# Reducing Garbled Circuit Size While Preserving Circuit Gate Privacy *

Yongge Wang[1] and Qutaibah m. Malluhi[2]

[1] Department of SIS, UNC Charlotte, USA
yonwang@uncc.edu
[2] Department of CS, Qatar University, Qatar
qmalluhi@qu.edu.qa

**Abstract.** Yao's garbled circuits have been extensively used in Secure Function Evaluations (SFE). Several improvements have been proposed to improve the efficiency of garbled circuits. Kolesnikov and Schneider (2008) proposed the free-XOR technique. Naor, Pinkas, and Sumner (1999) introduced garbled row-reduction technique GRR3 to reduce each garbled gate to three ciphertexts, Pinkas et al (2009) proposed GRR2, and Zahur, Rosulek, and Evans (2015) introduced a half-gates technique to design free-XOR compatible GRR2. The GRR2, half-gates, and free-XOR garbling schemes improve efficiency by leaking locations of XOR gates in the circuit. This kind of information leakage is acceptable for SFE protocols though it maybe be unacceptable for other protocols such as Private Function Evaluation (PFE). For example, these techniques could not be used to improve the efficiency of existing non-universal-circuit-based constant round PFE protocols. The first result of this paper is a Gate Privacy preserving Garbled Row Reduction technique GPGRR2 for designing garbled circuits with at most two ciphertexts for each garbled gate. Compared with state-of-the-art gate-privacy-preserving garbling scheme GRR3, the scheme GPGRR2 reduces the garbled circuit size by a factor of at least 33%. The second result is the design of a linear (over integers) garbling scheme to garble a single odd gate to one ciphertext. Zahur, Rosulek, and Evans (2015) proved that a linear garbling scheme garbles each odd gate to at least 2 ciphertexts. Our result shows that Zahur et al's result should be formulated more appropriately by restricting the "linear concept" explicitly to the field $\mathbb{F}_{2^t}$. The third result is to use the GPGRR2 scheme to reduce the number of ciphertexts in non-universal-circuit based PFE protocols by a factor of 25%.

## 1 Introduction

Yao [18] introduced the garbled circuit concept which allows computing a function $f$ on an input $x$ without leaking any information about the input $x$ or individual circuit gate functionality used for the computation of $f(x)$. Since then,

* The work reported in this paper is supported by Qatar Foundation Grant NPRP X-063-1-014

garbled circuit based protocols have been used in numerous places and it has become one of the fundamental components of secure multi-party computation (SMC), secure function evaluation (SFE), and private function evaluation (PFE) protocols. In a PFE protocol, one participant $P_1$ holds a circuit $C$ and a private input $x_1$ and every other participant $P_i$ ($i \geq 2$) holds a private input $x_i$. The PFE protocol's goal is that a subset (or all) of the participants learns the circuit output $C(x_1, \cdots, x_n)$ but nothing beyond this. In particular, the participant $P_i$ ($i \geq 2$) should not learn anything else except the size of $C$ and, optionally, the output. Note that a PFE protocol is different from standard SMC/SFE protocols where the circuit $C$ is publicly known to all participants in SMC/SFE protocols.

Bellare et al [3] provides a rigorous definition of circuit garbling schemes and analyzed garbling scheme security from aspects of privacy, obliviousness, and authenticity. Specifically, Bellare et al [3] pointed out that garbling schemes that are secure for $\Phi_{\texttt{circ}}$ (that is, it does not conceal the circuit) is sufficient for the design of SFE/SMC protocols. However, for a PFE protocol, one needs a garbling scheme that is secure for $\Phi_{\texttt{size}}$ (that is, it only leaks the circuit size). Though Yao's circuit garbling scheme is only secure for $\Phi_{\texttt{topo}}$ (that is, it only reveals the circuit topology) and not secure for $\Phi_{\texttt{size}}$, one can use universal circuit to convert a $\Phi_{\texttt{topo}}$-secure garbling scheme to a $\Phi_{\texttt{size}}$-secure garbling scheme (see, e.g., Bellare et al [3]).

We first review Yao's garbled circuit construction using Beaver, Micali, and Rogaway's point-permute (or called external index) technique [2]. Note that the external index technique makes it possible to design garbled circuits without using CPA-secure encryption schemes. Unless stated otherwise, throughout the paper we will use lower case letters $u$, $v$, $w$, $x$, $y$, $z$ etc. to denote wires within a circuit and use $b_u, b_v, b_w, b_x, b_y, b_z \in \{0,1\}$ as variables to denote the values on the wires $u$, $v$, $w$, $x$, $y$, $z$ respectively. For a given number $t$ that is dependent on the security parameter $\kappa$, the circuit owner assigns two random values $k_x^0, k_x^1 \in \{0,1\}^t$ to each wire $x$ corresponding to 0 and 1 values of the wire. The circuit owner chooses a secret random permutation $\pi_x$ over $\{0,1\}$ for each wire $x$. The garbled values for the wire $x$ consist of $k_x^0||\pi_x(0)$ and $k_x^1||\pi_x(1)$ where $\pi_x(b)$ is considered as an external index for $k_x^b$. It is easily observed that for any $b \in \{0,1\}$, we have $b = \pi_x(b) \oplus \pi_x(0)$. For a gate $z = g(x,y)$, the garbled gate $\tilde{g}$ consists of four ciphertexts that are ordered using the external index $\pi_x(b_x)||\pi_y(b_y)$. For example, if we assume that $\pi_x(0) = \pi_y(0) = 1$ and $\pi_x(1) = \pi_y(1) = 0$, then the garbled gate $\tilde{g}$ is described using the following four ciphertexts.

$$\begin{array}{l} \pi_x(1)||\pi_y(1) : (k_z^{g(1,1)}||\pi_z(g(1,1))) \oplus H_g(k_x^1 \circ k_y^1) \\ \pi_x(1)||\pi_y(0) : (k_z^{g(1,0)}||\pi_z(g(1,0))) \oplus H_g(k_x^1 \circ k_y^0) \\ \pi_x(0)||\pi_y(1) : (k_z^{g(0,1)}||\pi_z(g(0,1))) \oplus H_g(k_x^0 \circ k_y^1) \\ \pi_x(0)||\pi_y(0) : (k_z^{g(0,0)}||\pi_z(g(0,0))) \oplus H_g(k_x^0 \circ k_y^0) \end{array} \quad (1)$$

where $H_g$ is a gate $g$ specific pseudorandom function (e.g., a secure hash function or an encryption scheme) whose output length is $|k_z^b|+1$ and $\circ$ is an operator. For example, one may define $k_1 \circ k_2 = k_1||k_2$ or $k_1 \circ k_2 = k_1 \oplus k_2$ or $k_1 \circ k_2 = k_1 + k_2 \bmod 2^t$ etc.. For most applications, we take a pseudorandom function $H$ (e.g.,

a cryptographic hash function) and define $H_g(\cdot) = H(\texttt{gID}, \cdot)$ where gID is an identity string for the gate $g$. At the start of the protocol, the circuit owner provides the evaluator with a garbled version $\tilde{g}$ for each gate $g$ of the circuit. During the evaluation process, the circuit owner provides garbled input values to the evaluator and the evaluator evaluates the garbled circuits gate by gate. As an example, if the input is $(x, y) = (1, 0)$, then the circuit owner sends garbled values $k_x^1||\pi_x(1) = k_x^1||0$ and $k_y^0||\pi_y(0) = k_y^0||1$ to the evaluator. Since the external index bit value $\pi_x(1)||\pi_y(0) = 01$, the evaluator uses the corresponding second ciphertext to recover the garbled value $k_z^{g(1,0)}||\pi_z(g(1,0))$ for the output wire $z$, which corresponds to the output $g(1, 0)$.

Several efforts have been made to reduce the garbled circuit size. Kolesnikov and Schneider [10] observed that if there is a circuit-wide global offset value $\Delta \in \{0,1\}^t$ such that garbled values for each wire $x$ within the circuit satisfy the invariance property $k_x^1 = k_x^0 \oplus \Delta$, then the XOR gate could be garbled for free since we have $k_x^{b_x} \oplus k_y^{b_y} = k_x^0 \oplus k_y^0 \oplus ((b_x \oplus b_y) \cdot \Delta)$ where $1 \cdot \Delta = \Delta$ and $0 \cdot \Delta = 0^t$.

Naor, Pinkas, and Sumner [13] observed that one can choose a randomly fixed pair $(b_x, b_y) \in \{0,1\}^2$ and let

$$k_z^{g(b_x,b_y)}||\pi(g(b_x, b_y)) = H_g(k_x^{b_x} \circ k_y^{b_y}).$$

Then the corresponding ciphertext for the row $(\pi(b_x), \pi(b_y))$ is a zero string and one does not need to store it. In other words, one can reduce the number of ciphertexts from 4 to 3 for each garbled gate. In this paper, we will refer this approach as GRR3.

Pinkas et al [15] used polynomial interpolation to reduce each gate to two ciphertexts. However, Pinkas et al's technique is not compatible with the free-XOR technique. Recently, Zahur, Rosulek, and Evans [19] introduced the state-of-the-art half-gates technique to design free-XOR compatible garbling schemes so that each AND/OR gate could be represented using two ciphertexts.

The aforementioned free-XOR, GRR2, and half-gates garbling schemes reduce garbled circuit sizes by leaking the number and locations of XOR gates within circuits. This kind of side information leakage is acceptable for SFE (secure function evaluation) though it may be unacceptable for other applications. For example, these techniques cannot be used to improve the efficiency of non-universal circuit based PFE protocols in Katz and Malka [7] and Mohassel and Sadeghian [12]. In this paper, we investigate the possibility of reducing garbled circuit size without leaking any further information beyond circuit topology. Specifically, we design garbled circuits with at most two ciphertexts for each garbled gate such that the only leaked information is the circuit topology. We then apply our techniques to PFE protocols in Katz and Malka [7] and Mohassel and Sadeghian [12] to reduce the number of ciphertexts by a factor of 25%.

It has been an interesting and challenging question to study the lower bounds of garbled circuit sizes. Zahur, Rosulek, and Evans [19] proved that any "linear" garbling scheme garbles an AND gate to at least two ciphertexts. However, the statement of their lower bound theorem is inaccurate. In this paper, we

present a linear (over integers) garbling scheme that garbles an AND gate to one ciphertext. By examining the proofs in [19], it is clear that their proof is based on linear operations in the finite field $\mathbb{F}_{2^t}$. Thus one should bear in mind that the result in [19] only applies to the finite field $\mathbb{F}_{2^t}$.

We conclude this section with the introduction of some notations. We use $\kappa$ to denote the security parameter, $p(\cdot)$ to denote a function $p$ that takes one input, and $p(\cdot,\cdot)$ to denote a function $p$ that takes two inputs. A function $f$ is said to be negligible in an input parameter $\kappa$ if for all $d > 0$, there exists $K$ such that for all $\kappa > K$, $f(\kappa) < \kappa^{-d}$. For convenience, we write $f(\kappa) = \text{negl}(\kappa)$. Two ensembles, $X = \{X_\kappa\}_{\kappa\in N}$ and $Y = \{Y_\kappa\}_{\kappa\in N}$, are said to be computationally indistinguishable (written as $X \overset{c}{\sim} Y$ or $X \overset{c}{=} Y$) if for all probabilistic polynomial-time algorithm $D$, we have

$$|Pr[D(X_\kappa, 1^\kappa) = 1] - Pr[D(Y_\kappa, 1^\kappa) = 1]| = \text{negl}(\kappa).$$

Throughout the paper, we use probabilistic experiments and denote their outputs using random variables. For example, $\text{Exp}^{\text{real}}_{E,A}(1^\kappa)$ represents the output of the real experiment for scheme $E$ with adversary $A$ on security parameter $\kappa$.

The structure of this paper is as follows. Section 2 reviews security definition for garbling schemes. Section 3 reviews GRR2 techniques. Section 4 presents our linear interpolation based garbled circuit construction techniques where each garbled gate uses two ciphertexts. Section 5 provides an optimization of GP-GRR2 and shows that one can linearly garble an AND gate with one ciphertext. Section 6 uses linear interpolation garbling schemes to reduce the size of garbled circuits for PFE protocols in various adversary security models. Section 7 presents a revised circuit garbling scheme GRRcirc that is only secure for input privacy (it reveals the number and positions of XOR gates).

## 2 Circuit garbling schemes and their security

In this section, we briefly review the formal definition of circuit garbling schemes formalized by Bellare, Hoang, and Rogaway [3].

**Definition 1.** *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in N}$ be a family of circuits such that $C_n$ is a set of boolean circuits that take $n$-bit inputs. A garbling scheme for $\mathcal{C}$ is a tuple of probabilistic polynomial time algorithms $\mathtt{GS} = (\mathtt{Gb}, \mathtt{Enc}, \mathtt{Eval}, \mathtt{Dec})$ with the following properties*

- $(\tilde{C}, \mathtt{sk}, \mathtt{dk}) = \mathtt{GS.Gb}(1^\kappa, C)$ *outputs a garbled circuit $\tilde{C}$, a secret key $\mathtt{sk}$, and a decoding key $\mathtt{dk}$ for circuits $C \in \mathcal{C}_n$ on the security parameter input $\kappa$.*
- $c = \mathtt{GS.Enc}(\mathtt{sk}, x)$ *outputs an encoding $c$ for an input $x \in \{0,1\}^*$.*
- $\tilde{y} = \mathtt{GS.Eval}(\tilde{C}, c)$ *outputs a garbled value $\tilde{y}$.*
- $y = \mathtt{GS.Dec}(\mathtt{dk}, \tilde{y})$ *outputs a circuit output.*

*The garbling scheme* $\mathtt{GS}$ *is* correct *if we have*

$$Pr[\mathtt{GS.Dec}(\mathtt{dk}, \mathtt{GS.Eval}(\tilde{C}, \mathtt{GS.Enc}(\mathtt{sk}, x))) \neq C(x)|\mathtt{GS}] = \text{negl}(\kappa).$$

*The garbling scheme* $\mathtt{GS}$ *is* efficient *if the size of* $\tilde{C}$ *is bounded by a polynomial and the run-time of* $c = \mathtt{GS.Enc}(\mathtt{sk}, x)$ *is also bounded by a polynomial.*

The security of garbling schemes is defined in terms of input and circuit privacy in the literature. For a garbled circuit, some side-information such as the number of inputs, outputs, gates, and the topology of the circuit $C$ (that is, the connection of gates but not gate types) and other information is leaked inherently. We denote such kind of side information as $\Phi(C)$. Thus a security definition of garbling schemes should capture the intuition that the adversary learns no information except $\Phi(C)$ and the output given one evaluation of the garbled circuit. The following definition requires that for any circuit or input chosen by the adversary, one can simulate the garbled circuit and the encoding based on the computation result and $\Phi(C)$ in polynomial time. In the definition, the variable $\alpha$ represents any state that the adversary $A$ may want to give to the algorithm $D$.

**Definition 2.** *(Privacy for garbling schemes) A garbling scheme* $\mathtt{GS}$ *for a family of circuits* $\mathcal{C}$ *is said to be* input and circuit private *if there exists a probabilistic polynomial time simulator* $\mathtt{Sim}_{\mathtt{GS}}$ *such that for all probabilistic polynomial time adversaries* $A$ *and algorithms* $D$ *and all large* $\kappa$*, we have*

$$\left| \begin{array}{r} Pr[D(\alpha, x, C, \tilde{C}, c) = 1 | \mathtt{REAL}] - \\ Pr[D(\alpha, x, C, \tilde{C}_{\mathtt{sim}}, \tilde{c}) = 1 | \mathtt{SIM}] \end{array} \right| = \mathrm{negl}(\kappa)$$

*where* $\mathtt{REAL}$ *is the following event*

$$\begin{aligned} (x, C, \alpha) &= A(1^\kappa); \\ (\tilde{C}, \mathtt{sk}, \mathtt{dk}) &= \mathtt{GS.Gb}(1^\kappa, C); \\ c &= \mathtt{GS.Enc}(\mathtt{sk}, x) \\ C(x) &= \mathtt{GS.Dec}(\mathtt{dk}, \mathtt{GS.Eval}(\tilde{C}, c)) \end{aligned}$$

*and* $\mathtt{SIM}$ *is the following event*

$$\begin{aligned} &(x, C, \alpha) = A(1^\kappa); \\ &(\tilde{C}_{\mathtt{sim}}, \tilde{c}) = \mathtt{Sim}_{\mathtt{GS}}(1^\kappa, C(x), \Phi(C), 1^{|C|}, 1^{|x|}). \end{aligned}$$

The authors of [3] considered the following three kinds of commonly used side-information functions.

1. $\Phi_{\mathtt{size}}(C) = (n, m, q)$ where $n, m, q$ are the number of inputs, outputs, and gates of the circuit $C$ respectively.
2. $\Phi_{\mathtt{topo}}(C) = C_{\mathtt{topo}}$ where a topolgical circuit $C_{\mathtt{topo}}$ is like the conventional circuit $C$ except that the functionality of the gates is unspecified.
3. $\Phi_{\mathtt{circ}}(C) = C$ where the side information is the circuit itself. That is, the entire circuit $C$ is revealed.

The authors of [3] then argued that, by varying $\Phi(C)$, one can encompass the customary setting for secure function evaluation (SFE) with $\Phi(C) = \Phi_{\mathtt{circ}}(C)$

and private function evaluation (PFE) with $\Phi(C) = \Phi_{\mathtt{size}}(C)$. It is straightforward to observe that garbling schemes based on free-XOR, GGR2, and half-gates are secure for $\Phi_{\mathtt{circ}}$ but not secure for $\Phi_{\mathtt{topo}}$.

It is pointed out in Bellare et al [3, Sections 3.8] that, for both indistinguishability-based security notion and simulation-based security notion, each $\Phi_{\mathtt{topo}}$-secure garbling scheme $\mathtt{GS}_{\mathtt{topo}}$ can be converted to a $\Phi_{\mathtt{size}}$-secure garbling scheme $\mathtt{GS}_{\mathtt{size}}$ using universal circuits. $\mathtt{GS}_{\mathtt{size}}$ and oblivious transfers can then be used to design secure PFE protocols. If the security notion is based on simulation, then one can use $\Phi_{\mathtt{circ}}$-secure garbling schemes, universal circuits, and oblivious transfers to design secure PFE protocols. However, no proof has been presented to show whether one can use $\Phi_{\mathtt{circ}}$-secure garbling schemes, universal circuits, and oblivious transfers to design secure PFE protocols using the indistinguishability-based security notion[3].

We conclude this section by pointing out a circuit complexity result which shows the important information leakage by identifying the number (or locations) of XOR gates within a topological circuit $C_{\mathtt{topo}}$. Let $\mathtt{AC}^i$ denote the family of polynomial size circuits of depth $O(\log^i n)$ with unlimited-fanin AND and OR gates (NOT gates are only allowed at inputs). Let $\mathtt{NC}^i$ denote the family of polynomial size circuits of depth $O(\log^i n)$ with bounded-fanin AND and OR gates. It is a folklore that

$$\mathtt{NC}^i \subseteq \mathtt{AC}^i \subseteq \mathtt{NC}^{i+1}.$$

Let $f_{\mathtt{parity}}(x_1, \cdots, x_n) = x_1 \oplus x_1 \oplus \cdots \oplus x_n$ be the parity function. It is well known that $f_{\mathtt{parity}} \in \mathtt{NC}^1$. Ajtai et al [1] and Furst et al [5] showed that $f_{\mathtt{parity}} \notin \mathtt{AC}^0$. That is, $f_{\mathtt{parity}} \in \mathtt{NC}^1 \setminus \mathtt{AC}^0$.

Let $C_{\mathtt{parity}}$ be a randomly selected polynomial size circuit of constant depth with unlimited-fanin XOR gates that computes $f_{\mathtt{parity}}$. Note that there are many such kind of circuits. Let $f_{\mathtt{ac}_0} \in \mathtt{AC}^0$ and $C_{f_{\mathtt{ac}_0}}$ be a circuit that computes $f_{\mathtt{ac}_0}$ such that $C_{f_{\mathtt{ac}_0},\mathtt{topo}} = C_{\mathtt{parity},\mathtt{topo}}$. That is, the topological circuits of $C_{f_{\mathtt{ac}_0}}$ and $C_{\mathtt{parity}}$ are identical.

Assume that a circuit owner randomly selects $C_{\mathtt{parity}}$ or $C_{f_{\mathtt{ac}_0}}$ to garble. Given $\Phi_{\mathtt{topo}}$-secure garbled circuits for $C_{\mathtt{parity}}$ or $C_{f_{\mathtt{ac}_0}}$, the evaluator cannot tell whether the evaluated function is in $\mathtt{AC}^0$ or not. On the other hand, if the evaluator receives $\Phi_{\mathtt{circ}}$-secure garbled circuits for $C_{\mathtt{parity}}$ or $C_{f_{\mathtt{ac}_0}}$, it can distinguish whether the evaluated function is in $\mathtt{AC}^0$. More involved examples that contain both odd and even gates but with a large fraction of even gates could be constructed using Håstad's switching lemma [6]. The aforementioned example shows that it is preferred to design efficient garbling schemes that do not reveal the number and positions of XOR gates.

## 3 Pinkas et al's Garbled Row Reduction GRR2

We first review Pinkas et al's Garbled Row Reduction GRR2 [15]. Let $t$ be the length in terms of number of bits of wire lables. That is, we have $t = |k_x^b|$ for all

[3] The authors would like to thank Dr. Viet Tung Hoang for several valuable discussions on this question and other results in [3].

wires $x$ and $b = 0, 1$. Wire labels $k_x^b$ and integers $0, 1, 2, 3, \cdots$ can be interpreted as elements of the finite field $\mathbb{F}_{2^t}$. A binary gate is said to be odd if its truth table has an odd number of '1' entries (e.g. an AND or OR gate), otherwise it is called an even gate (e.g., an XOR gate). Using polynomial interpolation, Pinkas et al showed that each gate could be represented by only two ciphertexts. Specifically, for an odd gate $g$ (e.g., an AND or OR gate), assume that the first three ciphertexts $C_1, C_2, C_3$ encrypt the same wire label $k_z^b||\pi_z(b)$ via $C_i = (k_z^b||\pi_z(b)) \oplus (K_i||M_i)$ (for $i = 1, 2, 3$) and the fourth ciphertext $C_4$ encrypts the wire label $k_z^{1-b}||\pi_z(1-b)$ via $C_4 = (k_z^{1-b}||\pi_z(1-b)) \oplus (K_4||M_4)$ where $b, M_i \in \{0, 1\}$ for $i = 1, 2, 3, 4$. Let $P(X)$ be a degree two polynomial over $\mathbb{F}_{2^t}$ passing through points $(1, K_1), (2, K_2)$, and $(3, K_3)$. Let $Q(X)$ be another degree two polynomial over $\mathbb{F}_{2^t}$ passing through points $(5, P(5)), (6, P(6))$, and $(4, K_4)$. Then by setting $k_z^b = P(0)$ and $k_z^{1-b} = Q(0)$, one can replace the garbled table with $\langle P(5), P(6), c_1, c_2, c_3, c_4\rangle$ where $P(5)$ and $P(6)$ are elements from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_z(g(b_x, b_y)) = c_i \oplus M_i$ for $i = 2\pi_x(b_x) + \pi_y(b_y) + 1$. The total size of the garbled gate is $2t+4$ bits. Interpolating the polynomial passing through points $(5; P(5))$, $(6; P(6))$, and $(i; K_i)$ for $i = 1, 2, 3, 4$ will produce either polynomial $P(X)$ or $Q(X)$, which can be evaluated at $X = 0$ to get the appropriate value $k_z^b$ or $k_z^{1-b}$.

For an even gate $g$ (e.g., an XOR or NXOR gate), assume that ciphertexts $C_{i_1}, C_{i_2}$ encrypt the wire label $k_z^0||\pi_z(0)$ via $C_{i_j} = (k_z^0||\pi_z(0)) \oplus (K_{i_j}||M_{i_j})$ (for $j = 1, 2$) and the ciphertexts $C_{i_3}, C_{i_4}$ encrypt the wire label $k_z^1||\pi_z(1)$ via $C_{i_j} = (k_z^1||\pi_z(1)) \oplus (K_{i_j}||M_{i_j})$ (for $j = 3, 4$) where $M_{i_j} \in \{0, 1\}$ for $i_j = 1, 2, 3, 4$. Let $P(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through points $(i_1, K_{i_1})$ and $(i_2, K_{i_2})$. Let $Q(X)$ be another linear polynomial over $\mathbb{F}_{2^t}$ passing through points $(i_3, K_{i_3})$ and $(i_4, K_{i_4})$. Define $k_z^0 = P(0)$ and $k_z^1 = Q(0)$. If $\pi_z(0) = 0$ then the garbled gate is represented as $\langle P(5), Q(5), c_1, c_2, c_3, c_4\rangle$. Otherwise, $\pi_z(1) = 0$ and the garbled gate is represented as $\langle Q(5), P(5), c_1, c_2, c_3, c_4\rangle$. In the garbled gate, $P(5)$ and $Q(5)$ are elements from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_z(g(b_x, b_y)) = c_i \oplus M_i$ for $i = 2\pi_x(b_x) + \pi_y(b_y) + 1$. The total size of the garbled gate is $2t + 4$ bits. The evaluator receives the garbled gate in the format of $\langle Y_1, Y_2, c_1, c_2, c_3, c_4\rangle$. At the time of evaluation, the evaluator first calculates the value $\pi_z(b_z)$ using the ingoing wire labels. If $\pi_z(b_z) = 0$, then it interpolates the linear polynomial passing through points $(5; Y_1)$ and $(i; K_i)$ for $i = 1, 2, 3, 4$ which will produce either polynomial $P(X)$ or $Q(X)$, which can be evaluated at $X = 0$ to get the appropriate value $k_z^{b_z}$. If $\pi_z(b_z) = 1$, then it interpolates the linear polynomial passing through points $(5; Y_2)$ and $(i; K_i)$ for $i = 1, 2, 3, 4$ to obtain the appropriate value $k_z^{b_z}$.

The ciphertexts for odd gates and even gates have different format with GRR2 techniques. Thus GGR2 garbled circuits are not secure for $\Phi_{\texttt{topo}}$. It is also easy to check that in the GRR2 garbling scheme, the number of ciphertexts for an even gate could be reduced to one ciphertext by using the GRR3 approach.

## 4 Garbled gate size reduction using linear interpolation

### 4.1 Gate Privacy preserving Garbled Row Reduction GPGRR2

As we mentioned in the preceding section, Pinkas et al's GRR2 garbling scheme [15] leaks the number and positions of even/odd gate types. For example, an evaluator evaluates a garbled odd gate using degree two polynomial interpolation and evaluates a garbled even gate using linear interpolation. The free-XOR techniques proposed by Kolesnikov and Schneider [10] leaks the number and positions of XOR gates and the half-gates techniques by Zahur, Rosulek, and Evans [19] leaks the number and positions of XOR gates also. In this section, we propose a gate privacy preserving garbled row reduction GPGRR2 technique to garble circuits with security for $\Phi_{\texttt{topo}}$. Our garbling scheme GPGRR2 does not require the external index bits. For reason of convenience, the following construction still includes the external index bits. These external index bits are used in next Sections for the design of garbling schemes that are only secure for $\Phi_{\texttt{circ}}$.

First select two parameters $t$ and $\tau$ based on the security requirements. It is recommended to select $\tau$ such that $10 \leq \tau < t$. Each ciphertext will be of length $t+1$ bits. In order to garble a circuit $C$, the circuit owner first chooses a circuit-wide global offset value $\Delta \in \{0,1\}^t$ uniformly at random. Furthermore, let $H$ be a pseudo-random function with $(t+\tau+1)$-bits output. The circuit $C$ will be garbled in such a way that for all wires $x$, the garbled values $k_x^0||\pi_x(0)$ and $k_x^1||\pi_x(1)$ for the wire $x$ satisfy the following invariance property:

$$k_x^1 = k_x^0 + \Delta \mod 2^t \tag{2}$$

In the following, we formally describe the process of garbling a gate $z = g(x,y)$ in a circuit $C$. Let $k_x^0||\pi_x(0)$, $k_x^1||\pi_x(1)$, $k_y^0||\pi_y(0)$, and $k_y^1||\pi_y(1)$ be the garbled input wire values for the wires $x$ and $y$ respectively. Let $k_z^0||\pi_z(0)$, $k_z^1||\pi_z(1)$ be the garbled output wire values for the output wire $z = g(x,y)$ that will be defined. Define the operator $\circ$ as the integer addition modulo $2^t$. Then we have

$$\begin{array}{l} k_x^0 \circ k_y^0 = k_x^0 + k_y^0 = \bar{x}_1 \mod 2^t \\ k_x^0 \circ k_y^1 = k_x^1 \circ k_y^0 = k_x^0 + k_y^0 + \Delta = \bar{x}_1 + \Delta \mod 2^t \\ k_x^1 \circ k_y^1 = k_x^0 + k_y^0 + 2\Delta = \bar{x}_1 + 2\Delta \mod 2^t \end{array} \tag{3}$$

for some $\bar{x}_1 \in \{0,1\}^t$. For these garbled input wire values, we have

$$\begin{array}{l} K_{00}||M_{00}||N_{00} = H_g(k_x^0 \circ k_y^0) = H_g(\bar{x}_1 \mod 2^t) \\ K_{01}||M_{01}||N_{01} = H_g(k_x^0 \circ k_y^1) = H_g(\bar{x}_1 + \Delta \mod 2^t) \\ K_{10}||M_{10}||N_{10} = H_g(k_x^1 \circ k_y^0) = H_g(\bar{x}_1 + \Delta \mod 2^t) \\ K_{11}||M_{11}||N_{11} = H_g(k_x^1 \circ k_y^1) = H_g(\bar{x}_1 + 2\Delta \mod 2^t) \end{array} \tag{4}$$

where $M_{00}, M_{01}, M_{10}, M_{11} \in \{0,1\}$ and $N_{00}, N_{01}, N_{10}, N_{11} \in \{0,1\}^\tau$. It follows that

$$K_{01}||M_{01}||N_{01} = K_{10}||M_{10}||N_{10}.$$

In case that there exist two values in

$$N_{00},\ N_{10},\ \text{and}\ \ N_{11} \tag{5}$$

that are identical, re-start the garbling process and choose different garbled input wire values for the wires $x$ and $y$. We distinguish the following two cases depending on whether $g$ is an even gate or an odd gate.

**Garbling an odd gate** $g$. First we assume that $g$ is an OR gate. Let $P(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the following two points

$$(N_{10}, K_{10})\ \text{and}\ (N_{11}, K_{11})\,.$$

Set $k_z^1 = P(0)$ and $k_z^0 = k_z^1 - \Delta \mod 2^t$ where we interpret $k_z^0$, $k_z^1$, and $\Delta$ as integers modulo $2^t$. Let $Q(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the following two points

$$(0, k_z^0)\ \text{and}\ (N_{00}, K_{00})\,.$$

Notice that $P(X)$ is interpolated using the points corresponding to the situation when the output of the OR gate is 1 and $Q(X)$ is interpolated using points corresponding to the situation when the output of the OR gate is 0. Let $X_z$ be a solution of the equation $P(X) = Q(X)$ over $\mathbb{F}_{2^t}$. Then the garbled table for the gate $g$ is $\langle X_z, P(X_z), c_1, c_2, c_3, c_4\rangle = \langle X_z, Q(X_z), c_1, c_2, c_3, c_4\rangle$ where $X_z$ and $P(X_z)$ are elements from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_z(g(b_x, b_y)) = c_i \oplus M_i$ for $i = 2\pi_x(b_x) + \pi_y(b_y) + 1$. The total size of the garbled gate is $2t + 4$ bits. Without the external index bits, the total size of the garbled gate is $2t$ bits.

For an AND gate $g$, one chooses the linear polynomial $P(X)$ to pass through the points $(N_{10}, K_{10})$ and $(N_{00}, K_{00})$. Let $k_z^0 = P(0)$ and $k_z^1 = k_z^0 + \Delta \mod 2^t$. Then choose the linear polynomial $Q(X)$ to pass the points $(0, k_z^1)$ and $(N_{11}, K_{11})$. That is, $P(X)$ is interpolated using the points corresponding to the situation when the output of the AND gate is 0 and $Q(X)$ is interpolated using points corresponding to the situation when the output of the AND gate is 1. The other steps remain the same.

It should be noted that, alternatively, one can use $\langle P(2^\tau), Q(2^\tau), c_1, c_2, c_3, c_4\rangle$ as the garbled table for the gate $g$ instead of using the solution point $X_z$ for the equation $P(X) = Q(X)$. In this case, the external index bit will be used to determine whether $P(2^\tau)$ or $Q(2^\tau)$ should be used for the linear interpolation at the evaluation time.

**Garbling an even gate** $g$. Without loss of generality, assume that $g$ is an XOR gate. The case for an NXOR gate is dealt with similarly by swapping the values of $k_z^0$ and $k_z^1$. Let $P(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the following two points

$$(N_{00}, K_{00})\ \text{and}\ (N_{11}, K_{11})\,.$$

Set $k_z^0 = P(0)$ and $k_z^1 = k_z^0 + \Delta \mod 2^t$ where we interpret $k_z^0$, $k_z^1$, and $\Delta$ as integers modulo $2^t$. Let $Q(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the following two points

$$(0, k_z^1)\ \text{and}\ (N_{10}, K_{10})\,.$$

In other words, $P(X)$ is interpolated using the points corresponding to the situation when the output of the XOR gate is 0 and $Q(X)$ is interpolated using points corresponding to the situation when the output of the XOR gate is 1. Let $X_z$, be a solution of the equation $P(X) = Q(X)$ over $\mathbb{F}_{2^t}$. Then the garbled table for the gate $g$ is $\langle X_z, P(X_z), c_1, c_2, c_3, c_4\rangle$ where $X_z$ and $P(X_z)$ are elements from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_z(g(b_x, b_y)) = c_i \oplus M_i$ for $i = 2\pi_x(b_x) + \pi_y(b_y) + 1$. The total size of the garbled gate is $2t + 4$ bits. Without the external index bits, the total size of the garbled gate is $2t$ bits. Similarly, we may also use $\langle P(2^\tau), Q(2^\tau), c_1, c_2, c_3, c_4\rangle$ as the garbled table for the gate $g$ and use the external index bit information for determining whether $P(2^\tau)$ or $Q(2^\tau)$ should be used for the linear interpolation at the evaluation time.

**Evaluation of a garbled circuit.** For a garbled gate $\tilde{g} = \langle X_z, P(X_z), c_1, c_2, c_3, c_4\rangle$ where the evaluator does not know whether $\tilde{g}$ is an even gate or an odd gate, the evaluator receives garbled values $k_x^{b_x}||\pi(b_x)$ and $k_y^{b_y}||\pi(b_y)$ on the wires $x$ and $y$ respectively. The evaluator first computes

$$K||M||N = H_g(k_x^{b_x} + k_y^{b_y} \mod 2^t).$$

Let $R(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through points $(N, K)$ and $(X_z, P(X_z))$. Then $k_z^{g(b_x,b_y)} = R(0)$. The external index bit for output wire is calculated using $\pi_z(g(b_x, b_y)) = c_{2\pi_x(b_x)+\pi_y(b_y)+1} \oplus M$.

In the above evaluation process, the evaluator needs to carry out one cryptographic hash function operation and one linear polynomial interpolation operation. For the linear polynomial interpolation, the evaluator needs to find $a, b \in \mathbb{F}_{2^t}$ such that $aN_i + b = K_i$ and $aX_z + b = P(X_z)$. That is, $a = (N_i - X_z)^{-1}(K_i - P(X_z))$ over $\mathbb{F}_{2^t}$. In a summary, the major cost for the evaluator is one cryptographic hash function operation and one field element inverse operation over $\mathbb{F}_{2^t}$.

In the above garbling scheme, an additional parameter $\tau$ is used. For larger circuits, one should choose a larger $\tau$ though for smaller circuits, one can use a smaller $\tau$. The value of $\tau$ does not have impact on the garbling scheme security. However, it has impact on the efficiency of the garbling process. If the value of $\tau$ is too small, then the probability for two values in (5) to be identical is high and one has to re-start the garbling process more frequently. On the other hand, for large enough $\tau$, the probability for two values in (5) to be identical is very small and one does not need to restart the garbling process at all. It is also noted that the value of $\tau$ has no impact on the garbled circuit size.

### 4.2 Provable security of GPGRR2 for $\Phi_{\texttt{topo}}$

In Section 4.1, we proposed a Gate Privacy preserving Garbled Row Reduction technique GPGRR2 such that each garbled gate contains two ciphertexts and a four-bits ciphertext. The four-bits ciphertext is optional and could be ignored since we do not use it for the scheme GPGRR2. For both odd gates and even gates, the two ciphertexts are the coordinates of a point in a two dimensional

space over $\mathbb{F}_{2^t}$. Thus the evaluator cannot distinguish the type of a garbled gate. The remaining part of the security proof is similar to that of the garbling scheme `Garble1` security for $\Phi_{\texttt{topo}}$ by Bellare, Hoang, and Rogaway [3]. The proof of `Garble1` security in [3] is based on the observation that, given a pair of garbled values of the input wires, the evaluator can compute one garbled output value, but cannot distinguish the other garbled output value from random. This is true for GPGRR2 since the other garbled value is defined using a linear interpolation with a value which is unknown to the evaluator (indeed, the evaluator cannot distinguish that unknown value from random). The details are omitted here.

## 5 Optimized GPGRR2 and lower bounds for garbled AND gate ciphertexts

GPGRR2 is secure for $\Phi_{\texttt{topo}}$ and has comparable efficiency with other GRR techniques that are only secure for $\Phi_{\texttt{circ}}$. For example, Pinkas et al's Garbled Row Reduction GRR2 [15] converts each odd gate to two ciphertexts and each even gate to one ciphertext. Pinkas et al's GRR2 technique requires the evaluator to carry out a degree two polynomial interpolation while GPGRR2 only requires a linear interpolation.

Zahur, Rosulek, and Evans [19] proved that "every ideally secure linear garbling scheme for AND gates must have two ciphertexts for each garbled gate". Zahur, Rosulek, and Evans's proof is based on linear operations in the finite field $\mathbb{F}_{2^t}$. In this section, we show that if we use linear operations over integers (instead of linear operations over $\mathbb{F}_{2^t}$), we can design a secure linear garbling scheme that garbles an AND/OR gate to only one ciphertext. This technique is further used to optimize the garbling scheme GPGRR2. In an ideal case, the optimized garbling scheme GPGRR2 may generate garbled circuits of $1.5n$ ciphertexts for circuits of $n$ gates. That is, the garbled circuit is around $1.5nt$ bits. But the reader should be reminded that generally this ideal size is not achievable. Indeed, the problem of finding an optimized garbled circuit for a given circuit is **NP**-complete following a similar proof as that in FleXOR [9].

The garbling scheme GPGRR2 in Section 4.1 used a circuit-wide global offset value $\Delta$ though it is not necessary to have this offset value $\Delta$ to be global. In order for the construction in Section 4.1 to work, it suffices to have the following invariance property

$$k_x^0 + k_y^1 = k_x^1 + k_y^0 \mod 2^t \tag{6}$$

for all gates $z = g(x, y)$ with garbled input wire values $k_x^0||\pi_x(0)$, $k_x^1||\pi_x(1)$, $k_y^0||\pi_y(0)$, and $k_y^1||\pi_y(1)$ respectively. Based on this observation, the garbling scheme GPGRR2 could be optimized using the following principle: for each gate $g$ with two input wires $x$ and $y$, if $x$ is the output wire of a gate $g_1$ and $y$ is the output wire of a gate $g_2$, then we can construct a garbled gate for $g_1$ with one ciphertext and a garbled gate for $g_2$ with two ciphertexts. The gates $g_1$ and $g_2$ are constructed in such a way that the equation (6) is satisfied.

As an example of optimized garbling scheme GPGRR2, we construct a $\Phi_{\texttt{topo}}$-secure garbled circuit of 4-ciphertexts for the 3-gate circuit "$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$".

Let $g_1$ be the gate $x_5 = (x_1 \wedge x_2)$, $g_2$ be the gate $x_6 = (x_3 \wedge x_4)$, and $g_3$ be the gate $x_7 = (x_5 \vee x_6)$ respectively. Assume that the invariance property (6) is satisfied for garbled input wire labels for gates $g_1$ and $g_2$. That is, (6) is satisfied by replacing $x, y$ with $x_1, x_2$ (or with $x_3, x_4$) respectively. Similar to the original GPGRR2 garbling scheme, we define the operator $\circ$ as the integer addition modulo $2^t$.

**Garbling the gate** $g_1$: "$x_5 = (x_1 \wedge x_2)$". Let $k^0_{x_1}||\pi_{x_1}(0), k^1_{x_1}||\pi_{x_1}(1), k^0_{x_2}||\pi_{x_2}(0)$, and $k^1_{x_2}||\pi_{x_2}(1)$ be the garbled input wire values for the wires $x_1$ and $x_2$ respectively. For $i_1, i_2 \in \{0, 1\}$, let

$$K_{i_1 i_2}||M_{i_1 i_2}||N_{i_1 i_2} = H_{g_1}(k^{i_1}_{x_1} \circ k^{i_2}_{x_2}).$$

By the invariance property (6), we have

$$k^0_{x_1} \circ k^1_{x_2} = k^1_{x_1} \circ k^0_{x_2} \mod 2^t$$

This implies that $K_{01}||M_{01}||N_{01} = K_{10}||M_{10}||N_{10}$. In case that there are two values from $N_{00}$, $N_{01}$, and $N_{11}$ that are identical, re-start the garbling process to choose different garbled input wire values for the wires $x_1$ and $x_2$.

Let $P(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the two points $(N_{00}, K_{00})$ and $(N_{01}, K_{01})$. Let $Q(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the two points $(N_{11}, K_{11})$ and $(2^\tau, P(2^\tau))$. Set $k^0_{x_5} = P(0)$ and $k^1_{x_5} = Q(0)$. Then the garbled table for the gate $g_1$ is $\langle P(2^\tau), c_1, c_2, c_3, c_4\rangle$ where $P(2^\tau)$ is an element from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_{x_5}(g(b_{x_1}, b_{x_2})) = c_i \oplus M_i$ for $i = 2\pi_{x_1}(b_{x_1}) + \pi_{x_2}(b_{x_2}) + 1$. The total size of the garbled gate is $t + 4$ bits.

**Garbling the gate** $g_2$: "$x_6 = (x_3 \wedge x_4)$". Let $k^0_{x_3}||\pi_{x_3}(0), k^1_{x_3}||\pi_{x_3}(1), k^0_{x_4}||\pi_{x_4}(0)$, and $k^1_{x_4}||\pi_{x_4}(1)$ be the garbled input wire values for the wires $x_3$ and $x_4$ respectively. For $i_1, i_2 \in \{0, 1\}$, let

$$K_{i_1 i_2}||M_{i_1 i_2}||N_{i_1 i_2} = H_{g_1}(k^{i_1}_{x_3} \circ k^{i_2}_{x_4}).$$

By the invariance property (6), we have

$$k^0_{x_3} \circ k^1_{x_4} = k^1_{x_3} \circ k^0_{x_4} \mod 2^t$$

This implies that $K_{01}||M_{01}||N_{01} = K_{10}||M_{10}||N_{10}$. In case that there are two values from $N_{00}, N_{01},$ and $N_{11}$ that are identical, re-start the garbling process to choose different garbled input wire values for the wires $x_3$ and $x_4$.

Let $P(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the two points $(N_{00}, K_{00})$ and $(N_{01}, K_{01})$. Set $k^0_{x_6} = P(0)$ and

$$k^1_{x_6} = k^0_{x_6} + k^1_{x_5} - k^0_{x_5} \mod 2^t \tag{7}$$

where we interpret $k^0_{x_5}$, $k^1_{x_5}$, $k^0_{x_6}$, and $k^1_{x_6}$ as integers modulo $2^t$. Note that the equation (7) guarantees that the invariance (6) is satisfied for the gate $g_3$ with input wires $x_5, x_6$. Let $Q(X)$ be a linear polynomial over $\mathbb{F}_{2^t}$ passing through the two points $(0, k^1_{x_6})$ and $(N_{11}, K_{11})$. Let $X_z$, be a solution of the equation $P(X) =$

$Q(X)$ over $\mathbb{F}_{2^t}$. Then the garbled table for the gate $g$ is $\langle X_z, P(X_z), c_1, c_2, c_3, c_4\rangle$ where $X_z$ and $P(X_z)$ are elements from $\mathbb{F}_{2^t}$ and $c_1, c_2, c_3, c_4$ are bits encrypting the external index bits. That is, $\pi_{x_6}(g(b_{x_3}, b_{x_4})) = c_i \oplus M_i$ for $i = 2\pi_{x_2}(b_{x_2}) + \pi_{x_4}(b_{x_4}) + 1$. The total size of the garbled gate is $2t + 4$ bits.

**Garbling the gate** $g_3$: "$x_7 = (x_5 \vee x_6)$". By the equation (7), the invariance (6) is satisfied for the gate $g_3$ with input wires $x_5, x_6$. Thus the garbling process for the gate $g_1$ could be used to construct a garbled gate $\tilde{g_3}$ with one ciphertext and 4 bits. That is, the total size of the garbled gate $\tilde{g_3}$ is $t + 4$ bits. The details are omitted here.

As a summary, the garbled circuit for the 3-gate circuit "$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$" contains four ciphertexts (one for $g_1$, two for $g_2$, and one for $g_3$) and twelve bits. The total size of the garbled circuit is $4t + 12$ bits. Note that for such kind of 3-gate circuit, the best reported garbled circuit size in the literature is $6t + 12$ bits.

The proof of security for the optimized GPGRR2 remains the same as that for GPGRR2 and the details are omitted here.

## 6 Reducing ciphertext size in Private Function Evaluation (PFE) Protocols

In a two party PFE protocol, participant $P_1$ has a string $x$, participant $P_2$ has a function $f$ and the outcome of the protocol is that $P_2$ learns $f(x)$ and nothing about $x$ (beyond its length), while $P_1$ learns nothing about $f$ (beyond side information we are willing to leak, such as the number of gates in the circuit $f$). Similarly, the outcome of the two party PFE protocol could be that $P_1$ learns $f(x)$ and nothing about $f$, while $P_2$ learns nothing about $x$. For the general case that $P_2$ has a private input $x_2$ himself, one can include the value of $x_2$ in the circuit computing $f$ itself.

Traditionally, there are two approaches to design PFE protocols: using universal circuits and using homomorphic encryption. Universal circuit based PFE protocols introduce extra overhead and result in more complicated implementations. For the class of size $n$ circuits, Valiant's universal circuit [17] is of size $19n \log n$ with depth $O(n)$ and Kolesnikov and Schneider's universal circuit [11] is of size $1.5n \log^2 n$ though it has smaller universal circuits for circuit sizes less than 5000. Kiss and Schneider [8] further reduced Valiant's bound by constructing universal circuit where the number of AND gates is bounded by $5n \log n$ and where the number of total gates is bounded by $20n \log n$. Though Kiss and Schneider [8] showed that it is practical to implement PFE using Valiant's size-optimized universal circuits, they claimed that "*universal circuits are not the most efficient solution to perform PFE*". Specifically, SFE protocol implementation for functions with billions of gates has been reported in the literature though the best reported universal circuit based PFE protocol implementation

[8] is for simulated circuits of 300,000 gates, which results in a universal circuit of at most $245,627,140$ gates (and at most 61,406,785 AND gates)[4].

### 6.1 PFE in semi-honest security model

Katz and Malka [7] and Mohassel and Sadeghian [12] proposed efficient constant-round Yao's garbled circuit based PFE protocols with communication/computational complexity linear in the size of the circuit computing $f$. The PFE protocols in [7] and [12] require that each circuit gate contain four ciphertexts. In the following, we use our GPGGR2 techniques to reduce the number of each garbled gate's ciphertexts to three in these PFE protocols. Thus we have a 25% reduction in the garbled circuit size for these PFE protocols. Note that free-XOR, GGR2, and half-gates could not be used to reduce the ciphertext numbers in these PFE protocols. The garble row reduction technique GGR3 cannot be used to reduce the ciphertext numbers in these PFE protocols either since the wire label values are obliviously chosen by both parties.

Katz and Malka [7] introduced one PFE protocol with provable security in the semi-honest security model with the assumption of semantic security for homomorphic encryption schemes and linear-related key security for symmetric encryption schemes. They also introduced a more efficient variant PFE protocol with provable security in the random oracle model. The second protocol is roughly twice as efficient as the first one. The authors of [7] mentioned that the random oracle requirement for the second protocol may not be necessary and its security without random oracle may be proved if further assumptions on the symmetric-key encryption scheme is made. In the following, we reduce the number of ciphertexts in the second PFE protocol [7] (with security in random oracle) by a factor of 25%. The same reduction could be made for the PFE protocols in Mohassel and Sadeghian [12].

PFE protocols in [7, 12] use a singly homomorphic public-key encryption scheme $\mathtt{sHE}(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ such as the additive homomorphic Paillier encryption scheme [14]. In the following, we will give the protocol description in sufficient details without a formal definition. For a formal definition, the readers are referred to [7]. In our discussion, we assume that $P_2$ learns the output $f(x)$. The protocol can be modified to let $P_1$ learn the output straightforwardly. Let $C_f$ be a circuit that computes $P_2$'s function $f$ and that $C_f$ contains only NAND gates. Assume that $C_f$ have $n$ gates and it take $l$-bit inputs. In a high level, the PFE protocol proceeds as follows.

1. Given the pair $(n, l)$, $P_1$ generates a sequence of $n$ gates.
2. $P_2$ obliviously connects these gates to form a circuit $C_f$ using a singly homomorphic encryption scheme.
3. $P_1$ produces a garbled circuit corresponding to the circuit $C_f$ by garbling the $n$ gates independently (which are connected obliviously).

[4] The authors would like to thank Agnes Kiss for pointing out an inaccurate calculation here in an early version of this paper.

4. $P_1$ gives an encoded version of the input $x$ to $P_2$ and $P_2$ evaluates the garbled circuit to obtain the circuit output $C_f(x) = f(x)$.

In the following, we describe an instantiation of the above PFE protocol with reduced number of ciphertexts for each garbled gate. Let the outgoing wires set $\mathtt{OW} = \{\mathtt{ow}_1, \cdots, \mathtt{ow}_l, \cdots, \mathtt{ow}_{l+n}\}$ be the union of the set of $l$ input wires and the $n$ output wires for all gates in the circuit $C_f$. Let the incoming wires set $\mathtt{IW} = \{\mathtt{iw}_1, \cdots, \mathtt{iw}_{2n}\}$ be the set of input wires to each gate of the circuit. The topology of the circuit $C_f$ can be described by a mapping $\pi_C : \{1, \cdots, |\mathtt{OW}|\} \rightarrow \{1, \cdots, |\mathtt{IW}|\}$. Though each internal gate has only a single outgoing wire, it can have arbitrary fan-out. This is handled by mapping an outgoing wire $\mathtt{ow_i} \in \mathtt{OW}$ to multiple incoming wires in $\mathtt{IW}$. The full protocol $\mathtt{semiPFE}$ is described in Figure 1.

**Correctness.** In step (5) of the protocl $\mathtt{semiPFE}$, if the linear polynomial $T_i(X) = P_i(X)$, then the equation (9) shows that $k_{l+i} = k^0_{l+i} + \Delta$. Otherwise $T_i(X) = Q_i(X)$ and the equation (9) shows that $k_{l+i} = k^0_{l+i}$. This shows the correctness of the protocol.

**Security.** The security for PFE protocols can be defined in the semi-honest adversary model and in the malicious adversary model. In the semi-honest model, we assume that both participants follow the protocol honestly but both of them may be curious and try to learn some additional information from their protocol view. Let $\mathtt{view}_i(1^\kappa, x, C_f)$ $(i = 1, 2)$ be the view of the participant $P_i$ during the PFE protocol execution when $P_1$ holds input $x$ and $P_2$ holds $C_f \in \mathcal{C}$, where $\mathcal{C}$ is a class of circuits. The protocol is called a secure $\mathcal{C}$-PFE protocol if there exist probabilistic polynomial time simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that for all probabilistic polynomial time algorithm $D$, we have

$$|Pr[D(\mathcal{S}_1(1^\kappa, x)) = 1] - Pr[D(\mathtt{view}_1(1^\kappa, x, C_f)) = 1]| = \mathrm{negl}(\kappa)$$

and

$$|Pr[D(\mathcal{S}_2(1^\kappa, C_f, C_f(x)) = 1] - Pr[D(\mathtt{view}_2(1^\kappa, x, C_f)) = 1]| = \mathrm{negl}(\kappa)$$

The provable security in the above semi-honest adversary model for our protocol $\mathtt{semiPFE}$ follows from the proof in Katz and Malka [7] by observing the following fact: given a pair of key values of the incoming wires of a gate, $P_2$ can compute one key values for the outgoing wire, but cannot distinguish the other key values for the outgoing wire from random. This is true for the protocol $\mathtt{semiPFE}$ since the other key values for the outgoing wire is defined using a linear interpolation with a value which is unknown to $P_2$. The details are omitted here.

Mohassel and Sadeghian [12] proposed a framework for designing PFE protocols by considering circuit topology privacy and secure evaluation of circuit gates independently. Specifically, they reduce the task of the circuit topology hiding (CTH) to oblivious evaluation of a mapping that encodes the topology of the circuit and they design a private gate evaluation (PGE) sub-protocol. Mohassel and Sadeghian then showed how to naturally combine CTH and PGE to obtain an efficient and secure PFE. The CTH functionality is implemented by

**Fig. 1.** Protocol `semiPFE`

1. $P_1$ generates a private and public key pair $(\mathtt{sk}, \mathtt{pk})$ for an additive homomorphic encryption scheme such as Paillier scheme and chooses a circuit-wide global offset value $\Delta \in \mathbb{F}_{2^t}$. $P_1$ chooses $l+n$ outgoing-wire keys $k_i^0 \in \mathbb{F}_{2^t}$ and sets $k_i^1 = k_i^0 + \Delta$ for $1 \leq i \leq l+n$. $P_1$ sends $\mathtt{pk}, \mathtt{Enc}_{\mathtt{pk}}(k_1^0), \cdots, \mathtt{Enc}_{\mathtt{pk}}(k_{l+n}^0)$ to $P_2$.
2. For each gate $i$ with incoming wires $\mathtt{ow}_\mathtt{j}, \mathtt{ow}_\mathtt{k}$, $P_2$ chooses random $a_i, a_i' \in \mathbb{F}_{2^t}$ and re-randomize the encrypted wire labels for gate $i$ as

$$\mathtt{encG}_i = \left(\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i), \mathtt{Enc}_{\mathtt{pk}}(k_k^0 + a_i'), \mathtt{Enc}_{\mathtt{pk}}(k_{l+i}^0)\right).$$

   $P_2$ sends $\mathtt{encG}_1, \cdots, \mathtt{encG}_n$ to $P_1$.
3. For each $i = 1, \cdots, n$, $P_1$ decrypts $\mathtt{encG}_i$ to obtain the keys $(L_i^0, R_i^0, k_{l+i}^0)$ where $L_i^0 = k_j^0 + a_i$ and $R_i^0 = k_k^0 + a_i'$. $P_1$ defines $L_i^1 = L_i^0 + \Delta$, $R_i^1 = R_i^0 + \Delta$, and prepares the garbled version of the $i$-th gate as follows. Let $H_i$ be a gate $i$ specific hash function with $(t+\tau)$-bit outputs and let

$$\begin{array}{l} K_{00}||N_{00} = H_i(L_i^0 + R_i^0 \mod 2^t) \\ K_{01}||N_{01} = H_i(L_i^0 + R_i^0 + \Delta \mod 2^t) \\ K_{10}||N_{10} = H_i(L_i^0 + R_i^0 + \Delta \mod 2^t) \\ K_{11}||N_{11} = H_i(L_i^0 + R_i^0 + 2\Delta \mod 2^t) \end{array} \tag{8}$$

   where $K_{00}, K_{01}, K_{10}, K_{11} \in \mathbb{F}_{2^t}$ and $N_{00}, N_{01}, N_{10}, N_{11} \in \mathbb{F}_{2^\tau}$.
   (a) Let $P_i(X)$ be a linear polynomial passing through the points: $(N_{00}, K_{00})$ and $(N_{01}, K_{01})$.
   (b) Set $\gamma_i^0 = P_i(0) - \Delta \mod 2^t$ and let $Q_i(X)$ be a linear polynomial passing through the points: $(0, \gamma_i^0)$ and $(N_{11}, K_{11})$.
   (c) Let $X_{i,0}$ be a solution of the equation $P_i(X) = Q_i(X)$.
   (d) The garbled version of the $i$-th gate is $\mathtt{GG}_i = \langle X_{i,0}, P_i(X_{i,0}), k_{l+i}^0 - \gamma_i^0 \rangle$
4. $P_1$ sends $\mathtt{GG}_1, \cdots, \mathtt{GG}_n$ to $P_2$. In addition, for the input $x = x_1 \cdots x_l$ that $P_1$ holds, $P_1$ sends $k_1^{x_1}, \cdots, k_l^{x_l}$ to $P_2$.
5. Now $P_2$ has the keys $k_1^{x_1}, \cdots, k_l^{x_l}$ for the outgoing wire $i \in \{1, \cdots, l\}$. For each gate $i$ that $P_2$ has both incoming wire key labels, $P_2$ computes the $i$-gate outgoing wire key label $k_{l+i}$ as follows: Assume that the $i$-th gate have incoming wires $\mathtt{ow}_\mathtt{j}, \mathtt{ow}_\mathtt{k}$ and $P_2$ have already determined keys $k_j, k_k$ for outgoing wires $\mathtt{ow}_\mathtt{j}, \mathtt{ow}_\mathtt{k}$. $P_2$ computes keys $L_i = k_j + a_i$ and $R_i = k_k + a_i'$ for the left and right incoming wires to gate $i$ respectively. $P_2$ computes

$$K_i||N_i = H_i(R_i + L_i \mod 2^t).$$

   Let $T_i(X)$ be a linear polynomial passing through the following two points: $(N_i, M_i)$ and $(X_{i,0}, P_i(X_{i,0}))$. $P_2$ sets the outgoing wire keys

$$k_{l+i} = T_i(0) + k_{l+i}^0 - \gamma_i^0 = k_{l+i}^0 + T_i(0) - P_i(0) + \Delta \mod 2^t. \tag{9}$$

   Once $P_2$ has determined key $k_{l+n}$, it can use an oblivious transfer protocol with $P_1$ to learn the circuit output $f(x)$.

an efficient oblivious evaluation of the mapping $\pi_C$ using generalized switching networks and oblivious transfers. The PGE functionality is a PFE protocol for a single gate circuit where $P_2$ provides the gate's functionality and $P_1$ provides the input to the gate. The PGE functionality is based on Yao's garbled circuit and our above linear interpolation approach in the protocol `semiPFE` could be used in the same way to improve the PGE efficiency by a factor of 25%. The details are omitted here.

### 6.2 PFE protocols against malicious participants

Section 6.1 presents an efficient protocol `semiPFE` against semi-honest adversaries. This protocol is insecure against active adversaries. For example, in step (2) of the E protocol `semiPFE`, $P_2$ may generate the wires $\texttt{encG}_i$ in a malicious way to learn $P_1$'s private input $x$. Specifically, assume that $\texttt{ow}_1, \cdots, \texttt{ow}_l$ are the circuit input wires. For each gate $i$, $P_2$ can choose random $a_i, a_i' \in \mathbb{F}_{2^t}$ and re-randomize the encrypted wire labels for gate $i$ as

$$\texttt{encG}_i = \left(\texttt{Enc}_{\texttt{pk}}(\lambda(j,l)k_j^0 + a_i), \texttt{Enc}_{\texttt{pk}}(\lambda(j,l)k_k^0 + a_i'), \texttt{Enc}_{\texttt{pk}}(k_{l+i}^0)\right)$$

where

$$\lambda(j,l) = \begin{cases} 0 & \text{if } j < l \\ 1 & \text{if } j \geq l \end{cases}$$

$P_2$ sends $\texttt{encG}_1, \cdots, \texttt{encG}_n$ to $P_1$. With this revision of the $\texttt{encG}_i$, $P_2$ may learn the last bit $x_l$ of $P_1$'s private input $x = x_1 \cdots x_l$. Assume that $P_1$ provide the wire labels $k_1^{x_1}, \cdots, k_l^{x_l}$ for the private input $x$. By the construction of $\texttt{encG}_i$, $P_2$ can evaluate the garbled circuit to obtain $f(0, \cdots, 0, x_l)$. By comparing whether $f(0, \cdots, 0, x_l) = f(0, \cdots, 0, 0)$ or $f(0, \cdots, 0, x_l) = f(0, \cdots, 0, 1)$, $P_2$ may learn the value of $x_l$.

Zero-knowledge proofs could be used to make the protocol `semiPFE` secure against active malicious participants. However, performance of the resulting protocol could not compete with PFE protocols based on SFE with universal circuits in the malicious adversary model.

Security definition for PFE protocols against malicious adversaries uses the real protocol execution to simulate an ideal world protocol execution by a trusted party (see, e.g., Canetti [4]). In the real-world execution, protocol participants jointly run the protocol and the adversary $\mathcal{A}$ is allowed to corrupt a participant. Let $\mathcal{A}_i$ $(i = 1, 2)$ be the probabilistic polynomial time adversary that corrupts the participant $P_i$. In the ideal world evaluation, all participants submit their inputs to a trusted party who will evaluate the entire protocol himself and there is a simulator $\mathcal{S}_i$ for the subset of participants controlled by the adversary $\mathcal{A}_i$ in the real world evaluation. Intuitively, a protocol is called a secure $\mathcal{C}$-PFE protocol if there exist simulators such that the real world protocol evaluation simulates the ideal world protocol evaluation. This intuition is formally captured by requiring that the following two distributions are computationally indistinguishable.

- The honest participants' outputs and the adversary $\mathcal{A}$'s view in the real-world execution.

– The honest participants' outputs and the simulator $\mathcal{S}$'s view in the ideal-world execution.

**Real-world execution.** In the real world execution, let $\mathtt{out}_1, \mathtt{out}_2$ denote the output of $P_1$ and $P_2$ respectively. For each individual adversary $\mathcal{A}_i$ $(i = 1, 2)$, there are two candidate views that we should consider. As an example, for the adversary $\mathcal{A}_1$, we need to consider the following two scenarios.

– If $P_2$ is honest, then we need to consider $\mathtt{view}_{\mathcal{A}_1,1} = \mathtt{view}_{\mathcal{A}_1} \cup \mathtt{out}_2$.
– If $P_2$ is dishonest, then we need to consider $\mathtt{view}_{\mathcal{A}_1,0} = \mathtt{view}_{\mathcal{A}_1}$.

**Ideal-world execution.** In the ideal world execution, $P_1$ sends $x$ to the trusted party and $P_2$ sends $C_f$ to the trusted party if they are honest. For a dishonest participant, she sends either what she holds or a random string (it could be in the correct syntax format of a legal protocol message) to the trusted party. The trusted party computes $C_f(x)$ and sends it to $P_2$. We use $\mathtt{out}_1, \mathtt{out}_2$ to denote the output sent to $P_1$ and $P_2$ respectively by the trusted party and use $\mathcal{S}_1, \mathcal{S}_1$ to denote the simulators for $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. For each individual adversary, we need to construct a simulator $\mathcal{S}$ (that is, $\mathcal{S}_1$ or $\mathcal{S}_2$). But for this single simulator $\mathcal{S}$, we need to consider two candidate views derived from the other adversary who may control the other participants. As an example, for the adversary $\mathcal{A}_1$, we need to consider the following two potential views.

– If $P_2$ is honest, then we need to consider $\mathtt{view}_{\mathcal{S}_1,1} = \mathtt{view}_{\mathcal{S}_1} \cup \mathtt{out}_2$.
– If $P_2$ is dishonest, then we need to consider $\mathtt{view}_{\mathcal{S}_1,0} = \mathtt{view}_{\mathcal{S}_1}$.

**Definition 3.** *A two party protocol $\Pi$ is called a* secure *$\mathcal{C}$-PFE protocol if there are probabilistic polynomial time simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that the following four pairs of probabilistic distributions are computationally indistinguishable over the security parameter $\kappa$.*

$$\begin{aligned}
\mathtt{view}_{\mathcal{S}_1,0}(1^\kappa, x, C_f) &\overset{c}{\sim} \mathtt{view}_{\mathcal{A}_1,0}(1^\kappa, x, C_f)\\
\mathtt{view}_{\mathcal{S}_1,1}(1^\kappa, x, C_f) &\overset{c}{\sim} \mathtt{view}_{\mathcal{A}_1,1}(1^\kappa, x, C_f)\\
\mathtt{view}_{\mathcal{S}_2,0}(1^\kappa, x, C_f) &\overset{c}{\sim} \mathtt{view}_{\mathcal{A}_2,0}(1^\kappa, x, C_f)\\
\mathtt{view}_{\mathcal{S}_2,1}(1^\kappa, x, C_f) &\overset{c}{\sim} \mathtt{view}_{\mathcal{A}_2,1}(1^\kappa, x, C_f).
\end{aligned}$$

*In the above list, the views are dependent on the security parameter $\kappa$ which is omitted.*

Katz and Malka [7] proposed a revision of their PFE protocol to achieve security against a malicious participant $P_1$. Specifically, they revised their protocol by requiring $P_1$ to prove to $P_2$ the following facts (in the following, we use our protocol `semiPFE` instead of their original protocols):

– The public key `pk` communicated in Step 1 of `semiPFE` was generated using the specified key generation algorithm `sHE.Gen`.
– The ciphertexts $\mathtt{Enc}_{\mathtt{pk}}(k_1^0), \cdots, \mathtt{Enc}_{\mathtt{pk}}(k_{l+n}^0)$ communicated in Step 1 of `semiPFE` are well-formed ciphertexts using the public key `pk`.

– The garbled circuits in Step 3 are constructed correctly.
– The inputs are encoded correctly in Step 6.

A similar proof as in [7] could be used to show that

$$\begin{array}{l}\mathtt{view}_{\mathcal{S}_1,1}(1^\kappa, x, C_f) \overset{c}{\sim} \mathtt{view}_{\mathcal{A}_1,1}(1^\kappa, x, C_f)\\ \mathtt{view}_{\mathcal{S}_1,0}(1^\kappa, x, C_f) \overset{c}{\sim} \mathtt{view}_{\mathcal{A}_1,0}(1^\kappa, x, C_f)\end{array}$$

for our protocol `semiPFE`. In the same way, if we require $P_2$ to prove to $P_1$ the knowledge of $a_i, a_i' \in \mathbb{F}_{2^t}$ $(i = 1, \cdots, n)$ for the ciphertexts

$$\mathtt{encG}_i = \left(\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i), \mathtt{Enc}_{\mathtt{pk}}(k_k^0 + a_i'), \mathtt{Enc}_{\mathtt{pk}}(k_{l+i}^0)\right)$$

communicated in Step 2 and that the circuit encoded using $\mathtt{encG}_1, \cdots, \mathtt{encG}_{n+l}$ belongs to $\mathcal{C}$, then we can show that

$$\begin{array}{l}\mathtt{view}_{\mathcal{S}_2,1}(1^\kappa, x, C_f) \overset{c}{\sim} \mathtt{view}_{\mathcal{A}_2,1}(1^\kappa, x, C_f)\\ \mathtt{view}_{\mathcal{S}_2,0}(1^\kappa, x, C_f) \overset{c}{\sim} \mathtt{view}_{\mathcal{A}_2,0}(1^\kappa, x, C_f)\end{array}.$$

### 6.3 Circuit private PFE protocols with malicious $P_1$

The discussion in the preceding section shows that the protocol `semiPFE` could be revised to be secure against malicious participants using zero-knowledge proofs. Zero-knowledge proofs are normally expensive and the resulting protocols may not outperform universal circuit based PFE protocols. In certain practical applications, we may want to protect the circuit privacy from a malicious participant $P_1$ and assume that $P_2$ is semi-honest. For this kind of scenarios, it is not necessary for the participant $P_1$ to prove to $P_2$ that the garbled circuits in Step 3 are constructed correctly. Since it will not help $P_1$ to learn any information of $P_2$'s circuit $C_f$ by constructing incorrect garbled circuits in Step 3 of `semiPFE`. Similarly, $P_1$ does not need to prove to $P_2$ that the inputs are encoded correctly in Step 6. In the following paragraphs, we sketch the construction of a more efficient protocol `privPFE` that leaks zero information about the circuit $C_f$ to a malicious $P_1$.

Though other singly homomorphic encryption schemes could be used, we use Paillier's encryption scheme to simplify the discussion. In Paillier's scheme, the public key $\mathtt{pk} = (n, g)$ consists of two integers where $n = pq$ divides the order of $g \in \mathbb{Z}_{n^2}^*$ and $p, q$ are two prime numbers. The private key $\mathtt{sk} = (\lambda, \mu)$ is a pair of integers where $\lambda = \mathrm{lcm}(p-1, q-1)$ and $\mu = \left(\frac{(g^\lambda \mod n^2)-1}{n}\right)^{-1} \mod n$. A message $m$ is encrypted to $c = \mathtt{Enc}_{\mathtt{pk}}(m) = g^m \cdot r^n \mod n^2$ for a randomly selected $r \in \mathbb{Z}_n^*$. A ciphertext $c$ is decrypted to the message $m = \mathtt{Dec}_{\mathtt{sk}}(c) = \frac{\mu((c^\lambda \mod n^2)-1)}{n} \mod n$.

In the protocol `semiPFE`, the only message that $P_2$ sends to $P_1$ is the oblivious gates $\mathtt{encG}_i$ $(i = 1, \cdots, n)$. $P_1$ will not learn any information about the circuit $C_f$ if $P_1$ cannot correlate the ciphertext $\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i)$ to the ciphertext

$\mathtt{Enc}_{\mathtt{pk}}(k_j^0)$. This is guaranteed by the semantic security of the homomorphic encryption scheme. In other words, if $P_1$ can prove to $P_2$ that the public key is generated using the specified key generation algorithm then the circuit privacy is guaranteed. However, if Paillier's scheme is used then zero knowledge proof is necessary at all. It is sufficient for $P_2$ to check that the condition $g \in \mathbb{Z}_{n^2}^*$ holds for the public key $(n, g)$ generated by $P_1$ where we assume that $P_2$ will not leak any information about the circuit $C_f$ on purpose. Specifically, the new protocol `privPFE` is described in Figure 2.

**Fig. 2.** Protocol `privPFE`

1. $P_1$ generates a private and public key pair $(\mathtt{sk}, \mathtt{pk})$ for Paillier's encryption scheme where $\mathtt{sk} = (\lambda, \mu)$, $\mathtt{pk} = (n, g)$, and $n = pq$.
2. $P_1$ chooses two circuit-wide global offset values $\Delta, \Gamma \in \mathbb{F}_{2^t}$. $P_1$ chooses $l+n$ outgoing-wire keys $k_i^0 \in \mathbb{F}_{2^t}$ and sets $k_i^1 = k_i^0 + \Delta$ for $1 \le i \le l+n$. $P_1$ sends $\mathtt{pk}, \mathtt{Enc}_{\mathtt{pk}}(k_1^0), \cdots, \mathtt{Enc}_{\mathtt{pk}}(k_{l+n}^0)$ to $P_2$.
3. $P_2$ verifies that $g \in \mathbb{Z}_{n^2}^*$ and $\mathtt{Enc}_{\mathtt{pk}}(k_i^0) \in \mathbb{Z}_{n^2}^*$ for $i = 1, \cdots, n+l$.
4. For each gate $i$ with incoming wires $\mathtt{ow}_\mathtt{j}, \mathtt{ow}_\mathtt{k}$, $P_2$ chooses random $a_i, a_i' \in \mathbb{F}_{2^t}$ and re-randomize the encrypted wire labels for gate $i$ as
$$\mathtt{encG}_i = \left(\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i), \mathtt{Enc}_{\mathtt{pk}}(k_k^0 + a_i'), \mathtt{Enc}_{\mathtt{pk}}(k_{l+i}^0)\right).$$
$P_2$ sends $\mathtt{encG}_1, \cdots, \mathtt{encG}_n$ to $P_1$.
5. For each $i = 1, \cdots, n$, $P_1$ decrypts $\mathtt{encG}_i$ to obtain the keys $(L_i^0, R_i^0, k_{l+i}^0)$ where $L_i^0 = k_j^0 + a_i$ and $R_i^0 = k_k^0 + a_i'$. $P_1$ defines $L_i^1 = L_i^0 + \Delta$, $R_i^1 = R_i^0 + \Delta$, and prepares the garbled version of the $i$-th gate as follows. Let $H_i$ be a gate $i$ specific hash function with $(t+\tau)$-bit outputs and let
$$\begin{array}{l} K_{00}||N_{00} = H_i(\Gamma + L_i^0 + R_i^0 \mod 2^t) \\ K_{01}||N_{01} = H_i(\Gamma + L_i^0 + R_i^0 + \Delta \mod 2^t) \\ K_{10}||N_{10} = H_i(\Gamma + L_i^0 + R_i^0 + \Delta \mod 2^t) \\ K_{11}||N_{11} = H_i(\Gamma + L_i^0 + R_i^0 + 2\Delta \mod 2^t) \end{array} \quad (10)$$
where $K_{00}, K_{01}, K_{10}, K_{11} \in \mathbb{F}_{2^t}$ and $N_{00}, N_{01}, N_{10}, N_{11} \in \mathbb{F}_{2^\tau}$. Set the garbled version of the $i$-th gate as $\mathtt{GG}_i = \langle X_{i,0}, P_i(X_{i,0}), k_{l+i}^0 - \gamma_i^0 + \Gamma \rangle$ where $\langle X_{i,0}, P_i(X_{i,0}), k_{l+i}^0 - \gamma_i^0 \rangle$ is constructed in the same way as Step 5 of the protocol `semiPFE`.
6. $P_1$ sends $\mathtt{GG}_1, \cdots, \mathtt{GG}_n$ to $P_2$. In addition, for the input $x = x_1 \cdots x_l$ that $P_1$ holds, $P_1$ sends $\Gamma + k_1^{x_1}, \cdots, \Gamma + k_l^{x_l}$ to $P_2$.
7. The process for $P_2$ to evaluate the garbled circuit to obtain $C_f(x)$ remains the same as the Step 5 of the protocol `semiPFE`.

**Correctness.** The correctness of the protocol `privPFE` can be verified straightforwardly in the same way as that for the protocol `semiPFE`.

**Privacy**. In the following, we sketch a proof of privacy for the protocol `privPFE` against malicious $P_1$. First we show that a dishonest $P_1$ will learn nothing about the circuit $C_f$ except the circuit size unless $P_2$ leaks certain information about

$C_f$ on purpose. As we have mentioned in the preceding paragraphs, the only information that $P_2$ sends to $P_1$ is the set of oblivious gates

$$\texttt{encG}_i = \left(\texttt{Enc}_{\texttt{pk}}(k_j^0 + a_i), \texttt{Enc}_{\texttt{pk}}(k_k^0 + a_i'), \texttt{Enc}_{\texttt{pk}}(k_{l+i}^0)\right)$$

for $i = 1, \cdots, n$. In Step 3, $P_2$ verifies that $g \in \mathbb{Z}_{n^2}^*$ and $\texttt{Enc}_{\texttt{pk}}(k_i^0) \in \mathbb{Z}_{n^2}^*$ for $i = 1, \cdots, n+l$. Thus if $a_i, a_i'$ are chosen uniformly at random, then $\texttt{Enc}_{\texttt{pk}}(k_j^0 + a_i), \texttt{Enc}_{\texttt{pk}}(k_k^0 + a_i')$ are values distributed uniformly at random over $\mathbb{Z}_{n^2}^*$ and are independent of the values $\texttt{Enc}_{\texttt{pk}}(k_j^0)$ and $\texttt{Enc}_{\texttt{pk}}(k_k^0)$. In a summary, unless $P_2$ chooses $a_i, a_i'$ nonuniformly, $P_1$ learns no information about the circuit $C_f$ except the circuit size. Note that the privacy of $C_f$ is preserved unconditionally. Secondly, a semi-honest participant $P_2$ learns nothing about the private input $x$ except the final output $f(x)$. The proof is similar to the proofs in [7] and the details are omitted. This completes the proof of privacy.

### 6.4 Secure PFE protocols against two malicious participants

In order to protect the privacy of $P_1$'s input $x$ againt a malicious $P_2$ in the protocol `privPFE`, $P_2$ needs to prove to $P_1$ that the circuit defined by the oblivious gates $\texttt{encG}_1, \cdots, \texttt{encG}_{n+l}$ belongs to the specified circuit class $\mathcal{C}$. Otherwise, the circuit corresponding to these oblivious gates could be a simple circuit such as $\neg(x_1 \wedge x_1)$ which leaks information about the input value $x = x_1 \cdots x_l$ from the output $C_f(x)$. In other words, the protocol `privPFE` is not secure against malicious participant $P_2$.

For PFE protocols with circuit owner $P_2$ learning the final output $C_f(x)$, it seems to be inherently necessary to have participant $P_2$ to prove to $P_1$ that the circuit defined by the oblivious gates $\texttt{encG}_1, \cdots, \texttt{encG}_{n+l}$ belongs to the specified circuit class $\mathcal{C}$. Otherwise, the protocol could not be secure against a malicious participant $P_2$. For applications where only the participant $P_1$ needs to learn the final output $C_f(x)$, the protocol `privPFE` is also secure against both malicious $P_1$ and malicious $P_2$. Let us revise the protocol `privPFE` to a new protocol `secPFE` as in Figure 3.

**Fig. 3.** Protocol `secPFE`

The protocol proceeds as in `privPFE`. In the last step, instead of $P_2$ obliviously learning exactly one of the key labels $k_{n+l}^0$ and $k_{n+l}^0 + \Delta$, $P_2$ handles the garbled circuit evaluation output to $P_1$. Note that the garbled circuit evaluation output is either $k_{n+l}^0$ or $k_{n+l}^0 + \Delta$.

The correctness of the protocol `secPFE` is straightforward. For the security proof, we distinguish two cases. In the first case we assume that $P_1$ is malicious. In this case the proof is identical to the privacy proof for the protocol `privPFE` since the only extra information that $P_2$ delivers to $P_1$ is the final output key label $k_{n+l}^0$ or $k_{n+l}^0 + \Delta$ which leaks no information about the circuit topology. In

other words, a malicious $P_1$ learns no information about the circuit $C_f$. In the second case, we assume that $P_2$ is malicious. In this case, let $\mathtt{ow}_1, \cdots, \mathtt{ow}_l$ be the $l$ input wires. Assume that the $i$th gate

$$\mathtt{encG}_i = \left(\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i), \mathtt{Enc}_{\mathtt{pk}}(k_k^0 + a_i'), \mathtt{Enc}_{\mathtt{pk}}(k_{l+i}^0)\right)$$

contains one or two input wires. Note that $P_1$ garbles this $i$th gate using ingoing key labels $(\Gamma + k_j^0 + a_i, \Gamma + k_j^0 + a_i + \Delta)$ and $(\Gamma + k_k^0 + a_k, \Gamma + k_k^0 + a_k + \Delta)$ respectively. Without loss of generality, we may assume that $j \leq l$ (that is, $\mathtt{ow}_j$ is an input wire). For the input wire $\mathtt{ow}_j$, $P_1$ provides the input key label $\Gamma + k_j^{x_j} = \Gamma + k_j^0 + x_j\Delta$ to $P_2$ corresponding to the input bit $x_j$. We can distinguish the following two cases:

- $P_2$ knows the value of $k_j^0 + a_i$. In this case, unless Paillier's encryption scheme is not semantically secure, $P_2$ does not follow the protocol by choosing a random $a_i$ to generate the ciphertext $\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i)$. Instead, $P_2$ chooses a value $c_i = k_j^0 + a_i$ and let $\mathtt{Enc}_{\mathtt{pk}}(k_j^0 + a_i) = \mathtt{Enc}_{\mathtt{pk}}(c_i)$. In this case, $P_2$ can not distinguish $a_i$ from a random value. Thus $P_2$ can not distinguish $\Gamma + k_j^0 + a_i + x_j\Delta$ from a random value. Consequently, $P_2$ cannot go ahead to evaluate the $i$th garbled gate.
- $P_2$ does not know the value of $k_j^0 + a_i$. In this case, $P_2$ may or may not follow the protocol. In either case, $P_2$ can not distinguish $k_j^0$ from a random value. If $P_2$ followed the protocol and selected a known value $a_i$, then $P_2$ can compute the key value $\Gamma + k_j^0 + a_i + x_j\Delta$ and continue the garbled gate evaluation. If $P_2$ has not followed the protocol and selected $k_j^0 + a_i$ in a way that she does not know the value of $a_i$, then $P_2$ can not compute the key value $\Gamma + k_j^0 + a_i + x_j\Delta$ and cannot continue the garbled gate evaluation.

With above discussion, a similar proof for garbled circuit security as in [3, 7] could be used to show that a malicious participant $P_2$ learns no information about the input $x$ in case that the Paillier's encryption scheme is semantically secure and the hash functions used in the protocol can be considered as random oracles. The details of the proof are omitted.

## 7 Circuit garbling scheme GRRcirc

The half-gates technique garbles each odd gates to two ciphertexts and even gates are free. However, the evaluator needs to carry out two cryptographic hash (or encryption) operations for each odd gate. In our GPGRR2 scheme, the evaluator needs to carry out one cryptographic hash (or encryption) operation and one multiplicative inverse operation in the finite field $\mathbb{F}_{2^t}$. In case that one needs a $\Phi_{\mathtt{circ}}$-secure garbling scheme and prefers multiplicative inverse operations than cryptographic hash (or encryption) operations, one may revise the scheme GPGRR2 by adding additional conversion processes (either free or with one additional ciphertexts) to obtain a free-XOR compatible GRRcirc scheme.

As a high level description, the conversion process is as follows. For each odd gate, if an output wire $z$ is going to even gates, then we can let the output wire

$z$ to satisfy the condition "$k_z^1 = k_z^0 \oplus \Delta$" instead of "$k_z^1 = k_z^0 + \Delta \mod 2^t$". For each odd gate, if one or two input wires $x$ satisfy "$k_x^1 = k_x^0 \oplus \Delta$" instead of "$k_z^1 = k_z^0 + \Delta \mod 2^t$", we can add a conversion ciphertext to translate the condition "$k_x^1 = k_x^0 \oplus \Delta$" to the condition "$k_z^1 = k_z^0 + \Delta \mod 2^t$". Furthermore, we use the GPGRR2 optimization technique to reduce two ciphertexts to one ciphertext for as many odd gates as possible. After the above revision, all even gates are free and each odd gate has one, two, or three ciphertexts. Specifically, the garbling scheme GRRcirc proceeds as follows.

1. For each odd gate such that all input wire labels satisfy the invariance property (6) and the output wire is only used by odd gates, garble the gate using the scheme GPGRR2. That is, let the garbled output wire labels satisfy the property $k_z^1 = k_z^0 + \Delta \mod 2^t$. This garbled gate contains two ciphertexts.
2. For each odd gate such that at least one input wire label does not satisfy the invariance property (6) and the output is only used by odd gates, garble the gate using the GRR3 with three ciphertexts.
3. For each odd gate with all fanout wires $z$ going to even gates, depending on whether the input wires satisfy the invariance property (6) or not, revise either the above step 1 or the above step 2 to garble the gate so that the output wire has garbled wire labels $k_z^0$ and $k_z^1 = k_z^0 \oplus \Delta$. This garbled gate contains two or three ciphertexts.
4. For each odd gate with fanout wires going to both odd and even gates, garble it with three ciphertexts so that the output wire has garbled wire labels $k_z^0$ and $k_z^1 = k_z^0 \oplus \Delta$ for even gates and has garbled wire labels $k_z^0$ and $k_z^1 = k_z^0 + \Delta \mod 2^t$ for odd gates. Our experiments have not found such kind of gates for the commonly used circuits that we will discuss later.
5. For each odd gate, use the following process to reduce the number of ciphertexts to one if possible. In the following process, a gate $g_1$ is called a sibling gate of $g_2$ if there exists a gate $g_3$ such that the two input wires of $g_3$ are the output wires of $g_1$ and $g_2$ respectively.
   (a) Mark all even gates as "FINAL".
   (b) If all gates are marked either as "1-Cipher" or as "FINAL". Then the process is over. Otherwise, choose a random odd gate $g$ that is not marked as "1-Cipher" or "FINAL". Let $S = \{g\}$.
   (c) If there exists a gate $g' \notin S$, $g'$ is a sibling of some gate in $S$, and $g'$ is marked as "1-Cipher" or "FINAL", mark all gates in $S$ as "FINAL" and go to Step (5b).
   (d) If there exists a gate $g' \notin S$, $g'$ is a sibling of some gate in $S$, and $g'$ is neither marked as "1-Cipher" nor marked as "FINAL", let $S = S \cup \{g'\}$ and go to Step (5c).
   (e) If there is no gate $g' \notin S$ such that $g'$ is a sibling of some gate in $S$, use the optimized GPGRR2 technique to garble $g$ using one ciphertext and all other gates in $S$ using two ciphertexts appropriately. It is noted that if the garbled gate $g$ contains three ciphertexts originally, then we can only reduce the number of ciphertexts to two instead of one. Mark $g$ as "1-Cipher" and all other gates in $S$ as "FINAL". Go to Step (5b).

6. Each even gate is for free. That is, no ciphertext is required.

We used the above process to compare the proposed garbling scheme GRRcirc against other garbling schemes from the literature. Since it is optional to use the external index bits in GRRcirc, we do not include the external index bits for GRRcirc garbling scheme in the following comparison. Specifically, we compare the garbled circuit sizes for the following circuits that are available from [16]: AES (Key Expanded), DES (Key Expanded), MD5, SHA-1, SHA-256. Note that the circuits for these functions [16] contains AND, XOR, and INV gates. For our comparison, we integrated the INV gates into the AND/XOR gates to obtain OR and NXOR gates. Thus we will only consider circuits with AND/OR/XOR/NXOR gates. We will use $t$ to denote the size of wire labels (e.g., we may take $t = 80$). For the garbling schemes, we compare Yao's classical scheme [18], point-permute [2], GRR3 [13], GRR2 [15], free XOR+GRR3 [10], FleXOR [9], and half-gates [19]. For the FleXOR garbling scheme [9], we used the data for the best performance "safe ordering heuristics" reported in Figure 9 of [9]. For each garbling scheme in Table 1, we have two rows of values for each circuit. The top row contains the number of ciphertexts of the garbled circuits and the bottom row contains the size of the garbled circuits when $t = 80$.

The comparison results in Table 1 show that, GRRcirc has comparable performance with FleXOR. However, it has large garbled circuit size compared with half-gates techniques. As we have mentioned in the previous sections, one may choose to use GRRcirc instead of half-gates if one prefers field multiplicative inverse operations than cryptographic hash (or encryption) operations since for half-gates garbled circuits, each odd gate evaluation requires two cryptographic hash (or encryption) operations while for GRRcirc garbled circuits, each odd gate evaluation requires one cryptographic hash (or encryption) operation and one field multiplicative inverse operation.

## 8 Conclusion

Using a linear interpolation method, we proposed a circuit garbling scheme to garble each circuit gate to at most two ciphertexts with gate functionality privacy. We also proposed an optimization process to garble a circuit in such a way that some gates only contain one ciphertext. It would be interesting to investigate the lower bound for garbled circuit size. We also applied our garbling schemes to constant round PFE protocols and proposed a more efficient PFE protocol that is secure against malicious participant $P_1$ if $P_2$ learns the final output and is secure against two malicious participants $P_1/P_2$ if only $P_1$ learns the final output.

## References

1. M. Ajtai, J. Komlós, and E. Szemerédi. An $o(n \log n)$ sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1983.

**Table 1.** Garbled Circuit Size Comparison

| | AES (KE) | DES (KE) | MD5 | SHA-1 | SHA-256 |
|---|---|---|---|---|---|
| # AND/OR [16] | 5440 | 18175 | 29084 | 37300 | 90825 |
| # XOR/NXOR [16] | 20325 | 1351 | 14150 | 24166 | 42029 |
| # Total gates | 25765 | 19526 | 43234 | 61466 | 132854 |
| classical [18] | 103060t | 78104t | 172936t | 245864t | 531416t |
| | 0.98MB | 0.74MB | 1.65MB | 2.34MB | 5.07MB |
| point-permute [2] | 103060(t+1)t | 78104(t+1) | 172936(t+1) | 245864(t+1) | 531416(t+1) |
| | 1MB | 0.75MB | 1.67MB | 2.37MB | 5.13MB |
| GGR3 [13] | 77295(t+1) | 58578(t+1) | 129702(t+1) | 184398(t+1) | 398562(t+1) |
| | 0.75MB | 0.57MB | 1.25MB | 1.78MB | 3.85MB |
| GGR2 [15], | 51530(t+1) | 39052(t+1) | 86468(t+1) | 122932(t+1) | 265708(t+1) |
| | 0.50MB | 0.38MB | 0.83MB | 1.19MB | 2.57MB |
| free XOR+GRR3 [10] | 16320(t+1) | 54525(t+1) | 87252(t+1) | 111900(t+1) | 272475(t+1) |
| | 0.16MB | 0.53MB | 0.84MB | 1.08MB | 2.63MB |
| FleXOR [9] | 18550(t+1) | 36904(t+1) | N/A | 85438(t+1) | 207253(t+1) |
| | 0.18MB | 0.36MB | N/A | 0.82MB | 2MB |
| half-gates [19] | 10880(t+1) | 36350(t+1) | 58168(t+1) | 74600(t+1) | 181650(t+1) |
| | **0.11MB** | **0.35MB** | **0.56MB** | **0.72MB** | **1.75MB** |
| GRRcirc | 16640t | 37198t | 75584t | 97080t | 225498t |
| | **0.16MB** | **0.35MB** | **0.72MB** | **0.92MB** | **2.15MB** |


2. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. 22nd ACM STOC*, pages 503–513. ACM, 1990.
3. M. Bellare, V. Hoang, and P. Rogaway. Foundations of garbled circuits. In *Proc. 2012 ACM CCS*, pages 784–796. ACM, 2012.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.
5. M. Furst, J.B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.
6. J. Håstad. Computational limitations of small-depth circuits. *PhD Thesis, MIT*, 1987.
7. J. Katz and L. Malka. Constant-round private function evaluation with linear complexity. In *Asiancrypt*, pages 556–571. Springer, 2011.
8. Á. Kiss and T. Schneider. Valiant's universal circuit is practical. In *EUROCRYPT*, pages 699–728. Springer, 2016.
9. V. Kolesnikov, P. Mohassel, and M. Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In *CRYPTO 2014*, pages 440–457. Springer, 2014.
10. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *Proc. ICALP*, pages 486–498. Springer, 2008.
11. V. Kolesnikov and T. Schneider. A practical universal circuit construction and secure evaluation of private functions. In *Financial Cryptography*, pages 83–97. Springer, 2008.
12. P. Mohassel and S. Sadeghian. How to hide circuits in mpc an efficient framework for private function evaluation. In *Proc. EUROCRYPT*, pages 557–574. Springer, 2013.
13. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conf. Electronic Commerce*, pages 129–139. ACM, 1999.
14. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer, 1999.

15. B. Pinkas, T. Schneider, N.P. Smart, and S.C. Williams. Secure two-party computation is practical. In *ASIACRYPT 2009*, pages 250–267. Springer, 2009.
16. N.P. Smart and S. Tillich. Circuits of basic functions suitable for MPC and FHE. http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/.
17. L. Valiant. Universal circuits. In *Proc. 8th ACM STOC*, pages 196–203. ACM, 1976.
18. A. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE FOCS*, pages 162–167. IEEE, 1986.
19. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole. In *Proc. EUROCRYPT*, pages 220–250. Springer, 2015.