Areas of Cryptography*, pp. 120–128, 1994.
- [53] D. Naccache and J. Stern, “A new public key cryptosystem based on higher residues,” in *Proceedings of the 5th ACM Conference on Computer and Communications Security, CCS ’98*, (New York, NY, USA), pp. 59–66, ACM, 1998.
- [54] T. Okamoto and S. Uchiyama, “A new public-key cryptosystem as secure as factoring,” in *Advances in Cryptology — EUROCRYPT ’98* (K. Nyberg, ed.), (Berlin, Heidelberg), pp. 308–318, Springer Berlin Heidelberg, 1998.
- [55] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Advances in Cryptology — EUROCRYPT ’99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings* (J. Stern, ed.), (Berlin, Heidelberg), pp. 223–238, Springer Berlin Heidelberg, 1999.
- [56] I. Damgård and M. Jurik, “A generalisation, a simplification and some applications of paillier’s probabilistic public-key system,” in *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography, PKC ’01*, (London, UK, UK), pp. 119–136, Springer-Verlag, 2001.
- [57] A. Kawachi, K. Tanaka, and K. Xagawa, “Multi-bit cryptosystems based on lattice problems,” in *Proceedings of the 10th International Conference on Practice and Theory in Public-key Cryptography, PKC’07*, (Berlin, Heidelberg), pp. 315–329, Springer-Verlag, 2007.
- [58] L. Fousse, P. Lafourcade, and M. Alnuaimi, “Benaloh’s dense probabilistic encryption revisited,” in *Progress in Cryptology – AFRICACRYPT 2011* (A. Nitaj and D. Pointcheval, eds.), (Berlin, Heidelberg), pp. 348–362, Springer Berlin Heidelberg, 2011.
- [59] M. Fellows and N. Koblitz, “Combinatorial cryptosystems galore!,” in *Proceedings of the Second International Symposium on Finite Fields, Las Vegas, Nevada, August, 1993*, vol. 168 of *Contemporary Mathematics*, p. 5161, American Mathematical Society, 1994.

- [60] T. Sander, A. Young, and M. Yung, “Non-interactive cryptocomputing for nc1,” in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS ’99, (Washington, DC, USA), pp. 554–566, IEEE Computer Society, 1999.
- [61] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-dnf formulas on ciphertexts,” in *Proceedings of the Second International Conference on Theory of Cryptography*, TCC’05, (Berlin, Heidelberg), pp. 325–341, Springer-Verlag, 2005.
- [62] Y. Ishai and A. Paskin, “Evaluating branching programs on encrypted data,” in *Proceedings of the 4th Conference on Theory of Cryptography*, TCC’07, (Berlin, Heidelberg), pp. 575–594, Springer-Verlag, 2007.
- [63] C. A. Melchor, P. Gaborit, and J. Herranz, “Additively homomorphic encryption with d-operand multiplications,” in *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO’10, (Berlin, Heidelberg), pp. 138–154, Springer-Verlag, 2010.
- [64] C. Gentry, “Toward basing fully homomorphic encryption on worst-case hardness,” in *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO’10, (Berlin, Heidelberg), pp. 116–137, Springer-Verlag, 2010.
- [65] N. P. Smart and F. Vercauteren, “Fully homomorphic encryption with relatively small key and ciphertext sizes,” in *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, PKC’10, (Berlin, Heidelberg), pp. 420–443, Springer-Verlag, 2010.
- [66] C. Gentry and S. Halevi, “Implementing gentry’s fully-homomorphic encryption scheme,” in *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT’11, (Berlin, Heidelberg), pp. 129–148, Springer-Verlag, 2011.
- [67] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, “Fully homomorphic encryption over the integers with shorter public keys,” in *Proceedings of the 31st Annual Conference on Advances in Cryptology*, CRYPTO’11, (Berlin, Heidelberg), pp. 487–504, Springer-Verlag, 2011.
- [68] C. Gentry, S. Halevi, and N. P. Smart, “Homomorphic evaluation of the aes circuit,” in *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, (Berlin, Heidelberg), pp. 850–867, Springer-Verlag, 2012.
- [69] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” in *Proceedings of the 3rd Innovations in*

- Theoretical Computer Science Conference, ITCS '12*, (New York, NY, USA), pp. 309–325, ACM, 2012.
- [70] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Trans. Comput. Theory*, vol. 6, pp. 13:1–13:36, July 2014.
- [71] Y. Doröz, E. Öztürk, E. Savaş, and B. Sunar, “Accelerating Itv based homomorphic encryption in reconfigurable hardware,” in *Cryptographic Hardware and Embedded Systems – CHES 2015* (T. Güneysu and H. Handschuh, eds.), (Berlin, Heidelberg), pp. 185–204, Springer Berlin Heidelberg, 2015.
- [72] M. 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>.
- [73] S. Even, O. Goldreich, and A. Lempel, “A randomized protocol for signing contracts,” *Commun. ACM*, vol. 28, pp. 637–647, jun 1985.
- [74] C. Crépeau, “Equivalence between two flavours of oblivious transfers,” in *Proceedings of the Advances in Cryptology – CRYPTO 1987*, pp. 350–354, Springer-Verlag, 1987.
- [75] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, “Extending oblivious transfers efficiently,” in *Proceedings of the Advances in Cryptology – CRYPTO 2003*, pp. 145–161, Santa Barbara, CA, USA: Springer, Berlin, Heidelberg, 17-21 August 2003.
- [76] V. Kolesnikov and R. Kumaresan, “Improved OT extension for transferring short secrets,” in *Proceedings of the Advances in Cryptology - CRYPTO 2013*, (Santa Barbara, CA, USA), pp. 54–70, Springer, Berlin, Heidelberg, 18-12 August 2013.
- [77] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, “More efficient oblivious transfer and extensions for faster secure computation,” in *Proceedings of the ACM Computer and Communications Security Conference (CCS '13)*, (Berlin, Germany), pp. 535–548, ACM, NY, USA, 4-8 November 2013.
- [78] W. Du and M. J. Atallah, “Secure Multi-party Computation Problems and Their Applications: A Review and Open Problems,” in *Proceedings of the 2001 Workshop on New Security Paradigms, NSPW '01*, (New York, NY, USA), pp. 13–22, ACM, 2001.
- [79] L. Kamm and J. Willemson, “Secure floating point arithmetic and private satellite collision analysis,” *International Journal of Information Security*, pp. 1–18, 2014.

- [80] R. Küsters, T. Truderung, and A. Vogt, “A game-based definition of coercion resistance and its applications,” *J. Comput. Secur.*, vol. 20, pp. 709–764, Nov. 2012.
- [81] R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” in *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT’97, (Berlin, Heidelberg), pp. 103–118, Springer-Verlag, 1997.
- [82] S. Kardas, M. S. Kiraz, M. A. Bingöl, and F. Birinci, “Norwegian internet voting protocol revisited: ballot box and receipt generator are allowed to col-lude,” *Security and Communication Networks*, vol. 9, no. 18, pp. 5051–5063, 2016.
- [83] P. S. Naidu, R. Kharat, R. Tekade, P. Mendhe, and V. Magade, “E-voting system using visual cryptography amp; secure multi-party computation,” in *2016 International Conference on Computing Communication Control and automation (ICCCUBEA)*, pp. 1–4, Aug 2016.
- [84] M. Naor, B. Pinkas, and R. Sumner, “Privacy preserving auctions and mecha-nism design,” in *Proceedings of the ACM Conference on Electronic Commerce (EC’99)*, (Denver, Colorado, USA), pp. 129–139, ACM Press, NY, USA, 3-5 November 1999.
- [85] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakob-sen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, “Financial cryptography and data security,” ch. Secure Multiparty Computation Goes Live, pp. 325–343, Berlin, Heidel-berg: Springer-Verlag, 2009.
- [86] D. Bogdanov, R. Talviste, and J. Willemson, “Deploying secure multi-party computation for financial data analysis,” in *Financial Cryptography and Data Security* (A. D. Keromytis, ed.), (Berlin, Heidelberg), pp. 57–64, Springer Berlin Heidelberg, 2012.
- [87] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” in *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryp-tology*, CRYPTO ’00, (London, UK, UK), pp. 36–54, Springer-Verlag, 2000.
- [88] Y. Lindell and B. Pinkas, “Secure multiparty computation for privacy-preserving data mining,” *The Journal of Privacy and Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.
- [89] D. Bogdanov, “How to securely perform computations on secret-shared data,” Master’s thesis, University of Tartu, Estonia, 2007. [https://cyber.ee/research/theses/dan\\_bogdanov\\_msc.pdf](https://cyber.ee/research/theses/dan_bogdanov_msc.pdf).

- [90] D. Bogdanov, *Sharemind: Programmable secure computations with practical applications*. PhD thesis, University of Tartu, Estonia, 2013.
- [91] D. Bogdanov, S. Laur, and J. Willemson, “Sharemind: A framework for fast privacy-preserving computations,” in *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS ’08, (Berlin, Heidelberg), pp. 192–206, Springer-Verlag, 2008.
- [92] D. C. Snchez, “Raziel: Private and verifiable smart contracts on blockchains.” Cryptology ePrint Archive, Report 2017/878, 2017. <https://eprint.iacr.org/2017/878>.
- [93] F. Benhamouda, S. Halevi, and T. Halevi, “Supporting private data on hyperledger fabric with secure multiparty computation,” in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 357–363, April 2018.
- [94] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol.” Cryptology ePrint Archive, Report 2016/889, 2016. <https://eprint.iacr.org/2016/889>.
- [95] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, “Coin-Party: Secure multi-party mixing of bitcoins,” in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY ’15, (New York, NY, USA), pp. 75–86, ACM, 2015.
- [96] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, “Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’18, (New York, NY, USA), pp. 913–930, ACM, 2018.
- [97] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, “Secure multiparty computations on bitcoin,” *Commun. ACM*, vol. 59, pp. 76–84, Mar. 2016.
- [98] V. K. David Evans and M. Rosulek, *A Pragmatic Introduction to Secure Multi-Party Computation*. NOW Publishers, 1st ed., 2018.
- [99] O. Bicer, “Efficiency optimizations on yaos garbled circuits and their practical applications,” Master’s thesis, Istanbul Sehir University, Turkey, 2017. <http://earsiv.sehir.edu.tr:8080/xmlui/bitstream/handle/11498/39690/000130184002.pdf?sequence=1>.
- [100] T. Schneider, *Engineering Secure Two-Party Computation Protocols – Advances in Design, Optimization, and Applications of Efficient Secure Function Evaluation*. PhD thesis, Ruhr-University Bochum, Germany, Information Sciences, 2011. Available at <http://thomaschneider.de/papers/S11Thesis.pdf>.

- [101] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, “From keys to databases - real-world applications of secure multi-party computation,” *Comput. J.*, vol. 61, no. 12, pp. 1749–1771, 2018.
- [102] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, “Fairplay—a Secure Two-party Computation System,” in *Proceedings of the 13th Conference on USENIX Security Symposium*, (Berkeley, CA, USA), pp. 287–302, USENIX Association, 2004.
- [103] A. C. Yao, “Protocols for secure computations,” in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS’82)*, (Chicago, IL), pp. 160–164, IEEE Computer Society, Washington, DC, USA, 3-5 November 1982.
- [104] A. C. Yao, “How to generate and exchange secrets,” in *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pp. 162–167, Oct 1986.
- [105] S. Goldwasser, “Multi-Party Computations: Past and Present,” in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pp. 1–6, 1997.
- [106] O. Goldreich, “Secure multi-party computation (working draft),” 2005.
- [107] 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)*, (Baltimore, Maryland, USA), pp. 503–513, ACM, NY, USA, 13-16 May 1990.
- [108] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, “Secure two-party computation is practical,” in *Proceedings of the Advances in Cryptology – ASIACRYPT 2009*, (Tokyo, Japan), pp. 250–267, Springer-Verlag Berlin, Heidelberg, 6-10 December 2009.
- [109] Y. Huang, D. Evans, J. Katz, and L. Malka, “Faster secure two-party computation using garbled circuits,” in *Proceedings of the 20th USENIX Conference on Security (SEC’11)*, (San Francisco, CA), pp. 35–50, USENIX Association Berkeley, CA, USA, 8-12 August 2011.
- [110] T. Frederiksen, T. Jakobsen, J. Nielsen, P. Nordholt, and C. Orlandi, “Mini-LEGO: Efficient Secure Two-Party Computation from General Assumptions,” in *Advances in Cryptology EUROCRYPT 2013*, vol. 7881 of *Lecture Notes in Computer Science*, pp. 537–556, Springer Berlin Heidelberg, 2013.
- [111] S. G. Choi, J. Katz, R. Kumaresan, and H.-S. Zhou, “On the security of the “Free-XOR” technique,” in *Proceedings of the Theory of Cryptography*



- Conference (TCC'12)*, (Taormina, Sicily, Italy), pp. 39–53, Springer-Verlag Berlin, Heidelberg, 12-19 March 2012.
- [112] A. Shamir, “How to Share a Secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
  - [113] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, “From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design,” 2010.
  - [114] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. New York, NY, USA: Cambridge University Press, 2004.
  - [115] Y. Aumann and Y. Lindell, “Security against covert adversaries: Efficient protocols for realistic adversaries,” *J. Cryptol.*, vol. 23, no. 2, pp. 281–343, 2010.
  - [116] abhi shelat and C. hao Shen, “Two-output secure computation with malicious adversaries,” in *Advances in Cryptology EUROCRYPT 2011*, vol. 6632 of *LNCS*, pp. 386–405, Springer Berlin Heidelberg, 2011.
  - [117] Y. Lindell and B. Pinkas, “An efficient protocol for secure two-party computation in the presence of malicious adversaries,” in *Advances in Cryptology - EUROCRYPT 2007*, vol. 4515 of *LNCS*, pp. 52–78, Springer Berlin Heidelberg, 2007.
  - [118] Y. Lindell and B. Pinkas, “An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries,” in *Advances in Cryptology Eurocrypt 2007*, vol. 4515 of *LNCS*, pp. 52–78, Springer-Verlag, 2007.
  - [119] M. S. Kiraz and B. Schoenmakers, “A protocol issue for the malicious case of yaos garbled circuit construction,” in *In Proceedings of 27th Symposium on Information Theory in the Benelux*, pp. 283–290, 2006.
  - [120] M. S. Kiraz, *Secure and Fair Two-Party Computation*. PhD thesis, Eindhoven University of Technology, Netherlands, 2008. Available at <http://alexandria.tue.nl/extra2/200811317.pdf>.
  - [121] M. S. Kiraz and B. Schoenmakers, “An efficient protocol for fair secure two-party computation,” in *Proceedings of the 2008 The Cryptographers’ Track at the RSA Conference on Topics in Cryptology, CT-RSA’08*, (Berlin, Heidelberg), pp. 88–105, Springer-Verlag, 2008.
  - [122] Y. Lindell, B. Pinkas, and N. Smart, “Implementing two-party computation efficiently with security against malicious adversaries,” in *Security and Cryptography for Networks*, vol. 5229 of *LNCS*, pp. 2–20, Springer Berlin Heidelberg, 2008.

- [123] V. Goyal, P. Mohassel, and A. Smith, “Efficient two party and multi party computation against covert adversaries,” in *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology, EUROCRYPT’08*, (Berlin, Heidelberg), pp. 289–306, Springer-Verlag, 2008.
- [124] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, “Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits,” in *Computer Security – ESORICS 2013* (J. Crampton, S. Jajodia, and K. Mayes, eds.), (Berlin, Heidelberg), pp. 1–18, Springer Berlin Heidelberg, 2013.
- [125] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, “Secure evaluation of private linear branching programs with medical applications,” in *Computer Security – ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*, (Berlin, Heidelberg), pp. 424–439, Springer Berlin Heidelberg, 2009.
- [126] D. K. Altop, M. A. Bingöl, A. Levi, and E. Savas, “DKEM: secure and efficient distributed key establishment protocol for wireless mesh networks,” *Ad Hoc Networks*, vol. 54, pp. 53–68, 2017.
- [127] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, “Privacy-preserving face recognition,” in *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies, PETS ’09*, (Berlin, Heidelberg), pp. 235–253, Springer-Verlag, 2009.
- [128] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, “Efficient privacy-preserving face recognition,” in *Proceedings of the 12th International Conference on Information Security and Cryptology, ICISC’09*, (Berlin, Heidelberg), pp. 229–244, Springer-Verlag, 2010.
- [129] C. Kempka, R. Kikuchi, and K. 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*, (Berlin, Heidelberg), pp. 967–997, Springer Berlin Heidelberg, 2016.
- [130] M. Ball, T. Malkin, and M. Rosulek, “Garbling gadgets for boolean and arithmetic circuits,” in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS’16)*, (Vienna, Austria), pp. 565–577, ACM, NY, USA, 24-28 October 2016.

- [131] 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, pp. 195–203, ACM, 1987.
- [132] M. Abadi and J. Feigenbaum, “Secure circuit evaluation,” *Journal of Cryptology*, vol. 2, pp. 1–12, Feb 1990.
- [133] L. G. Valiant, “Universal Circuits (Preliminary Report),” in *Proceedings of the ACM Symposium on Theory of Computing (STOC’76)*, (Hershey, Pennsylvania, USA), pp. 196–203, ACM New York, NY, USA, 3-5 May 1976.
- [134] T. Schneider, “Practical secure function evaluation,” Master’s thesis, Friedrich-Alexander University Erlangen-Nürnberg, Germany, February 27, 2008. <http://thomaschneider.de/papers/S08Thesis.pdf>.
- [135] P. Mohassel, S. Sadeghian, and N. P. Smart, “Actively secure private function evaluation,” in *Proceedings of the Advances in Cryptology – ASIACRYPT 2014*, (Kaoshiung, Taiwan), pp. 486–505, Springer Berlin, Heidelberg, 7-11 December 2014.
- [136] T. Schneider and M. Zohner, “GMW vs. Yao? Efficient Secure Two-Party Computation with Low Depth Circuits,” in *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers* (A.-R. Sadeghi, ed.), (Berlin, Heidelberg), pp. 275–292, Springer Berlin Heidelberg, 2013.
- [137] S. G. Choi, J. Katz, A. J. Malozemoff, and V. Zikas, “Efficient three-party computation from cut-and-choose,” in *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II* (J. A. Garay and R. Gennaro, eds.), (Berlin, Heidelberg), pp. 513–530, Springer Berlin Heidelberg, 2014.
- [138] Y. Lindell and B. Pinkas, “A proof of security of Yao’s protocol for two-party computation,” *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.
- [139] A. Waksman, “A permutation network,” *Journal of the ACM*, vol. 15, pp. 159–163, Jan. 1968.
- [140] V. T. Hoang, *Foundations of Garbled Circuits*. PhD thesis, University of California Davis, 2013.
- [141] M. Naor and O. Reingold, “Number-theoretic constructions of efficient pseudo-random functions,” *J. ACM*, vol. 51, pp. 231–262, Mar. 2004.
- [142] D. Boneh, “The decision diffie-hellman problem,” in *Algorithmic Number Theory: Third International Symposium, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, (Berlin, Heidelberg), pp. 48–63, Springer Berlin Heidelberg, 1998.

- [143] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.
- [144] A. K. Lenstra and E. R. Verheul, “Selecting cryptographic key sizes,” *Journal of Cryptology*, vol. 14, no. 4, pp. 255–293, 2001.
- [145] D. Giry, “Keylength – cryptographic key length recommendation,” 2016. <http://www.keylength.com/> (accessed on 2017-05-17).
- [146] 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*, (Berlin, Heidelberg), pp. 2–20, Springer Berlin Heidelberg, 2008.
- [147] D. Beaver, “Precomputing Oblivious Transfer,” in *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '95*, (London, UK, UK), pp. 97–109, Springer-Verlag, 1995.
- [148] M. Naor and B. Pinkas, “Efficient oblivious transfer protocols,” in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01*, (Philadelphia, PA, USA), pp. 448–457, Society for Industrial and Applied Mathematics, 2001.
- [149] ECRYPT-II and Vampire, “eBACS: ECRYPT Benchmarking of Cryptographic Systems,” 2018. <http://http://bench.cr.yj.to/> (accessed on 2018-01-22).