Towards Optimal Garbled Circuits in the Standard Model

Ruiyang Li[1,2], Chun Guo[1,2,3], and Xiao Wang[4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China ruiyang.li@main.sdu.edu.cn

[2] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China
chun.guo@sdu.edu.cn

[3] Shandong Research Institute of Industrial Technology, Jinan, Shandong, 250102, China

[4] Northwestern University, Evanston, USA
wangxiao@cs.northwestern.edu

Abstract. State-of-the-art garbling schemes for boolean circuits roughly consist of two families, i.e., *ideal model garbling* that combines linear operations and ideal blockciphers (aiming at maximizing performance), and *PRF-based garbling* that insists on using theoretically sound assumptions. In the linear garbling framework introduced by Zahur, Rosulek, and Evans (Eurocrypt 2015), it was established that garbling an AND gate requires at least $2(\kappa+1)$ bits of ciphertext, with κ as the security parameter. Recent contributions from Lei Fan et al. and Chunghun Baek et al. have provided a detailed model showing that, under the free-XOR setting, which relies on a non-standard assumption, garbling an AND gate requires at least $1.5\kappa+O(1)$ bits. In contrast, regarding PRF-based garbling, the general model and efficiency bounds remain open questions. In this paper, we present a comprehensive model for PRF-based garbled circuits and establish both the communication and computation lower bound. Specifically, we demonstrate that garbling an AND gate requires at least $2\kappa+2$ bits communication and 6 PRF calls, while an XOR gate requires a minimum of κ bits communication and 4 PRF calls. Notably, the state-of-the-art garbling scheme (GLNP scheme) under the PRF assumption, introduced by Shay, Yehuda, Ariel, and Benny (JOC 2018), uses $2\kappa+4$ bits and 8 PRF calls for an AND gate, which exceeds our established lower bound. We finally introduce an optimal garbling scheme showing that our communication and computation lower bounds are tight.

1 Introduction

Garbled circuits. Since proposed by Yao in 1980s [20], garbled circuits (GCs) have became a leading technique for constant-round two-party computation, and have attracted extensive attention. Virtually all existing garbling schemes for boolean circuits are constructed by combining symmetric cryptographic primitives (i.e.,

hash functions such as the SHA and blockciphers such as the AES) and simple operations (including XORs and finite field multiplications). Depending on the symmetric cryptographic assumptions in use, such schemes consists of two families, i.e., *ideal model garbling* and *PRF-based garbling*.

The first family, *ideal model garbling*, aims at optimizing the (communication and computation) costs by using advanced assumptions. In particular, state-of-the-art schemes [21,19] of this family are fully compatible with the Free-XOR optimization proposed by Kolesnikov and Schneider [13], due to which the evaluation of an XOR gate is simply achieved by XORing two relevant wire labels. This significantly improves efficiency because the computation and transmission of XOR gates do not require any AES computation or communication. The shortage of these schemes is the extensive use of strong non-standard cryptographic assumptions, including related-key security [7, Sect. 5], (circular) correlation robustness [12,17,5,6,21,8,19] and even ideal model [3]. In theory, it remains open if one-way functions imply these assumptions. In practice, these ideal models lack secure instantiations in the real world, leading to a gap between theoretical security proofs and the security of actual garbled circuit implementations. This gap could potentially cause security issues[8,9].

The second family, *PRF-based garbling*, insists on using the most conservative assumption that the underlying blockcipher is a pseudorandom function (PRF). Examples include Yao [20,14], PSSW [17, Sect. 5], BHR [4], GLNP [7, Sect. 3 & 4] and Rosulek [18]. As remarked in [7, Sect. 5], the labels of these garbled circuits have to be independent and (pseudo)random, and this cannot be compatible with the Free-XOR technique. Thus, this type of garbled circuit is slightly less efficient compared to the first method. Still, as demonstrated by Gueron et al. [7], this approach remains practical.

While state-of-the-art PRF-based garbling schemes are less efficient than the first family, PRF assumption is the mostly well-understood. In cryptanalytic community, it corresponds to the standard single-key attack scenario. In addition, PRF assumption is closer to NIST specification. In this respect, we remark that NIST has shown some preference towards the standard PRF assumption. In particular, the NIST FIPS 197 [16, Sect. 6.2] (specification of the AES) recommends following NIST SP 800-133 [2] to generate AES keys, while the latter [2, Sect. 3.1] states:

> *All keys shall be based directly or indirectly on the output of an approved RBG (random bit generator).*

Therefore, we believe PRF-based garbling has its position—and would offer important complementary to the upcoming NIST standardization of multi-party threshold schemes [1].

Currently, the best-known ideal model garbling scheme is the TreGates [19], whereas the best-known PRF-based scheme is GLNP [7]. All of them were built upon a long series of improvements, which produces fruitful techniques including point-and-permute [15], Free-XOR [13] and row reduction [15,11,17,21]. With the

above in mind, it is tempting to ask whether it is possible to improve further the state-of-the-art schemes in both ideal model and PRF-based garbled circuits.

In this respect, Zahur et al. [21] was the first that investigate these efficiency bounds. In detail, they introduced a unified design framework called the *linear garbling model*, aiming to capture all practical garbling schemes at that time. In their model, the Garble and Eval algorithms are constructed via linear operations and several non-adaptive calls to a random oracle. Zahur et al. then established a lower bound for the linear model, stating that any garbling scheme fitting into their linear model consumes at least two ciphertexts for garbling an AND gate where the ciphertext is $\kappa + 1$ bits because of the point and permute technique. The formulation of such a model and proving a lower bound are truly significant: *to achieve a smaller ciphertext size, novel constructions must transcend the boundaries of this model.*

Recently, Fan et al. refined the linear model of garbled circuits by introducing a "column-wise" perspective, resulting in several extensions: Model-1 (linear mappings), Model-2 (arbitrary mappings), Model-3 (introducing probabilistic), and Model-3' (slicing with multiple bases). They proved that the lower bounds for Model-1, Model-2, and Model-3 are all $2\kappa+O(1)$, indicating that the lower bound cannot be improved even with non-linear operations. Additionally, the lower bound with linear operations and slicing techniques is $3/2\kappa+O(1)$, which does not outperform TreGates.

Meanwhile, Baek et al. transformed the original linear model into an Algebraic Perspective using Truth Tables and Boolean Functions, extending the linear garbling model while maintaining oracle queries and linear operations. They also proved a lower bound of $3/2\kappa + O(1)$, reinforcing that linear operations alone do not surpass TreGates.

The models above demonstrate that the state-of-the-art TreGates in first-family garbled circuits is communication-optimal across a wide range of schemes. However, similar models and lower bounds for PRF-based garbling remain unexplored, as noted in [19] where this was identified as an open problem.

1.1 Our Contribution

Motivated by the above discussion, we initiate the model and lower bound of PRF-based garble circuits.

New perspective and new model for PRF-based garbled circuits Recalling the "column-wise" perspective from Fan et al., the garbler, with input labels A_0, A_1, B_0, B_1 (where $\mathrm{lsb}(A_0)$ and $\mathrm{lsb}(B_0)$ represent the permute bits for wire a and wire b, respectively), generates n sets of base values M_{ij}^k for $i, j \in \{0, 1\}$ and $k \in [n]$ (if not use "slicing" technique, $n = 1$.). The garbler then assigns appropriate C_0 and C_1, and sets the output labels $\mathsf{OL}_{00}, \mathsf{OL}_{01}, \mathsf{OL}_{10}, \mathsf{OL}_{11}$ based on the gate type and the permute bits. For example, if the gate is an AND gate and the permute bits are $\pi_a = 0$ and $\pi_b = 0$, then $\mathsf{OL}_{00} = \mathsf{OL}_{01} = \mathsf{OL}_{10} = C_0$ and $\mathsf{OL}_{11} = C_1$. Next, the garbler divides the base values and output labels bit by bit. Finally, it searches for the appropriate ciphertexts $G_1, \cdots, G_m$ to satisfy the mapping relation: $\mathsf{MAP}_{ij}(\mathsf{M}_{ij}(t), \widetilde{\mathsf{G}}_{ij}(t)) = \mathsf{OL}_{ij}(t)$, where $i, j \in \{0, 1\}$,

$t \in (\kappa+1)$, and $\widetilde{\mathsf{G}}_{ij}$ is computed from ciphertext $\mathsf{G}_1, \cdots, G_m$. Here, $\mathsf{OL}_{ij}(t)$, $\mathsf{M}_{ij}(t)$, and $\widetilde{\mathsf{G}}_{ij}(t)$ represents the t-th bit of M_{ij}, $\widetilde{\mathsf{G}}_{ij}$, and OL_{ij}. The evaluator, given A_i and B_j, can compute $\mathsf{M}_{ij}(t)$ for $t \in (\kappa+1)$ and use the corresponding MAP function to derive the output labels.

Motivated by the "column-wise" perspective, we introduce the following view for PRF-based garbled circuits. With independent (pseudo)random input labels A_0, A_1, B_0, B_1, the garbler generates n sets of main and auxiliary keys MK_{ij}^n and AK_{ij}^n (if the "slicing" technique is not used, then $n = 1$). The garbler assigns appropriate output labels C_0, C_1, and the permute bit π_c, where C_0, C_1 and A_0, A_1 are independent and pseudorandom, as are C_0, C_1 and B_0, B_1. The garbler then sets the output labels $\mathsf{OL}_{00}, \mathsf{OL}_{01}, \mathsf{OL}_{10}, \mathsf{OL}_{11}$ and output color bits $\mathsf{OC}_{00}, \mathsf{OC}_{01}, \mathsf{OC}_{10}, \mathsf{OC}_{11}$ based on the gate type and the input permute bits. For example, if the gate is an AND gate and the permute bits are $\pi_a = 0$ and $\pi_b = 0$, then $\mathsf{OL}_{00} = \mathsf{OL}_{01} = \mathsf{OL}_{10} = C_0$, $\mathsf{OL}_{11} = C_1$, and $\mathsf{OC}_{00} = \mathsf{OC}_{01} = \mathsf{OC}_{10} = \pi_c$, while $\mathsf{OC}_{11} = \bar{\pi_c}$. Finally, the garbler searches for the appropriate main ciphertexts $\mathsf{MG}_1, \ldots, \mathsf{MG}_m$ and auxiliary ciphertexts $\mathsf{AG}_1, \ldots, \mathsf{AG}_n$ to satisfy the following mapping relations: $\mathsf{MAP}_{ij}(\mathsf{MK}_{ij}, \widetilde{\mathsf{MG}}_{ij}) = \mathsf{OL}_{ij}$ and $\widehat{\mathsf{MAP}}_{ij}(\mathsf{AK}_{ij}, \widetilde{\mathsf{AG}}_{ij}) = \mathsf{OC}_{ij}$, where $i, j \in \{0, 1\}$, and $\widetilde{\mathsf{MG}}_{ij}$ and $\widetilde{\mathsf{AG}}_{ij}$ are computed from the main and auxiliary ciphertexts. The evaluator, given A_i and B_j, can compute MK_{ij} and AK_{ij}, and use the corresponding MAP and $\widehat{\mathsf{MAP}}$ functions to derive the output label and output color bit. Then, we propose a new PRF-based model based on our perspective of PRF-based garbled circuits.

Overall, our model differs from the column-wise models in the following aspects:

1. To satisfy the PRF assumption, we require that the input wire labels of each gate are independently random. We also require that the labels of any input wire are independently random from the output labels. All current PRF-based garbled circuits exhibit this characteristic, which appears essential for the current proof framework of garbled circuits from Y. Lindell and B. Pinkas [14] (we elaborate on this in Appendix A). Our model does not impose restrictions on the specific circuit topology, meaning that any two wires could potentially serve as input wires for a gate. Concretely, the labels of any input and output wires must be independently random for each gate.
2. Reviewing all previous models, labels, and permute bits have been treated as a single entity in the computation. However, labels and permute bits do not necessarily need to be handled in the same way. For example, in the most advanced PRF-based circuits, the GLNP-GRR2 scheme treats labels and permute bits differently, which results in these schemes requiring the transmission of $2\kappa + 4$ bits. Therefore, we question whether PRF-based schemes require extra bit transmission for handling permute bits rather than just 2 bits. In our model, we handle label processing and permute bit handling separately, analyzing each component independently to ensure clear and distinct treatment of both operations.
3. We do not adopt a bit-by-bit analysis approach here, which enhances the simplicity of both our perspective and model.

Model	Gate	Communication Lower Bound	Computation Lower Bound	Assumption
Linear Model	AND gate	$2\kappa + 2$	NO	RO
Model 1,2,3	AND gate	$2\kappa + O(1)$	NO	RO
Model 3'	AND gate	$(w+1)\kappa/w + O(1)$	NO	RO
Linear Model 3'	AND gate	$3/2\kappa + O(1)$	NO	RO
Our Model	AND gate	$2\kappa + 2$	6 PRF calls	PRF
Our Model	XOR gate	κ	4 PRF calls	PRF

Table 1: Models and their lower bounds. Here, κ represents the security parameter. The term 'linear model 3' refers to a variant of Model 3 where the map function is restricted to linear operations. RO denotes the Random Oracle assumption, while PRF refers to the Pseudorandom Function assumption.

Communication and Computation lower bound. Based on our model, we establish both *communication and computational lower bounds* (we explain various previously proposed models and their corresponding lower bounds in table1.). Our results show that for any garbled circuit scheme that satisfies our model, the communication complexity for an AND gate is at least $2\kappa + 2$ bits, and the computational complexity requires at least 6 PRF calls. For an XOR gate, the communication complexity is at least κ bits, and the computational complexity requires at least 4 PRF calls. Our conclusions lead to the following two sub-conclusions:

1. The dicing technique does not further reduce the communication lower bound of PRF-based garbled circuits.
2. Gueron et al. proposed the GLNP-GRR1 scheme for XOR gates [7], requiring only 4 PRF calls and outputting a k-bit ciphertext, which is optimal in our model. They later optimized the computational complexity, reducing it to 3 PRF calls, contradicting our lower bound. We point out that their 3 PRF XOR gate scheme overlooks a special case in the security proof. We clarify this overlooked case in Section 3.1 .
3. The current GLNP-GRR1 garbling scheme for the XOR gate is already communication-optimal and computation-optimal. However, the state-of-the-art GLNP-GRR2 for the AND gate is already communication-optimal in label numbers. However, there may still be room for improvement in handling permute bits, potentially reducing the communication complexity by up to 1%, when $\kappa = 127$. In addition, the GLNP-GRR2 scheme has room for computational improvement, with possible optimizations of up to 25%.

New Constructions Achieving Communication and Computation Lower Bounds. Based on our design model, we have redesigned a new GRR2 scheme, where we reconstructed the main keys only to require 6 PRF evaluations. Additionally, we devised a new method for communicating the color bit, which allows the color bit to be transmitted using just 2 bits. We then combined this AND gate scheme with the GLNP-GRR1 scheme to create a solution that fully

satisfies our lower computational and communication complexity bounds. Compared to the GLNP scheme, the communication complexity of the AND gate in our scheme increases by only about 1%, but its computational complexity is improved by 25%. More importantly, the design of our scheme is based on our perspective, offering an alternative approach to designing garbled circuits. Furthermore, our scheme meets the lower bound, indicating that our computational and communication lower bounds are tight.

2 Preliminaries

Notation. For a bit a, we use $\overline{a}$ to denote the negation of a. For a binary string A, we use $\mathsf{lsb}(A)$ to denote the least significant bit of A. Vectors are shown in bold, like $\mathbf{A}$, while matrices are in calligraphic bold, such as $\boldsymbol{\mathcal{P}}$. The inner product of two vectors is given by $\langle \cdot, \cdot \rangle$, and matrix multiplication is indicated by $\times$. pub refer to a public value.

2.1 Garbling schemes

Definition 1 (Garbling scheme). *Following [4,7], a garbling scheme consists of four algorithms:*

- $\mathsf{Garble}(f) \to (F, e, d)$ *is an algorithm that takes as input a description of a boolean circuit f and returns a triple (F, e, d), where F represents a garbled circuit, e represents input encoding information (i.e., all the labels on the input wires) and d represents output decoding information (i.e., all the labels on the output wires). Following [8], we focus on concrete security and do not use explicit security parameters.*
- $\mathsf{Encode}(e, x) \to X$ *is a function that computes the garbled input X of an input x according to the input encoding e.*
- $\mathsf{Eval}(F, X) \to Y$ *is a function that computes the garbled output Y of a garbled input X under a garbled circuit F.*
- $\mathsf{Decode}(Y, d) \to y$ *is a function that takes as input decoding information d and garbled output Y and returns either the real output y of the circuit or $\perp$.*

Definition 2 (Garbled Circuit Security). *A garbling scheme is secure if it is correct and achieves privacy, obliviousness, and authenticity as follows:* ***Correctness:*** *For any circuit f and input x, after sampling $(F, e, d) \leftarrow \mathit{Garble}(1^\kappa, f)$, $f(x) = \mathit{Decode}(d, \mathit{Eval}(\mathit{Encode}(e, x)))$ holds with all but negligible probability.*
Privacy: *There must be a simulator $\mathcal{S}$ such that for any circuit f and input x the following distributions are indistinguishable.*

$$\boxed{\begin{array}{c}(F, e, d) \leftarrow \mathit{Garble}(1^\kappa, f)\\ X := \mathit{Encode}(e, x)\\ \mathit{return}\ (F, X, d)\end{array}} \qquad \boxed{\begin{array}{c}(F, X, d) \leftarrow \mathcal{S}(1^\kappa, f, f(x))\\ \mathit{return}\ (F, X, d)\end{array}}$$

Obliviousness: *There must be a simulator $\mathcal{S}$ such that for any circuit f and input x the following distributions are indistinguishable.*

$$\boxed{\begin{array}{c}(F, e, d) \leftarrow \mathit{Garble}(1^\kappa, f)\\ X := \mathit{Encode}(e, x)\\ \mathit{return}\ (F, X)\end{array}} \qquad \boxed{\begin{array}{c}(F, X) \leftarrow \mathcal{S}(1^\kappa, f)\\ \mathit{return}\ (F, X)\end{array}}$$

Authenticity: *For any circuit f and input x, no PPT adversary $\mathcal{A}$ can make the following distribution output* `TRUE` *with non-negligible probability.*

$$\boxed{\begin{array}{c}(F, e, d) \leftarrow \mathit{Garble}(1^\kappa, f)\\ X := \mathit{Encode}(e, x)\\ Y \leftarrow \mathcal{A}(F, X)\\ \mathit{return}\ \mathit{Decode}(d, Y) \notin \{f(x), \perp\}\end{array}}$$

3 Our View of Garbling Schemes

Here, we consider all garbling schemes incorporating the point-and-permute optimization technique, so we will first review this optimization. In the point-permute optimization, the garbler secretly selects two labels A_0, A_1 and a "permute bit" π_a for each wire a. Then, through Yao's secure two-party computation protocol, the evaluator receives a label A and a color bit λ_a for each wire a, where the label A represents the logical bit $v_a = \pi_a \oplus \lambda_a$. Note that since the evaluator does not know the value of π_a, they also do not know the value of v_a.

This section focuses on a gate with input labels (A_0, A_1) and (B_0, B_1), and output labels (C_0, C_1). All labels are κ-bit strings. The subscript indicates the color bit for the input labels. For instance, for input wire a, the subscripts of A_0 and A_1 denote the color bit known to the evaluator. In addition, for output wire c, the subscripts of C_0 and C_1 correspond to the logical bit, which remains unknown to the evaluator.

3.1 The Garbled Scheme under PRF assumption

This subsection examines the garbled circuits under the PRF assumption. We consider a PRF $\{0,1\}^\kappa \times \{0,1\}^{\kappa+1} \to \{0,1\}^{\kappa+1}$, which takes an κ-bit key and has both input and output lengths of $\kappa + 1$ bits.

Classic Yao Scheme. We first examine the classical Yao scheme with the point-permute technique. For the garbler, given input labels A_0, A_1, B_0, and B_1, it generates a set of main keys and auxiliary keys to garble a gate, ordered by the input color bits.

$$\text{main keys} \begin{cases} \mathsf{MK}_{00} = F(A_0, 00)[1 \cdots n] \oplus F(B_0, 00)[1 \cdots n] \\ \mathsf{MK}_{01} = F(A_0, 01)[1 \cdots n] \oplus F(B_1, 01)[1 \cdots n] \\ \mathsf{MK}_{10} = F(A_1, 10)[1 \cdots n] \oplus F(B_0, 10)[1 \cdots n] \\ \mathsf{MK}_{11} = F(A_1, 11)[1 \cdots n] \oplus F(B_1, 11)[1 \cdots n] \end{cases}$$

$$\text{auxiliary keys} \begin{cases} \mathsf{AK}_{00} = \mathsf{lsb}(F(A_0, 00)) \oplus \mathsf{lsb}(F(B_0, 00)) \\ \mathsf{AK}_{01} = \mathsf{lsb}(F(A_0, 01)) \oplus \mathsf{lsb}(F(B_1, 01)) \\ \mathsf{AK}_{10} = \mathsf{lsb}(F(A_1, 10)) \oplus \mathsf{lsb}(F(B_0, 10)) \\ \mathsf{AK}_{11} = \mathsf{lsb}(F(A_1, 11)) \oplus \mathsf{lsb}(F(B_1, 11)) \end{cases}$$

The main keys consist of four rows, each κ bits long, and auxiliary keys consist of four rows, each 1 bit long. Next, the garbler randomly generates the permute bit π_c and two κ-bit output labels, C_0 and C_1, with C_0 corresponding to FALSE. We assume, without loss of generality, the input permute bit $\pi_a = 0$ and $\pi_b = 0$ (this convention persists in subsequent examples). Thus, for the AND gate, the fouth row of the main keys (resp. auxiliary keys), where the color bits are (1, 1), yields the output label C_1 (resp. output color bit $\bar{\pi}_c$) and other rows of main keys (resp. auxiliary keys) yields the output label C_0 (resp. output color bit π_c). For the XOR gate, the second and third rows of the main keys (resp. auxiliary keys) yield the output label C_1 (resp. output color bit $\bar{\pi}_c$), and the first and fourth rows of the main keys (resp. auxiliary keys) yield the output label C_0 (resp. output color bit π_c). The garbler's task is to map the main keys to the output labels and map the auxiliary keys to the output color bits, as illustrated below:

$$\underbrace{\begin{array}{ll} \mathsf{MK}_{00} \Rightarrow C_0 & \mathsf{AK}_{00} \Rightarrow c \\ \mathsf{MK}_{01} \Rightarrow C_0 & \mathsf{AK}_{01} \Rightarrow c \\ \mathsf{MK}_{10} \Rightarrow C_0 & \mathsf{AK}_{10} \Rightarrow c \\ \mathsf{MK}_{11} \Rightarrow C_1 & \mathsf{AK}_{11} \Rightarrow \bar{c} \end{array}}_{\text{AND Gate}} \qquad \underbrace{\begin{array}{ll} \mathsf{MK}_{00} \Rightarrow C_0 & \mathsf{AK}_{00} \Rightarrow c \\ \mathsf{MK}_{01} \Rightarrow C_1 & \mathsf{AK}_{01} \Rightarrow \bar{c} \\ \mathsf{MK}_{10} \Rightarrow C_1 & \mathsf{AK}_{10} \Rightarrow \bar{c} \\ \mathsf{MK}_{11} \Rightarrow C_0 & \mathsf{AK}_{11} \Rightarrow c \end{array}}_{\text{XOR Gate}}$$

We denote the main keys as the vector $\mathbf{MK} = (\mathsf{MK}_{00}, \mathsf{MK}_{01}, \mathsf{MK}_{10}, \mathsf{MK}_{11})$ and the auxiliary base as the vector $\mathbf{AK} = (\mathsf{AK}_{00}, \mathsf{AK}_{01}, \mathsf{AK}_{10}, \mathsf{AK}_{11})$. The output labels are represented by the vector $\mathbf{OL} = (\mathsf{OL}_{00}, \mathsf{OL}_{01}, \mathsf{OL}_{10}, \mathsf{OL}_{11})$, where for the AND gate, $\mathsf{OL}_{00} = \mathsf{OL}_{01} = \mathsf{OL}_{10} = C_0$ and $\mathsf{OL}_{11} = C_1$, and for the XOR gate, $\mathsf{OL}_{00} = \mathsf{OL}_{11} = C_0$ and $\mathsf{OL}_{01} = \mathsf{OL}_{10} = C_1$. Similarly, the output color bits are denoted by $\mathbf{OC} = (\mathsf{OC}_{00}, \mathsf{OC}_{01}, \mathsf{OC}_{10}, \mathsf{OC}_{11})$, where for the AND gate, $\mathsf{OC}_{00} = \mathsf{OC}_{01} = \mathsf{OC}_{10} = \pi_c$ and $\mathsf{OC}_{11} = \bar{\pi}_c$, and for the XOR gate, $\mathsf{OC}_{00} = \mathsf{OC}_{11} = \pi_c$ and $\mathsf{OC}_{01} = \mathsf{OC}_{10} = \bar{\pi}_c$.

In the classical Yao scheme, the garbler uses four main ciphertexts, $\mathbf{MG} = (\mathsf{MG}_{00}, \mathsf{MG}_{01}, \mathsf{MG}_{10}, \mathsf{MG}_{11})$, and four auxiliary ciphertexts, $\mathbf{AG} = (\mathsf{AG}_{00}, \mathsf{AG}_{01}, \mathsf{AG}_{10}, \mathsf{AG}_{11})$, to achieve the mapping functions MAP and $\widehat{\mathsf{MAP}}$, which are defined as follows:

$$\mathsf{MAP}(\mathbf{MK}, \mathbf{MG}) \Rightarrow \begin{pmatrix} \mathsf{MK}_{00} \oplus \mathsf{MG}_{00} \\ \mathsf{MK}_{01} \oplus \mathsf{MG}_{01} \\ \mathsf{MK}_{10} \oplus \mathsf{MG}_{10} \\ \mathsf{MK}_{11} \oplus \mathsf{MG}_{11} \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00} \\ \mathsf{OL}_{01} \\ \mathsf{OL}_{10} \\ \mathsf{OL}_{11} \end{pmatrix} = \mathbf{OL}$$

$$\widehat{\mathsf{MAP}}(\mathbf{AK}, \mathbf{AG}) \Rightarrow \begin{pmatrix} \mathsf{AK}_{00} \oplus \mathsf{AG}_{00} \\ \mathsf{AK}_{01} \oplus \mathsf{AG}_{01} \\ \mathsf{AK}_{10} \oplus \mathsf{AG}_{10} \\ \mathsf{AK}_{11} \oplus \mathsf{AG}_{11} \end{pmatrix} = \begin{pmatrix} \mathsf{OC}_{00} \\ \mathsf{OC}_{01} \\ \mathsf{OC}_{10} \\ \mathsf{OC}_{11} \end{pmatrix} = \mathbf{OC}$$

Since there are four main ciphertexts (each main ciphertext is κ bits) and four auxiliary ciphertexts (each auxiliary ciphertext is 1 bit), the total ciphertext length is $4(\kappa + 1)$.

For the evaluator, given input labels A_{λ_a} and B_{λ_b}, she can reconstruct the $(2\lambda_a + \lambda_b)$-th row of the main key $\mathsf{MK}_{2\lambda_a+\lambda_b}$ and auxiliary key $\mathsf{AK}_{2\lambda_a+\lambda_b}$. Then, she computes the output labels by decrypting the (λ_a, λ_b)'s main ciphertexts, as $\mathsf{MK}_{\lambda_a\lambda_b} \oplus \mathsf{MG}_{\lambda_a\lambda_b}$ and computes the output signal bit by decrypting the (i, j)'s auxiliary ciphertexts, as $\mathsf{AK}_{\lambda_a\lambda_b} \oplus \mathsf{AG}_{\lambda_a\lambda_b}$.

Row Reduction to three rows (GRR3). In the classical garbled circuit scheme, the garbler initially selects output labels for each wire, introducing two degrees of freedom. GRR3 removes one, effectively reducing the ciphertext length to $3(\kappa + 1)$.

The garbler generates a set of main and auxiliary keys similar to Yao's scheme. In the GRR3 scheme, instead of randomly selecting C_0 and the permutation bit c, the garbler assigns the first row of the main base as C_0 and the first row of the auxiliary base as the permutation bit c. Then, the garbler generates the output labels OL_{ij} and output color bits OC_{ij} as previews. Finally, the garbler constructs the main ciphertext $\mathbf{MG} = (0, \mathsf{MG}_1, \mathsf{MG}_2, \mathsf{MG}_3)$ to define the mapping MAP, and the auxiliary ciphertext $\mathbf{AG} = (0, \mathsf{AG}_1, \mathsf{AG}_2, \mathsf{AG}_3)$ to define the mapping $\widehat{\mathsf{MAP}}$. The mappings MAP and $\widehat{\mathsf{MAP}}$ are as follows:

$$\mathsf{MAP}(\mathbf{MK}, \mathbf{MG}) \Rightarrow \begin{pmatrix} \mathsf{MK}_{00} \oplus 0 \\ \mathsf{MK}_{01} \oplus \mathsf{MG}_{01} \\ \mathsf{MK}_{10} \oplus \mathsf{MG}_{10} \\ \mathsf{MK}_{11} \oplus \mathsf{MG}_{11} \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00} \\ \mathsf{OL}_{01} \\ \mathsf{OL}_{10} \\ \mathsf{OL}_{11} \end{pmatrix} = \mathbf{OL}$$

$$\widehat{\mathsf{MAP}}(\mathbf{AK}, \mathbf{AG}) \Rightarrow \begin{pmatrix} \mathsf{AK}_{00} \oplus 0 \\ \mathsf{AK}_{01} \oplus \mathsf{AG}_{01} \\ \mathsf{AK}_{10} \oplus \mathsf{AG}_{10} \\ \mathsf{AK}_{11} \oplus \mathsf{AG}_{11} \end{pmatrix} = \begin{pmatrix} \mathsf{OC}_{00} \\ \mathsf{OC}_{01} \\ \mathsf{OC}_{10} \\ \mathsf{OC}_{11} \end{pmatrix} = \mathbf{OC}$$

Since the first main ciphertext MG_{00} and the first auxiliary ciphertext AG_{00} is always 0, it need not be transmitted, reducing the ciphertext to $3(\kappa + 1)$ bits.

Row Reduction to two rows (GLNP-GRR2)[5]. In the GRR3 scheme above, the garbler set $C_0 = \mathsf{MK}_{00}$ to remove one freedom. Here, GLNP-GRR2 scheme removes another freedom by setting $C_1 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$. As for the

[5] This technique can only applied for AND gate

permutation bits, the garbler still chooses permutation bits randomly, which is the same as the classic Yao scheme. Specifically, the garbler selects the main ciphertext $\mathbf{MG} = (0, \mathsf{MG}_{01}, \mathsf{MG}_{10}, \mathsf{MG}_{01} \oplus \mathsf{MG}_{10})$ to finalize the mapping MAP and selects the auxiliary $\mathbf{AG} = (\mathsf{AG}_{00}, \mathsf{AG}_{01}, \mathsf{AG}_{10}, \mathsf{AG}_{11})$ to finalize the mapping $\widehat{\mathsf{MAP}}$. The mapping MAP and $\widehat{\mathsf{MAP}}$ are as follows:

$$\mathsf{MAP}(\mathbf{MK}, \mathbf{MG}) \Rightarrow \begin{pmatrix} \mathsf{MK}_{00} \oplus 0 \\ \mathsf{MK}_{01} \oplus \mathsf{MG}_{01} \\ \mathsf{MK}_{10} \oplus \mathsf{MG}_{10} \\ \mathsf{MK}_{11} \oplus \mathsf{MG}_{01} \oplus \mathsf{MG}_{10} \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00} \\ \mathsf{OL}_{01} \\ \mathsf{OL}_{10} \\ \mathsf{OL}_{11} \end{pmatrix} = \mathbf{OL}$$

$$\widehat{\mathsf{MAP}}(\mathbf{AK}, \mathbf{AG}) \Rightarrow \begin{pmatrix} \mathsf{AK}_{00} \oplus \mathsf{AG}_{00} \\ \mathsf{AK}_{01} \oplus \mathsf{AG}_{01} \\ \mathsf{AK}_{10} \oplus \mathsf{AG}_{10} \\ \mathsf{AK}_{11} \oplus \mathsf{AG}_{11} \end{pmatrix} = \begin{pmatrix} \mathsf{OC}_{00} \\ \mathsf{OC}_{01} \\ \mathsf{OC}_{10} \\ \mathsf{OC}_{11} \end{pmatrix} = \mathbf{OC}$$

Since there only need to be two main ciphertexts, every main ciphertext is κ bits. In addition, there are four auxiliary ciphertexts, and every auxiliary ciphertext is 1 bit. Thus, there are $2\kappa + 4$ bits.

Row Reduction to one row (GLNP-GRR1)[6]. The garbler generates a main keys and auxiliary keys for garbling the XOR gate, order by input color bits:

$$\begin{aligned} \mathsf{MK}_{00} &= F(A_0, 0)[1 \cdots n] \oplus F(B_0, 0)[1 \cdots n] & \mathsf{AK}_{00} &= 0 \oplus 0 \\ \mathsf{MK}_{01} &= F(A_0, 0)[1 \cdots n] \oplus F(B_1, 1)[1 \cdots n] & \mathsf{AK}_{01} &= 0 \oplus 1 \\ \mathsf{MK}_{10} &= F(A_1, 1)[1 \cdots n] \oplus F(B_0, 0)[1 \cdots n] & \mathsf{AK}_{10} &= 1 \oplus 0 \\ \mathsf{MK}_{11} &= F(A_1, 1)[1 \cdots n] \oplus F(B_1, 1)[1 \cdots n] & \mathsf{AK}_{11} &= 1 \oplus 1 \end{aligned}$$

Here, we still assume $\pi_a = 0$ and $\pi_B = 0$ such that the base's second and third row, where the color bits are (0, 1) and (1, 0), yields the output label C_1. The garbler set $C_0 = M_{00}$ and $C_1 = M_{10}$. Then, the garbler selects main ciphertext $\mathbf{MG} = (0, \mathsf{MG}_1, 0, \mathsf{MG}_1)$ to finalize the mapping MAP as follows:

$$\mathsf{MAP}(\mathbf{MK}, \mathbf{MG}) \Rightarrow \begin{pmatrix} \mathsf{MK}_{00} \oplus 0 \\ \mathsf{MK}_{01} \oplus \mathsf{MG}_{01} \\ \mathsf{MK}_{10} \oplus 0 \\ \mathsf{MK}_{11} \oplus \mathsf{MG}_{01} \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00} \\ \mathsf{OL}_{01} \\ \mathsf{OL}_{10} \\ \mathsf{OL}_{11} \end{pmatrix} = \mathbf{OL}$$

$$\widehat{\mathsf{MAP}}(\mathbf{AK}, \mathbf{AG}) \Rightarrow \begin{pmatrix} \mathsf{AK}_{00} \oplus 0 \\ \mathsf{AK}_{01} \oplus 0 \\ \mathsf{AK}_{10} \oplus 0 \\ \mathsf{AK}_{11} \oplus 0 \end{pmatrix} = \begin{pmatrix} \mathsf{OC}_{00} \\ \mathsf{OC}_{01} \\ \mathsf{OC}_{10} \\ \mathsf{OC}_{11} \end{pmatrix} = \mathbf{OC}$$

[6] This technique can only applied for XOR gate

Since there only needs to be one main ciphertext with κ bits. In addition, there is no auxiliary ciphertext. Thus, there will be κ bits.

Discuss on GLNP-GRR1 Gueron et al. proposed that $F(B_0, 0)$ can be directly replaced by B_0 for the main keys. In this case, $C_{\pi_a \oplus \pi_b} = F_1(A_0, 0) \oplus B_0$. When the evaluator has A_0 and B_1, the evaluator can compute $C_{\pi_a \oplus \pi_b} \oplus B_0 = F_1(A_0, 0)$. If these keys are used in another gate, then an evaluator (attacker) sees the XOR of two keys and the encryptions computed with each key separately. This is once again a related-key type assumption. In fact, during Gueron et al.'s proof, they did not consider the special case where the output label c and input label b are used again as input wires for other gates. If this special case is considered, when reducing to this gate, due to the two unknown labels having a relationship, it cannot be reduced to their 2PRF experiment (where the two PRF keys are independently random).

Observation and discuss. The above review found that these garble circuit schemes conform to a unified framework process. When garbling a circuit, the gates are processed in topological order. When a gate is processed, the labels of its input wires have already been determined, but the output labels may be determined due to garbing this gate. For each gate, the garbler calls the PRF to generate a set of main and auxiliary keys. These keys encrypt the output labels and color bits separately, generating the main and auxiliary ciphertext. This is done such that for each main key, decrypting the corresponding combination of main ciphers yields the correct output label, i.e., $(\mathsf{MK}_{ij} \oplus \widetilde{\mathsf{MG}}_{ij} = \mathsf{OL}_{ij})$for $i, j \in \{0, 1\}$, where $\widetilde{\mathsf{MG}}_{ij}$ is the linear combination of the main ciphers. Similarly, for each auxiliary key, decrypting the associated auxiliary ciphertexts produces the correct output signal bit, i.e., $(\mathsf{AK}_{ij} \oplus \widetilde{\mathsf{AG}}_{ij} = \mathsf{OC}_{ij})$for $i, j \in \{0, 1\}$, where $\widetilde{\mathsf{AG}}_{ij}$ is the linear combination of the auxiliary ciphers.

Then, we also observe the following properties common to existing garbling techniques under PRF assumption:

1. For any two wires (two input wires or one input wire and one output wire), the labels are independent and (pseudo)random to the evaluator.
2. For each gate, calls to the PRF are made statically. That is, neither Eval nor Garble ever uses the result of a PRF function to determine a future PRF function.
3. The Garble and Eval procedures only use *linear* operations apart from queries to the PRF function.

3.2 TreGates Garbling Scheme.

Recall that Rosulek and Roy use two novel techniques "slicing" and "dicing[7]" to move beyond the linear garbling model, and we wonder if these two techniques can optimize the PRF-based garbled circuits. Thus, we review the new techniques and analyze them from our perspective.

[7] We remark that "dicing" was first proposed by Kempka, Kikuchi and Suzuki in [10].

The slicing technique involves splitting wire labels into halves and using each half to compute a portion of the output label. For instance, in their scheme, the input labels and permute bits are written as $A_0\|0 = (A_0^L, A_0^R)$, $A_1\|1 = (A_1^L, A_1^R)$, $B_0\|0 = (B_0^L, B_0^R)$, and $B_1\|1 = (B_1^L, B_1^R)$. The output labels are written as $C_0\|\pi_c = (C_0^L, C_0^R)$ and $C_1\|\bar{\pi}_c = (C_1^L, C_1^R)$, where each half consists of $(\kappa+1)/2$ bits. They also employ a hash function H whose output is a $(\kappa+1)/2$-bit string, i.e., $H : \{0,1\}^* \to \{0,1\}^{(\kappa+1)/2}$. Based on the permute bit, the garbler parses the garbled circuit's output labels into $\mathsf{OL}_{00}^L, \mathsf{OL}_{01}^L, \mathsf{OL}_{10}^L, \mathsf{OL}_{11}^L$ and $\mathsf{OL}_{00}^R, \mathsf{OL}_{01}^R, \mathsf{OL}_{10}^R, \mathsf{OL}_{11}^R$. For $i, j \in \{0,1\}$, the evaluator can compute the output OL_{ij}^L and OL_{ij}^R using A_i, B_j, three $\kappa+1$-bit ciphertexts, and a linear function of A_i^L, A_i^R, B_j^L, and B_j^R.

However, the ciphertexts in this scheme rely on the permute bit, meaning that the generation of ciphertexts cannot be publicly determined as in previous schemes. Moreover, the evaluators computation is tied to the ciphertext generation process. TreGates addresses this issue using the "dicing" technique. In simple terms, dicing involves transmitting a small, constant-size ciphertext to guide the evaluator in computing the output label without revealing the garblers circuit generation process or exposing the permute bit.

In our view, roughly speaking, the "slicing" pertains to the introduction of two sets of keys, as detailed below (S_{ij}^v denotes a specific combination derived from the input labels A_i^L, A_i^R, B_j^L and B_j^R, where $i, j \in \{0,1\}$ and $v \in \{1,2\}$):

$$\overbrace{\begin{matrix} H(A_0) \oplus H(A_0 \oplus B_0) \oplus S_{00}^1 \\ H(A_0) \oplus H(A_0 \oplus B_1) \oplus S_{01}^1 \\ H(A_1) \oplus H(A_1 \oplus B_0) \oplus S_{10}^1 \\ H(A_1) \oplus H(A_1 \oplus B_1) \oplus S_{11}^1 \end{matrix}}^{\text{the first set of keys}} \qquad \overbrace{\begin{matrix} H(B_0) \oplus H(A_0 \oplus B_0) \oplus S_{00}^2 \\ H(B_1) \oplus H(A_0 \oplus B_1) \oplus S_{01}^2 \\ H(B_0) \oplus H(A_1 \oplus B_0) \oplus S_{10}^2 \\ H(B_1) \oplus H(A_1 \oplus B_1) \oplus S_{11}^2 \end{matrix}}^{\text{the second set of keys}}$$

In TreGates scheme, the garbler uses the first set of keys to generate the first half of the output label and the second set of keys for the second half. Specifically, these keys are combined using the three ciphertexts $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$ to produce both halves of the output label. As a result, the total ciphertext length is $3(\kappa+1)/2$. The garbler selects the ciphertexts $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$ and defines $\mathbf{G}^L = (0, \mathsf{G}_1, \mathsf{G}_1 \oplus \mathsf{G}_3, \mathsf{G}_3)$ and $\mathbf{G}^R = (0, \mathsf{G}_1 \oplus G_2, \mathsf{G}_1, \mathsf{G}_2)$ to finalize the mappings MAP^1 and MAP^2. These mappings, MAP^1 and MAP^2, are as follows:

$$\mathsf{MAP}^1(\mathsf{BK}^L, \mathbf{G}^L) \Rightarrow \begin{pmatrix} \mathsf{K}_{00}^L \oplus 0 \\ \mathsf{K}_{01}^L \oplus \mathsf{G}_1 \\ \mathsf{K}_{10}^L \oplus \mathsf{G}_1 \oplus \mathsf{G}_3 \\ \mathsf{K}_{11}^L \oplus \mathsf{G}_3 \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00}^L \\ \mathsf{OL}_{01}^L \\ \mathsf{OL}_{10}^L \\ \mathsf{OL}_{11}^L \end{pmatrix} = \mathbf{OL}^L$$

$$\mathsf{MAP}^2(\mathsf{BK}^R, \mathbf{G}^R) \Rightarrow \begin{pmatrix} \mathsf{K}_{00}^R \oplus 0 \\ \mathsf{K}_{01}^R \oplus \mathsf{G}_1 \oplus \mathsf{G}_2 \\ \mathsf{K}_{10}^R \oplus \mathsf{G}_1 \\ \mathsf{K}_{11}^R \oplus \mathsf{G}_2 \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00}^R \\ \mathsf{OL}_{01}^R \\ \mathsf{OL}_{10}^R \\ \mathsf{OL}_{11}^R \end{pmatrix} = \mathbf{OL}^R$$

In the previous scheme, each group of main keys required two ciphertexts. Now, let's explore why switching to two sets of main keys reduces the requirement to only three ciphertexts. The answer lies in the correlation between the two sets of keys. For example, when suitable S_{ij}^v are identified such that the XOR of the first and second rows of the left base matches the XOR of the first and third rows of the right base, namely

$$\begin{aligned} & H(A_0) \oplus H(A_0 \oplus B_0) \oplus S_{00}^1 \oplus H(A_0) \oplus H(A_0 \oplus B_1) \oplus S_{01}^1 \\ = \; & H(B_0) \oplus H(A_0 \oplus B_0) \oplus S_{00}^1 \oplus H(B_0) \oplus H(A_1 \oplus B_0) \oplus S_{10}^2 \end{aligned}$$

In simple terms, when the left and right groups of main keys are related, the ciphertext on the right can be obtained by linearly combining the two ciphertexts from the left. This reduces the overall number of ciphertexts needed.

However, the whole linear combination of input labels $S_{i,j}^k$ $(i, j, k \in \{0, 1\})$ depends on the permute bits. Therefore, revealing all linear combinations to the evaluator would compromise privacy. To address this, Rosulek and Roy employ a technique termed "dicing". In this approach, the garbler generates some constant-sized additional ciphertext. Given that the evaluator obtains input labels A_α and B_β, she can decrypt specific additional ciphertext through some random oracle queries with A_α, B_β and then obtain some control bits. Relying on these control bits, the evaluator can only determine the values of $S_{\alpha\beta}^1$ and $S_{\alpha\beta}^2$. This marginal $S_{\alpha\beta}^1$, $S_{\alpha\beta}^2$ view of the bases enables the evaluator to compute the output label while ensuring privacy.

Observation and discuss. Due to TreGates's use of slicing technology to combine two 64-bit computational results into a 127-bit label and a 1-bit color bit, it is impossible to directly separate the label and color bit for TreGates. Therefore, when constructing the model, we have two basic approaches. One approach is to apply the slicing technique only to the label while using a different method to garble the color bit. The other approach is to treat the label and color bit as a single entity and apply slicing to both. We adopt the first approach in our model to ensure it captures previous schemes and slicing and dicing technology as comprehensively as possible. We will also clarify in Appendix B that the second approach would yield the same conclusion.

4 Formalizing a Garbling Model under PRF assumption

Based on the observation from the previous section, we formalize our new model for garbling schemes under PRF assumption.

– Garbling algorithm Gb:
Parameterized by integers $s, m_1, n_A^0, n_A^1, n_B^0, n_B^1$, vectors $\{\mathbf{MK}_{ij}^{ab,u} | i,j,a,b \in \{0,1\}, u \in [s]\}$, PRF function $F_1(\cdot,\cdot)$, and mapping function MAP and matrix $\boldsymbol{\mathcal{P}}$. The length of each $\mathbf{MK}_{ij}^{ab,u}$ is $n_A^i + n_B^j + 2s$ where $i,j \in \{0,1\}$, with entries in $\mathrm{GF}(2^{\kappa/s})$. The PRF F_1 is mapped as $\{0,1\}^\kappa \times \{0,1\}^{\kappa+1+c} \to \{0,1\}^{\kappa/s}$. The MAP breaks down into two linear subfunction: $(\mathsf{MAP}_1, \mathsf{MAP}_2)$ where MAP_1 performs matrix mutiplication in $\mathrm{GF}(2^{\kappa/s})$, while MAP_2 carries out a bit-wise XOR operation. The matrix $\boldsymbol{\mathcal{P}}$ is $4s \times m_1$ in size with entries in $\mathrm{GF}(2^{\kappa/s})$.
Also parameterized by integers m_2, q, vectors $\{\mathbf{AK}_{ij} | i,j \in \{0,1\}\}$. PRF function $F_2(\cdot,\cdot)$ and mapping function $\widehat{\mathsf{MAP}}$ and matrix $\boldsymbol{\mathcal{X}}$. Vector $\mathbf{AK}_{ij}$ is of length $4+q$ with entries in $\mathrm{GF}(2)$. The PRF F_2 is mapped as $\{0,1\}^\kappa \times \{0,1\}^{\kappa+1+c} \to \{0,1\}$. The $\widehat{\mathsf{MAP}}$ breaks down into two linear sub-functions: $(\widehat{\mathsf{MAP}}_1, \widehat{\mathsf{MAP}}_2)$, where $\widehat{\mathsf{MAP}}_1$ performs matrix multiplication in $\mathrm{GF}(2)$, while $\widehat{\mathsf{MAP}}_2$ carries out a bit-wise XOR operation.
In addition, also parameterized by integers c, s, g, vectors $\{Y^{ab} | a,b \in \{0,1\}\}$, matrix $\boldsymbol{\mathcal{U}}$, PRF $F_3(\cdot,\cdot)$. The length of vector Y^{ab} is s, with entries in $\mathrm{GF}(2^c)$. The matrix size is $g \times (s+q)$, with entries in $\mathrm{GF}(2^c)$. The PRF function is mapped as $\{0,1\}^\kappa \times \{0,1\}^{\kappa+1+c} \to \{0,1\}c$

1. Choose random permute bits $\pi_a, \pi_b \leftarrow \{0,1\}$ for the two input wires.
2. Choose the input labels $(A_0, A_1, B_0, B_1) \leftarrow \mathrm{GF}(2^\kappa)$ randomly where A_a and B_b representing FALSE. Parse the labels A_i, B_i (where $i \in \{0,1\}$) into $A_{i,1}, \ldots, A_{i,s}$ and $B_{i,1}, \ldots, B_{i,s}$, where $A_{i,j}, B_{i,j} \in \mathrm{GF}(2^{k/s})$ and $j \in [s]$.
3. Make n_A^0, n_A^1, n_B^0, and n_B^1 queries to the PRF F_1, using A_0, A_1, B_0, and B_1 as the respective keys, with public values as inputs. Let $\mathbf{Q}_{A_0}, \mathbf{Q}_{A_1}$, $\mathbf{Q}_{B_0}$, $\mathbf{Q}_{B_1}$ denote the corresponding responses to these queries. Define $\mathbf{S}_{A_i} := (A_{i,1}, \ldots, A_{i,s}, Q_{A_i}^1, \ldots, Q_{A_i}^{n_A^i})$ and $\mathbf{S}_{B_i} := (B_{i,1}, \ldots, B_{i,s}, Q_{B_i}^1, \ldots, Q_{B_i}^{n_B^i})$ where $i \in \{0,1\}$. Finally, let $\mathbf{S}_{ij} = (\mathbf{S}_{A_i}, \mathbf{S}_{B_j})$.
4. For $u \in [s]$, $i,j \in \{0,1\}$, compute $\mathsf{MK}_{ij}^u = \langle \mathbf{MK}_{ij}^{ab,u}, \mathbf{S}_{ij} \rangle$. Then $\mathbf{MK}^u = (\mathsf{MK}_{00}^u, \mathsf{MK}_{01}^u, \mathsf{MK}_{10}^u, \mathsf{MK}_{11}^u)^\top$.
5. (Generate the main ciphertext) Find m_1 ciphertexts of length κ/s denoted by $\mathbf{MG}$, such that the following four conditions are satisfied:
   (a) $\mathsf{MAP}_1(\boldsymbol{\mathcal{P}}, \mathbf{MG}) = \boldsymbol{\mathcal{P}} \times \mathbf{MG}^T = \widetilde{\mathbf{MG}}^T$. Note that $\widetilde{\mathbf{MG}}^T$ has a size of $4s \times 1$, with entries in $\mathrm{GF}(2^{\kappa/s})$. We then split $\widetilde{\mathbf{MG}}^T$ into $\widetilde{\mathbf{MG}}^1, \ldots, \widetilde{\mathbf{MG}}^s$, where each has a size of 4×1.
   (b)

$$\mathsf{MAP}^u(\mathbf{MK}^u, \widetilde{\mathbf{MG}}^u) \Rightarrow \begin{pmatrix} \mathsf{MK}_{00}^u \oplus \widetilde{\mathsf{MG}}_{00}^u \\ \mathsf{MK}_{01}^u \oplus \widetilde{\mathsf{MG}}_{01}^u \\ \mathsf{MK}_{10}^u \oplus \widetilde{\mathsf{MG}}_{10}^u \\ \mathsf{MK}_{11}^u \oplus \widetilde{\mathsf{MG}}_{11}^u \end{pmatrix} = \begin{pmatrix} \mathsf{OL}_{00}^u \\ \mathsf{OL}_{01}^u \\ \mathsf{OL}_{10}^u \\ \mathsf{OL}_{11}^u \end{pmatrix} = \mathbf{OL}^u$$

(c)
- If the gate is AND gate:
  - $\mathsf{OL}^u_{\pi_a\pi_b} = \mathsf{OL}^u_{\bar{\pi}_a\pi_b} = \mathsf{OL}^u_{\pi_a\bar{\pi}_b}$
  - Let $\mathsf{OL}_{\pi_a\pi_b}$ as the output label C_0 and $\mathsf{OL}_{\bar{\pi}_a\bar{\pi}_b}$ as the output label C_1.
- If the gate is XOR gate:
  - $\mathsf{OL}^u_{\pi_a\pi_b} = \mathsf{OL}^u_{\bar{\pi}_a\bar{\pi}_b}$, $\mathsf{OL}^u_{\bar{\pi}_a\pi_b} = \mathsf{OL}^u_{\pi_a\bar{\pi}_b}$
  - Let $\mathsf{OL}_{\pi_a\pi_b}$ as the output label C_0 and $\mathsf{OL}_{\bar{\pi}_a\pi_b}$ as the output label C_1.

(d) For any $i, j \in \{0,1\}$, let $S_1 \subseteq \mathbf{Q}_{A_i}$, $S_2 \subseteq \mathbf{Q}_{B_j}$, $S_3 \subseteq \{A^1_i, B^1_j, C^1_k, \cdots, A^s_i, B^s_j, C^s_k, \mathsf{MG}_1, \cdots, \mathsf{MG}_{m_1}\}$ and $S_4 \in \{A^1_{\bar{i}}, B^1_{\bar{j}}, C^1_{\bar{k}}, \ldots, A^s_{\bar{i}}, B^s_{\bar{j}}, C^s_{\bar{k}}\}$, where $C_k = \mathsf{OL}_{ij}$ S_4 is non-empty. Then, the following holds: [8]

$$\sum_{Q_A \in S_1} Q_A \oplus \sum_{Q_B \in S_2} Q_B \oplus \sum_{u \in S_3} u \oplus \sum_{v \in S_4} v \neq 0$$

6. Make q_1 distinct queries to the PRF F_2, where the queries are determined by the input labels (used as the key for the PRF) and some additional public numbers (used as the input to the PRF). Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, and define $\mathbf{S}_1 := (0, 1, Q'_1, \ldots, Q'_q)$[9].
7. (Generate the auxiliary key). For $i, j \in \{0,1\}$, compute $\mathsf{AK}_{ij} = \langle \mathbf{AK}_{ij}, \mathbf{S}_1 \rangle$. Then $\mathbf{AK}^u = (\mathsf{AK}^u_{00}, \mathsf{AK}^u_{01}, \mathsf{AK}^u_{10}, \mathsf{AK}^u_{11})^\top$.
8. (Generate the auxiliary ciphertext). Find m_2 auxiliary ciphertexts such that the following three conditions are satisfied:

(a) $\mathsf{MAP}_1(\boldsymbol{\mathcal{X}}, \mathbf{AG}) = \boldsymbol{\mathcal{X}} \times \mathbf{AG}^\top = \widetilde{\mathbf{AG}}^\top$;

(b)

$$\mathsf{MAP}(\mathbf{AK}, \widetilde{\mathbf{AG}}) \Rightarrow \begin{pmatrix} \mathsf{AK}_{00} \oplus \widetilde{\mathsf{AG}} \\ \mathsf{AK}_{01} \oplus \widetilde{\mathsf{AG}} \\ \mathsf{AK}_{10} \oplus \widetilde{\mathsf{AG}} \\ \mathsf{AK}_{11} \oplus \widetilde{\mathsf{AG}} \end{pmatrix} = \begin{pmatrix} \mathsf{OC}_{00} \\ \mathsf{OC}_{01} \\ \mathsf{OC}_{10} \\ \mathsf{OC}_{11} \end{pmatrix} = \mathbf{OC};$$

(c)
- If the gate is AND gates:
  - $\mathsf{OC}_{ab} = \mathsf{OC}_{\bar{a}b} = \mathsf{OC}_{a\bar{b}} \neq \mathsf{OC}_{\bar{a}\bar{b}}$
  - Let OC_{ab} as the output permute bit c
- If the gate is XOR gate:
  - $\mathsf{OC}_{ab} = \mathsf{OC}_{\bar{a}\bar{b}} \neq \mathsf{OC}_{2\bar{a}+b} = \mathsf{OC}_{2a+\bar{b}}$.
  - Let OL_{ab} as the output permute bit c.

[8] In fact, this requirement indicates that for an evaluator with A_i and B_j, the evaluator cannot obtain the linear relationship between its unknown input label and the output label. This requirement is based on our observation that the labels on any two wires are independent and (pseudo)random to the evaluator.

[9] The values 0 and 1 are the color bits, which the evaluator may also use during the computation. Therefore, these two bits are included in $\mathbf{S}_1$.

9. Make q_2 distinct queries to the PRF F_3, which can be determined based on the input labels (as the key of PRF) and some additional public numbers (as the input of PRF). Let $Q_1'', \cdots, Q_q''$ denote the responses to these queries and define $\mathbf{S}_3 := (0, 1, Q_1'', \cdots, Q_q'')$.
10. (Generate of the additional ciphertext) Compute $\mathbf{E} := \boldsymbol{\mathcal{X}} \times (\mathbf{Y}^{ab} \| \mathbf{S}_3)$ as additional ciphertext.
    Then $\mathbf{G}$ and $\mathbf{E}$ constitute the whole ciphertext, with a total length of $\kappa/s \cdot m_1 + m_2 + g \cdot v$.

– Encoding algorithm Encode:
1. Given input $x_a, x_b \in \{0,1\}$, compute $\alpha := x_a \oplus a$ and $\beta := x_b \oplus b$, where a and b are previously selected permute bits. Then, output $A_\alpha \| \alpha$ and $B_\beta \| \beta$.

– Evaluation algorithm Eval:
Parameterized by integers q_1, q_2, q_3, s, v, g, vector $\{\mathbf{V}^u_{\alpha\beta} | \alpha, \beta \in \{0,1\}, u \in [s]\}$, $\{\mathbf{Y}_{\alpha\beta} | \alpha, \beta \in \{0,1\}\}$, $\{\mathbf{K}_{\alpha\beta} | \alpha, \beta \in \{0,1\}\}$, matrix $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{X}}$,PRF function $F_1(\cdot,\cdot)$, $F_2(\cdot,\cdot)$, $F_3(\cdot,\cdot)$ and mapping function MAP, $\widehat{\mathsf{MAP}}$. The vector $\mathbf{V}^u_{\alpha\beta}$ has length $2s + q_1$ with entries in $\mathrm{GF}(2^{\kappa/s})$, the vector $\mathbf{Y}$ has length $2 + q_2$ with entries in $\mathrm{GF}(2)$, and the vector $\mathbf{K}$ has length q_3 with entries in $\mathrm{GF}(2^v)$. The matrix $\boldsymbol{\mathcal{P}}$ has size $4s \times m_1$ with entries in $\mathrm{GF}(2^{k/s})$, and the matrix $\boldsymbol{\mathcal{X}}$ has size $4 \times m_2$ with entries in $\mathrm{GF}(2)$. The
MAP breaks down into two linear subfunction: $(\mathsf{MAP}_1, \mathsf{MAP}_2)$ where MAP_1 performs matrix mutiplication in $\mathrm{GF}(2^{\kappa/s})$, while MAP_2 carries out a bit-wise XOR operation and the $\widehat{\mathsf{MAP}}$ breaks down into two linear sub-functions: $(\widehat{\mathsf{MAP}}_1, \widehat{\mathsf{MAP}}_2)$, where $\widehat{\mathsf{MAP}}_1$ performs matrix multiplication in $\mathrm{GF}(2)$, while $\widehat{\mathsf{MAP}}_2$ carries out a bit-wise XOR operation.

1. The inputs are wire labels A_α, B_β, color bits α, β, the main ciphertext $\mathbf{MG}$, the auxiliary ciphertext $\mathbf{AG}$ and the additional ciphertext $\mathbf{E}$.
2. Make q_1, q_2, and q_3 distinct queries to the PRF functions F_1, F_2, and F_3, respectively, using keys A_α or B_β and public inputs. Let $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{Q}_3$ denote the responses.
3. Parse A_α, B_β as $A^1_\alpha, \ldots, A^s_\alpha$ and $B^1_\alpha, \ldots, B^s_\alpha$. Let $\mathbf{T}_1 = A^1_\alpha, \ldots, A^s_\alpha, B^1_\alpha, \ldots, B^s_\alpha, Q^1_1, \ldots, Q^{q_1}_1$, $\mathbf{T}_2 = 0, 1, Q^1_1, \ldots, Q^{q_1}_1$, $\mathbf{T}_3 = Q^1_1, \ldots, Q^{q_1}_1$
4. Compute $K_{\alpha\beta} = \langle \mathbf{K}_{\alpha\beta}, (\mathbf{E} \| \mathbf{T}_3) \rangle$ and parse it into w vectors, each with a length of $2s+q$ and entries in $\mathrm{GF}(2^{k/s})$, denote these vectors as $\{\hat{\mathbf{V}}^u_{\alpha\beta} | u \in [s]\}$.
5. For $u \in [s]$, compute $MK^u_{\alpha\beta} = \langle V^u_{\alpha\beta} + \hat{V}^u_{\alpha\beta}, \mathbf{T}_1 \rangle$ and compute $AK_{\alpha\beta} = \langle \mathbf{Y}_{\alpha\beta}, \mathbf{T}_2 \rangle$.
6. Compute $\widetilde{\mathbf{MG}}^T = \mathsf{MAP}_1(\boldsymbol{\mathcal{P}}, \mathbf{MG})$, and parse $\widetilde{\mathbf{MG}}^T$ into $\widetilde{\mathbf{MG}}^1, \ldots, \widetilde{\mathbf{MG}}^s$ where $\widetilde{\mathbf{MG}}^i = \{\widetilde{\mathsf{MG}}^i_{00}, \widetilde{\mathsf{MG}}^i_{01}, \widetilde{\mathsf{MG}}^i_{10}, \widetilde{\mathsf{MG}}^i_{11}\}$, then use $\mathsf{OL}^u_{\alpha\beta} = \mathsf{MK}^u_{\alpha\beta} \oplus \widetilde{\mathsf{MG}}^u_{\alpha\beta}$.
7. Compute $(\widetilde{\mathsf{AG}}_{00}, \widetilde{\mathsf{AG}}_{01}, \widetilde{\mathsf{AG}}_{10}, \widetilde{\mathsf{AG}}_{10}) = \mathsf{MAP}_1(\boldsymbol{\mathcal{X}}, \mathbf{AG})$, then use $\mathsf{OC}_{\alpha\beta} = \mathsf{AK}_{\alpha\beta} \oplus \widetilde{\mathsf{MG}}^u_{\alpha\beta}$.

Definition 3. *Let $\{x_1, x_2, \ldots, x_n\}$ be n Boolean string variables, where each $x_i \in \{0,1\}^k$, meaning each x_i is a Boolean string of length k. These n variables are said to be* ***linearly independent*** *if there does not exist a non-empty subset $S = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\}$ such that:*

$$x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_m} = v$$

where $v \in \{0,1\}^k$ is a public fixed Boolean string (e.g., the all-zero string $\mathbf{0}_k$ or another public value). If such a subset exists, these variables are said to be ***linearly dependent****.*

Definition 4. *Let $\{x_1, x_2, \ldots, x_m\}$ be a set of m Boolean string variables, where each $x_i \in \{0,1\}^k$, meaning each x_i is a Boolean string of length k. The set is said to have a* ***dimension*** *of n if there* ***exist*** *n Boolean strings that are linearly independent, but any subset of $n-1$ Boolean strings is linearly dependent.*

Definition 5. *We say that a garbling scheme has* ***ideal security***[10] *if no adversary (computationally unbounded, with bounded queries to a PRF/RF) has an advantage better than $poly(\kappa)/2^\kappa$ (rather than negligible) in the security games, where κ is the security parameter and key length of the PRF function.*

Lemma 1. *For any $u \in [s]$, the dimension of the u-th set of main keys $\boldsymbol{MK}^u$ is at least 3.*

Proof. Due to space limitations, this proof is provided in the Appendix C.1.

Lemma 2. *For any $u \in [s]$, the dimension of the u-th set of main keys $\boldsymbol{MK}^u$ for the AND gate is at least 4.*

Proof. Using a proof by contradiction, we show that if the dimension of the u-th set of main keys is not 4, the privacy of the garbled circuit will be compromised.

If the dimension of each set of main keys for the AND gate is not 4 (recall from Lemma 1 that it is at least 3), then it must hold that $\mathsf{MK}^u_{00} \oplus \mathsf{MK}^u_{01} \oplus \mathsf{MK}^u_{10} \oplus \mathsf{MK}^u_{11} = 0$. Review garble model step 8b) $\mathsf{AK}_{ij} \oplus \widetilde{\mathsf{AG}}_{ij} = \mathsf{OC}_{ij}$, then the evaluator can compute

$$\begin{aligned}
&C^u_0 \oplus C^u_1 \\
&= \mathsf{OL}^u_{00} \oplus \mathsf{OL}^u_{01} \oplus \mathsf{OL}^u_{10} \oplus \mathsf{OL}^u_{11} \\
&= \mathsf{MK}^u_{00} \oplus \widetilde{\mathsf{MG}}^u_{00} \oplus \mathsf{MK}^u_{01} \oplus \widetilde{\mathsf{MG}}^u_{01} \oplus \mathsf{MK}^u_{10} \oplus \widetilde{\mathsf{MG}}^u_{10} \oplus \mathsf{MK}^u_{11} \oplus \widetilde{\mathsf{MG}}^u_{11} \\
&= \widetilde{\mathsf{MG}}^u_{00} \oplus \widetilde{\mathsf{MG}}^u_{01} \oplus \widetilde{\mathsf{MG}}^u_{10} \oplus \widetilde{\mathsf{MG}}^u_{11}
\end{aligned}$$

This violate the privacy.

[10] This definition is adapted from Zahur, Rosulek, and Evans [21]. Consider setting the security parameter to $k - \log k$, which still satisfies the ideal security condition. However, when selecting a security parameter value for implementing a garbled circuit, such artificial reductions in the security parameter will inevitably weaken the concrete security of the scheme. Therefore, to prevent such reductions, we restrict the length of the labels (excluding the permute bit) and the PRF key to κ bits.

Lemma 3. *If the main keys* $\boldsymbol{MK}^1, \ldots, \boldsymbol{MK}^s$ *are linearly dependent, the garbling scheme for an AND gate cannot achieve ideal security.*

Proof. This proof is in Appendix C.2.

Lemma 4. *If the main keys* $\boldsymbol{MK}^1, \cdots, \boldsymbol{MK}^s$ *are linearly independent, any privacy garbling scheme for an AND gate conforming to our Model must satisfy* $|\mathsf{MG}| \geq 2k$.

Proof. Recall the step 6(b), for $u \in [s]$:

$$\begin{aligned}
\mathsf{MK}^u_{00} \oplus \widetilde{\mathsf{MG}}^u_{00} &= \mathsf{OL}^u_{00} \\
\mathsf{MK}^u_{01} \oplus \widetilde{\mathsf{MG}}^u_{01} &= \mathsf{OL}^u_{00} \\
\mathsf{MK}^u_{10} \oplus \widetilde{\mathsf{MG}}^u_{10} &= \mathsf{OL}^u_{00} \\
\mathsf{MK}^u_{11} \oplus \widetilde{\mathsf{MG}}^u_{11} &= \mathsf{OL}^u_{00}
\end{aligned}$$

We change the equations as

$$\begin{bmatrix} \mathsf{OL}^1_{00} \\ \mathsf{OL}^1_{01} \\ \mathsf{OL}^1_{10} \\ \mathsf{OL}^1_{11} \\ \vdots \\ \mathsf{OL}^s_{00} \\ \mathsf{OL}^s_{01} \\ \mathsf{OL}^s_{10} \\ \mathsf{OL}^s_{11} \end{bmatrix} \oplus (\boldsymbol{\mathcal{P}} \times \mathbf{MG}) = \begin{bmatrix} \mathsf{MK}^1_{00} \\ \mathsf{MK}^1_{01} \\ \mathsf{MK}^1_{10} \\ \mathsf{MK}^1_{11} \\ \vdots \\ \mathsf{MK}^s_{00} \\ \mathsf{MK}^s_{01} \\ \mathsf{MK}^s_{10} \\ \mathsf{MK}^s_{11} \end{bmatrix}$$

We change the equations as

$$T \times \begin{bmatrix} \mathsf{OL}^1_0 \\ \mathsf{OL}^1_1 \\ \vdots \\ \mathsf{OL}^s_0 \\ \mathsf{OL}^s_1 \\ \mathsf{MG}_0 \\ \vdots \\ \mathsf{MG}_m \end{bmatrix} = I_{4s} \times \begin{bmatrix} \mathsf{MK}^1_{00} \\ \mathsf{MK}^1_{01} \\ \mathsf{MK}^1_{10} \\ \mathsf{MK}^1_{11} \\ \vdots \\ \mathsf{MK}^s_{00} \\ \mathsf{MK}^s_{01} \\ \mathsf{MK}^s_{10} \\ \mathsf{MK}^s_{11} \end{bmatrix}$$

For the equation to be solvable, $\mathrm{Col}(T) \supseteq \mathrm{Col}(I_{4s})$. This implies that matrix T must have at least $4s$ columns. Therefore, the number of variables $\mathsf{OL}^1_0, \mathsf{OL}^1_1, \ldots, \mathsf{OL}^s_0, \mathsf{OL}^s_1, \mathsf{MG}_1, \ldots, \mathsf{MG}_m$ must be at least $4s$, which gives $m \geq 2s$. Since the length of each ciphertext is κ/s, the total length of all ciphertexts is at least $2s \times (\kappa/s) = 2\kappa$.

Lemma 5. *The auxiliary ciphertext* $\boldsymbol{AG}$ *consists of at least 2 bits for the AND gate.*

Proof. Point and permute enforces that the permute and color bit have a relationship $\lambda_c = \pi_c \oplus v_c$, which indicates that three of the output color bits OC_{00}, OC_{01}, OC_{10}, OC_{11} is equal to π_c and the other one is equal to $\bar{\pi}_c$.

First, we claim that the dimension of the auxiliary keys is at least 3. In other words, at least three auxiliary keys are linearly independent. Without loss of generality, assume that AK_{00}, AK_{01}, and AK_{11} are linearly independent. Additionally, suppose $p_a = p_b = 0$, then $\mathsf{OC}_{00} = \mathsf{OC}_{01} = \mathsf{OC}_{11} = \pi_c$. Recall step 8(b):

$$\begin{aligned}\mathsf{AK}_{00} \oplus \widetilde{\mathsf{AG}}_{00} &= \mathsf{OC}_{00},\\ \mathsf{AK}_{01} \oplus \widetilde{\mathsf{AG}}_{00} &= \mathsf{OC}_{01},\\ \mathsf{AK}_{11} \oplus \widetilde{\mathsf{AG}}_{00} &= \mathsf{OC}_{11}.\end{aligned}$$

Rearranging the above equations, we have:

$$\begin{aligned}\widetilde{\mathsf{AG}}_{00} &= \pi_c \oplus \mathsf{AK}_{00},\\ \widetilde{\mathsf{AG}}_{01} &= \pi_c \oplus \mathsf{AK}_{01},\\ \widetilde{\mathsf{AG}}_{11} &= \pi_c \oplus \mathsf{AK}_{11}.\end{aligned}$$

Thus, $\widetilde{\mathsf{AG}}_{00}$, $\widetilde{\mathsf{AG}}_{01}$, and $\widetilde{\mathsf{AG}}_{11}$ must be linearly independent. If $\mathbf{AG}$ consists of only one element AG (1 bit), then each $\widetilde{\mathsf{AG}}_{ij}$ would either be AG or 0, which contradicts the fact that $\widetilde{\mathsf{AG}}_{00}$, $\widetilde{\mathsf{AG}}_{01}$, and $\widetilde{\mathsf{AG}}_{11}$ are linearly independent. Therefore, the auxiliary ciphertext $\mathbf{AG}$ must consist of at least 2 bits for the AND gate.

Next, we prove our claim by assuming the dimension of the auxiliary keys is less than 3, meaning any three auxiliary keys are linearly dependent. This leads to two possible cases:

- Case 1: Two auxiliary keys have an XOR result equal to a public value (either 0 or 1).
- Case 2:Any three auxiliary keys have an XOR result equal to a public value.

For Case 1, without loss of generality, suppose $\mathsf{AK}_{00} \oplus \mathsf{AK}_{01} = \text{pub}$. In this scenario, when the evaluator possesses A_0 and B_0, they can compute not only OC_{00}, but also OC_{01}, thereby compromising privacy.

For Case 2, recall the equations:

$$\begin{aligned}\mathsf{AK}_{00} \oplus \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} &= \text{pub}_1,\\ \mathsf{AK}_{00} \oplus \mathsf{AK}_{01} \oplus \mathsf{AK}_{11} &= \text{pub}_2.\end{aligned}$$

By adding these two equations, we get:

$$\mathsf{AK}_{10} \oplus \mathsf{AK}_{11} = \text{pub}_1 \oplus \text{pub}_2.$$

This results in Case 1, which is impossible.
Thus, both cases are impossible, confirming that our claim is true.

Theorem 1. *Any ideally secure garbling scheme for an AND gate conforming to our model must satisfy* $|G| \geq 2k + 2$.

Proof. The result is easy to see from Lemma 3, Lemma 4 and Lemma 5.

Lemma 6. *The dimension of the main keys* $\mathsf{MK}_{ij}^{ab,u}$ *is at least* $3s$.

Proof. Due to space limitations, this proof is provided in the Appendix C.3.

Theorem 2. *Any ideally secure garbling scheme for an XOR gate conforming to Model 2 must satisfy* $|G| \geq k$.

Proof. Recall the step 6(b), for $u \in [s]$:

$$\begin{aligned}
\mathsf{MK}_{00}^{u} \oplus \widetilde{\mathsf{MG}}_{00}^{u} &= \mathsf{OL}_{00}^{u} \\
\mathsf{MK}_{01}^{u} \oplus \widetilde{\mathsf{MG}}_{01}^{u} &= \mathsf{OL}_{00}^{u} \\
\mathsf{MK}_{10}^{u} \oplus \widetilde{\mathsf{MG}}_{10}^{u} &= \mathsf{OL}_{00}^{u}
\end{aligned}$$

We change the equations as

$$\begin{bmatrix} \mathsf{OL}_{00}^{1} \\ \mathsf{OL}_{01}^{1} \\ \mathsf{OL}_{10}^{1} \\ \vdots \\ \mathsf{OL}_{00}^{s} \\ \mathsf{OL}_{01}^{s} \\ \mathsf{OL}_{10}^{s} \end{bmatrix} \oplus (\boldsymbol{\mathcal{P}} \times \mathbf{MG}) = \begin{bmatrix} \mathsf{MK}_{00}^{1} \\ \mathsf{MK}_{01}^{1} \\ \mathsf{MK}_{10}^{1} \\ \mathsf{MK}_{00}^{s} \\ \mathsf{MK}_{01}^{s} \\ \mathsf{MK}_{10}^{s} \end{bmatrix}$$

We change the equations as

$$T \times \begin{bmatrix} \mathsf{OL}_{0}^{1} \\ \mathsf{OL}_{1}^{1} \\ \vdots \\ \mathsf{OL}_{0}^{s} \\ \mathsf{OL}_{1}^{s} \\ \mathsf{MG}_{0} \\ \vdots \\ \mathsf{MG}_{m} \end{bmatrix} = I_{3s} \times \begin{bmatrix} \mathsf{MK}_{00}^{1} \\ \mathsf{MK}_{01}^{1} \\ \mathsf{MK}_{10}^{1} \\ \mathsf{MK}_{11}^{1} \\ \vdots \\ \mathsf{MK}_{00}^{s} \\ \mathsf{MK}_{01}^{s} \\ \mathsf{MK}_{10}^{s} \\ \mathsf{MK}_{11}^{s} \end{bmatrix}$$

For the equation to be solvable, $\mathrm{Col}(T) \supseteq \mathrm{Col}(I_{3s})$. This implies that matrix T must have at least $3s$ columns. Therefore, the number of variables $\mathsf{OL}_0^1, \mathsf{OL}_1^1, \ldots, \mathsf{OL}_0^s, \mathsf{OL}_1^s, \mathsf{MG}_1, \ldots, \mathsf{MG}_m$ must be at least $3s$, which gives $m \geq s$. Since the length of each ciphertext is k/s, the total length of all ciphertexts is at least $s \times (k/s) = k$.

Theorem 3. *Any ideally secure garbling scheme for an AND gate that follows our model must invoke the PRF F_1 at least six times.*

Proof. Proof outline: We first present two claims, and then prove the correctness of the theorem based on these two claims. Finally, we prove the correctness of the two claims.

Recall that

$$\begin{aligned}\mathsf{MK}_{ij} &= \langle \mathbf{MK}_{ij}^{ab,u}, \mathbf{S}_{ij} \rangle \\ &= \langle \mathbf{MK}_{ij}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{ij}^{ab,u,B}, \mathbf{S}_{B_j} \rangle\end{aligned}$$

Recall that

$$\begin{aligned}\mathsf{MK}_{ij} &= \langle \mathbf{MK}_{ij}^{ab,u}, \mathbf{S}_{ij} \rangle \\ &= \langle \mathbf{MK}_{ij}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{ij}^{ab,u,B}, \mathbf{S}_{B_j} \rangle\end{aligned}$$

Then, we claim that

1. The computation of MK_{ij} includes at least two PRF calls, and the keys for these two PRF calls are A_i and B_j, respectively.
2. $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ includes at least two PRF calls, and the keys for these PRF calls are either A_0, A_1 or B_0, B_1.

From claim 2, without loss of generality, we assume that the PRF calls involved in $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ use the keys A_0, A_1. From claim 1, we know that both MK_{00} and MK_{01} invoke $F_1(A_0, \cdot)$, and both MK_{10} and MK_{11} invoke $F_1(A_1, \cdot)$. Since the PRF calls in $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ use the keys A_0, A_1, the inputs to $F_1(A_0, \cdot)$ in MK_{00} and MK_{01} must differ, and similarly, the inputs to $F_1(A_1, \cdot)$ in MK_{10} and MK_{11} must differ as well. Therefore, at least two calls to $F_1(A_0, \cdot)$ and two calls to $F_1(A_1, \cdot)$ are made. Additionally, there is at least one call to $F_1(B_0, \cdot)$ and one call to $F_1(B_1, \cdot)$. Thus, at least six PRF calls to F_1 are made in total.

Due to page limits, the correctness of the two claims is refered to Appendix C.4

Theorem 4. *Any ideally secure garbling scheme for an XOR gate conforming to Model-1 calls PRF F_1 at least four times.*

Proof. Recalling Claim 1 in the proof of Theorem 3, the computation of MK_{ij} includes at least two PRF calls, and the keys for these two PRF calls are A_i and B_j, respectively. Therefore, the scheme includes at least the following PRF calls: $F_1(A_0, \mathsf{pub}_1)$, $F_1(A_1, \mathsf{pub}_2)$, $F_1(B_0, \mathsf{pub}_3)$, and $F_1(B_1, \mathsf{pub}_4)$.

5 Our scheme under PRF assumption

The state-of-the-art PRF-based garbling scheme is due to Gueron et al. [7], where AND gate costs 8 PRF calls and $2\kappa+4$ bits communication, and XOR gate costs

3 PRF calls and κ bits communication. While the XOR gate's computation and communication have already reached the lower bound of our model, the AND gate's computation and communication complexity have not yet achieved our model's lower bound. Here, we consider improving the AND gate further.

Firstly, we reconsider the construction of the main keys to improve the computation cost. Reviewing our proof of Theorem 3, we established the following two claims:

1. The equation consists of at least two PRF outputs.
2. The result of $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ consists of at least the key of these two PRF outputs are A_0, A_1 or B_0, B_1.

Based on these two claims, we construct the main keys as follows. The main keys need only 6 PRF calls and meet our computation lower bound.

$$\begin{aligned}
\mathsf{MK}_{00} &= F(A_0, 0)[1 \cdots n] \oplus F(B_0, 0)[1 \cdots n] \\
\mathsf{MK}_{01} &= F(A_0, 0)[1 \cdots n] \oplus F(B_1, 0)[1 \cdots n] \oplus F(A_0, 1)[1 \cdots n] \\
\mathsf{MK}_{10} &= F(A_1, 0)[1 \cdots n] \oplus F(B_0, 0)[1 \cdots n] \\
\mathsf{MK}_{11} &= F(A_1, 0)[1 \cdots n] \oplus F(B_1, 0)[1 \cdots n] \oplus F(A_1, 1)[1 \cdots n]
\end{aligned}$$

Without loss of generality, we consider $\pi_a = \pi_b = 0$, and we will consider the general scheme next subsection. Similar to GLNP-GRR2, we set $C_0 = \mathsf{MK}_{00}$ and $C_1 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ and set $\mathbf{MG} = (0, \mathsf{MG}_{01}, \mathsf{MG}_{10}, \mathsf{MG}_{01} \oplus \mathsf{MG}_{10})$. Then, we can compute

$$\begin{aligned}
\mathsf{MG}_{01} &= \mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \\
\mathsf{MG}_{10} &= \mathsf{MK}_{00} \oplus \mathsf{MK}_{10}
\end{aligned}$$

We can check for $\lambda_a = \lambda_b = 1$, the evaluator compute

$$\begin{aligned}
&\mathsf{MK}_{11} \oplus \mathsf{MG}_{01} \oplus \mathsf{MG}_{10} \\
&= \mathsf{MK}_{11} \oplus \mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{00} \oplus \mathsf{MK}_{10} \\
&= \mathsf{MK}_{11} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{01} \\
&= C_1
\end{aligned}$$

Then, we reconsider the construction of the auxiliary keys and ciphertext to improve the communication cost. Recall that we want to obtain the scheme that needs only two auxiliary ciphertexts. We set $\mathbf{AG} = (0, \mathsf{AG}_{01}, \mathsf{AG}_{10}, \mathsf{AG}_{01} \oplus \mathsf{AG}_{10})$. Recall that $\mathsf{OC}_{ij} = \mathsf{AK}_{ij} \oplus \widetilde{\mathsf{AG}}_{ij}$. Thus,

$$\begin{aligned}
&\mathsf{OC}_{00} \oplus \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \oplus \mathsf{OC}_{11} \\
&= \mathsf{AK}_{00} \oplus \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{11} \oplus \widetilde{\mathsf{AG}}_{00} \oplus \widetilde{\mathsf{AG}}_{01} \oplus \widetilde{\mathsf{AG}}_{10} \oplus \widetilde{\mathsf{AG}}_{11} \\
&= \mathsf{AK}_{00} \oplus \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}
\end{aligned}$$

Since three of $\mathsf{OC}_{ij} = \pi_c$ and one of $\mathsf{OC}_{ij} = \bar{\pi}_c$, thus $\mathsf{OC}_{00} \oplus \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \oplus \mathsf{OC}_{11} = 1$, Thus, $\mathsf{AK}_{00} \oplus \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{11} = 1$. Then, we set auxiliary keys as

$$\begin{aligned}
\mathsf{AK}_{00} &= \mathsf{lsb}(F(A_0, 0)) \oplus \mathsf{lsb}(F(B_0, 0)) \\
\mathsf{AK}_{01} &= \mathsf{lsb}(F(A_0, 0)) \oplus \mathsf{lsb}(F(B_1, 0)) \\
\mathsf{AK}_{10} &= \mathsf{lsb}(F(A_1, 0)) \oplus \mathsf{lsb}(F(B_0, 0)) \\
\mathsf{AK}_{11} &= \mathsf{lsb}(F(A_1, 0)) \oplus \mathsf{lsb}(F(B_1, 0)) \oplus 1
\end{aligned}$$

Then, set $\pi_c = \mathsf{AK}_{00}$. We can compute auxiliary ciphertext as follows:

$$\begin{aligned}
\mathsf{AG}_{01} &= \mathsf{AK}_{01} \oplus \mathsf{AK}_{00} \\
\mathsf{AG}_{10} &= \mathsf{AK}_{10} \oplus \mathsf{AK}_{00}
\end{aligned}$$

We can check for $\lambda_a = \lambda_b = 1$, the evaluator compute output color bit as

$$\begin{aligned}
&\mathsf{AK}_{11} \oplus \mathsf{AG}_{01} \oplus \mathsf{AG}_{10} \\
&= \mathsf{AK}_{11} \oplus \mathsf{AK}_{01} \oplus AB_{00} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{00} \\
&= \mathsf{AK}_{00} \oplus 1 \\
&= \bar{\pi}_c
\end{aligned}$$

5.1 Details of our scheme

notation. For wire i, w_i^0 and w_i^1 represent the labels, where w_i^0 corresponds to FALSE and w_i^1 to TRUE, while π_i denotes the permute bit and λ_i denotes the color bit.

In this section, we provide a full specification of our garbling scheme. This description uses our row reduction to two rows technique for the AND gate and the GLNP XOR gate for the XOR gate and GLNP NOR gate scheme. Our garbling scheme uses a pseudorandom function that takes an κ-bit key with input and output of length $\kappa + 1$. That is, $F : \{0,1\}^\kappa \times \{0,1\}^{\kappa+1} \to \{0,1\}^{\kappa+1}$ (formally, we consider a family of functions, where for every $\kappa \in \mathbb{N}$ the function is of this type). We denote by $F_k(x)[1 \ldots \kappa]$ the first κ bits of the output of $F_k(x)$, and we denote by $x \| y$ the concatenation of x with y. We define the method for garbling AND and XOR gates in Fig. 1 and Figs. 2 (for simplicity, we only consider XOR, AND, and NOT gates; the AND gate method can be extended to any odd gate type) and then proceed to the high-level garbling algorithm in Fig. 3. Finally, we describe the encoding, evaluation, and decoding algorithms in Fig. 4, Fig. 5, and Fig. 6.

5.2 Security of our scheme

In this section, we explain the security of our scheme. First, we demonstrate the correctness of our scheme, then we provide an intuitive explanation of why our scheme is secure, and finally, we present a detailed security proof.

Procedure $\mathsf{GbAND}(w_a^0, w_a^1, w_b^0, w_b^1, \pi_a, \pi_b)$:

1. Compute the PRF:

$$\tilde{w}_a^{\pi_a} = F_{w_a^{\pi_a}}(g\|0), \tilde{w}_a^{\bar{\pi}_a} = F_{w_a^{\bar{\pi}_a}}(g\|0)$$
$$\tilde{w}_b^{\pi_b} = F_{w_b^{\pi_b}}(g\|0), \tilde{w}_b^{\bar{\pi}_b} = F_{w_b^{\bar{\pi}_b}}(g\|0)$$
$$\hat{w}_a^{\pi_a} = F_{w_a^{\pi_a}}(g\|1), \hat{w}_a^{\bar{\pi}_a} = F_{w_a^{\bar{\pi}_a}}(g\|1)$$

2. Compute the main keys

$$\mathsf{MK}_{00} = (\tilde{w}_a^{\pi_a} \oplus \tilde{w}_b^{\pi_b})[1 \cdots \kappa]$$
$$\mathsf{MK}_{01} = (\tilde{w}_a^{\pi_a} \oplus \tilde{w}_b^{\bar{\pi}_b} \oplus \hat{w}_a^{\pi_a})[1 \cdots \kappa]$$
$$\mathsf{MK}_{10} = (\tilde{w}_a^{\bar{\pi}_a} \oplus \tilde{w}_b^{\pi_b})[1 \cdots \kappa]$$
$$\mathsf{MK}_{11} = (\tilde{w}_a^{\bar{\pi}_a} \oplus \tilde{w}_b^{\bar{\pi}_b} \oplus \hat{w}_a^{\bar{\pi}_a})[1 \cdots \kappa]$$

3. Compute the location of '1' in the truth table: $s = 2\pi_i + \pi_j$
4. Compute the output labels
   (a) If $s = 0$: $w_c^0 = \mathsf{MK}_{00}$ and $w_c^1 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$.
   (b) Else: $w_c^1 = \mathsf{MK}_{00}$ and $w_c^0 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$.
5. Compute MG_1 and MG_2:
   (a) If $s = 0$: $\mathsf{MG}_1 = \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$, $\mathsf{MG}_2 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{11}$
   (b) If $s = 1$: $\mathsf{MG}_1 = \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$, $\mathsf{MG}_2 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{10}$
   (c) If $s = 2$: $\mathsf{MG}_1 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{01}$, $\mathsf{MG}_2 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{11}$
   (d) If $s = 3$: $\mathsf{MG}_1 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{01}$, $\mathsf{MG}_2 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{10}$
6. Compute the auxiliary keys:

$$\mathsf{AK}_{00} = \mathsf{lsb}(\tilde{w}_a^{\pi_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\pi_b})$$
$$\mathsf{AK}_{01} = \mathsf{lsb}(\tilde{w}_a^{\pi_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\bar{\pi}_b})$$
$$\mathsf{AK}_{10} = \mathsf{lsb}(\tilde{w}_a^{\bar{\pi}_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\pi_b})$$
$$\mathsf{AK}_{11} = \mathsf{lsb}(\tilde{w}_a^{\bar{\pi}_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\bar{\pi}_b}) \oplus 1$$

7. Compute the output permute bit :
   – If $s = 0$: $\pi_c = \mathsf{AK}_{00}$; Else: $\pi_c = \mathsf{AK}_{00} \oplus 1$.
8. Compute the auxiliary ciphertexts:
   (a) If $s = 0$: $\mathsf{AG}_1 = \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}$, $\mathsf{AG}_2 = \mathsf{AK}_{01} \oplus \mathsf{AK}_{11}$
   (b) If $s = 1$: $\mathsf{AG}_1 = \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}$, $\mathsf{AG}_2 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{10}$
   (c) If $s = 2$: $\mathsf{AG}_1 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{01}$, $\mathsf{AG}_2 = \mathsf{AK}_{01} \oplus \mathsf{AK}_{11}$
   (d) If $s = 3$: $\mathsf{AG}_1 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{01}$, $\mathsf{AG}_2 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{10}$
9. Return $w_c^0, w_c^1, \pi_c, \mathsf{MG}_1, \mathsf{MG}_2, \mathsf{AG}_1, \mathsf{AG}_2$

Fig. 1: Garbling AND gate

Correctness. We begin by demonstrating correctness. The XOR and NOT gates follow the GLNP scheme, so we only need to prove the correctness of the

Procedure $\mathsf{GbXOR}(w_a^0, w_a^1, w_b^0, w_b^1, \pi_a, \pi_b)$:

1. Compute the main keys:

$$\mathsf{MK}_{00} = F_{w_a^{\pi_a}}(g\|0)[1\cdots\kappa] \oplus F_{w_b^{\pi_b}}(g\|0)[1\cdots\kappa]$$
$$\mathsf{MK}_{01} = F_{w_a^{\pi_a}}(g\|0)[1\cdots\kappa] \oplus F_{w_b^{\bar{\pi}_b}}(g\|1)[1\cdots\kappa]$$
$$\mathsf{MK}_{10} = F_{w_a^{\bar{\pi}_a}}(g\|1)[1\cdots\kappa] \oplus F_{w_b^{\pi_b}}(g\|0)[1\cdots\kappa]$$
$$\mathsf{MK}_{11} = F_{w_a^{\bar{\pi}_a}}(g\|1)[1\cdots\kappa] \oplus F_{w_b^{\bar{\pi}_b}}(g\|1)[1\cdots\kappa]$$

2. Compute the output labels:
   (a) If $\pi_a \oplus \pi_b = 0$: $w_c^0 = \mathsf{MK}_{00}$, $w_c^1 = \mathsf{MK}_{00} \oplus F_{w_a^{\pi_a}}(g\|0)[1\cdots\kappa] \oplus F_{w_a^{\bar{\pi}_a}}(g\|1)[1\cdots\kappa]$.
   (b) Else: $w_c^1 = \mathsf{MK}_{00}$, $w_c^0 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$.
3. Compute the output permute bit $\pi_c = \pi_a \oplus \pi_b$
4. Compute the main ciphertexts:
   – $\mathsf{MG} = w_c^0 \oplus w_c^1 \oplus \mathsf{MK}_{00} \oplus \mathsf{MK}_{01}$
5. Return $w_c^0, w_c^1, \pi_c, \mathsf{MG}$

Fig. 2: Garbling XOR gates

AND gate. We will demonstrate its correctness in two parts: first, by showing the correctness of the output label, and second, by showing the correctness of the output color bit.

Recall that $\mathsf{OL}_{00} = \mathsf{MK}_{00}$, $\mathsf{MG}_{01} = \mathsf{MK}_{01} \oplus \mathsf{OL}_{01}$, $\mathsf{MG}_{10} = \mathsf{MK}_{10} \oplus \mathsf{OL}_{10}$. This is immediate for the correctness of the output label when $(\lambda_a, \lambda_b) = (0,0), (0,1), (1,0)$. In addition, Recall that

$$\begin{aligned}&\mathsf{OL}_{00} \oplus \mathsf{OL}_{01} \oplus \mathsf{OL}_{10} \oplus \mathsf{OL}_{11}\\ &= w_c^0 \oplus w_c^1\\ &= \mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}\end{aligned}$$

Thus, $\mathsf{OL}_{01} \oplus \mathsf{OL}_{10} \oplus \mathsf{OL}_{11} = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$. In addition, When $\lambda_a = \lambda_b = 1$, the evaluator compute

$$\begin{aligned}&\mathsf{MK}_{11} \oplus \mathsf{MG}_{01} \oplus \mathsf{MG}_{10}\\ &= \mathsf{MK}_{11} \oplus \mathsf{MK}_{01} \oplus \mathsf{OL}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{OL}_{10}\\ &= \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11} \oplus \mathsf{OL}_{01} \oplus \mathsf{OL}_{10}\\ &= \mathsf{OL}_{01} \oplus \mathsf{OL}_{10} \oplus \mathsf{OL}_{11} \oplus \mathsf{OL}_{01} \oplus \mathsf{OL}_{10}\\ &= \mathsf{OL}_{11}\end{aligned}$$

Thus, the output label is computation correctly.

The garbling algorithm Garble$(1^n, c)$:

1. For each input wire j in c:
   (a) Choose two random keys: $w_j^0, w_j^1 \leftarrow \{0,1\}^\kappa$
   (b) Choose a permutation bit for the bit '0': $\pi_j \leftarrow \{0,1\}$
   (c) Prepare the encoding information: $e[j,0] := w_j^0 \parallel \pi_j$ and $e[j,1] := w_j^1 \parallel \overline{\pi_j}$
2. In topological order, for each gate g in circuit f:
   (a) If g is a XOR gate with input wires a, b and output wire c:
      i. $(w_c^0, w_c^1, \pi_c, \mathsf{MG}) \leftarrow \mathrm{GbXOR}(w_a^0, w_a^1, w_b^0, w_b^1, \pi_a, \pi_b)$
      ii. Set the keys on the output wire c to be w_c^0, w_c^1 and the permutation bit to be π_c
      iii. Set the garbled table for the gate: $G[g] := \mathsf{MG}$
   (b) If g is an AND gate with input wires a, b and output wire c:
      i. $(w_c^0, w_c^1, \pi_c, \mathsf{MG}_0, \mathsf{MG}_1, \mathsf{AG}_0, \mathsf{AG}_1) \leftarrow \mathrm{GbAND}(k_i^0, k_i^1, k_j^0, k_j^1, \pi_i, \pi_j)$
      ii. Set the keys on the output wire c to be w_c^0, w_c^1 and the permutation bit to be π_c
      iii. Set the garbled table for the gate: $C[g] := (\mathsf{MG}_0, \mathsf{MG}_1, \mathsf{AG}_0, \mathsf{AG}_1)$
   (c) If g is a NOT gate with input wire a and output wire c:
      i. Set $w_c^0 = w_a^1$ and $w_c^1 = w_a^0$, and set $\pi_c = \pi_i$
      ii. There is no garbled table
3. For each circuit-output wire j in c, prepare the decoding information: $d[j,0] := F_{w_j^{\pi_j}}(\mathrm{out} \| \pi_j)$ and $d[j,1] := F_{w_j^{\overline{\pi}_j}}(\mathrm{out} \| \overline{\pi}_j)$
4. Return (F, e, d)

Fig. 3: Full garbling algorithm

Procedure Encode(e, x):

1. For $i = 1$ to $|x|$: $X[i] := e[i, x_i]$
2. Return X

Fig. 4: Encoding algorithm

Then, we show the correctness of the output color bit. Recall that $\mathsf{OC}_{00} = \mathsf{AK}_{00}$, $\mathsf{AG}_{01} = \mathsf{AK}_{01} \oplus \mathsf{OC}_{01}$, $\mathsf{AG}_{10} = \mathsf{AK}_{10} \oplus \mathsf{OC}_{10}$. This is immediate for the correctness of the output color bit when $(\lambda_a, \lambda_b) = (0,0), (0,1), (1,0)$. In addition, Recall that

$$\begin{aligned}
&\mathsf{OC}_{00} \oplus \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \oplus \mathsf{OC}_{11} \\
&= \pi_c \oplus \bar{\pi}_c \\
&= \mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}
\end{aligned}$$

Procedure Eval(F, X):

1. For every input wire j in f, set $w_j \| \lambda_j := X[j]$
2. For each gate g in c, in topological order:
   (a) If g is a XOR gate with input wires i, j and output wire ℓ:
      i. Compute the main keys: $\mathsf{MK} = F_{w_a}(g\|\lambda_a)[1 \ldots \kappa] \oplus F_{w_b}(g\|\lambda_b)[1 \ldots \kappa]$
      ii. Compute the output label: $w_c = \mathsf{MK} \oplus \lambda_b \mathsf{MG}_{\lambda_a \lambda_b}$
      iii. Compute the output wire signal bit: $\lambda_c := \lambda_a \oplus \lambda_b$
   (b) If g is an AND gate with input wires a, b and output wire c:
      i. Compute the main keys: $\mathsf{MK} = F_{w_a}(g\|0)[1 \ldots \kappa] \oplus F_{w_b}(g\|0)[1 \ldots \kappa] \oplus \lambda_b F_{w_a}(g\|1)[1 \ldots \kappa]$
      ii. Compute the output label: $w_c = \lambda_a \mathsf{MK}_1 \oplus \lambda_b \mathsf{MK}_2 \oplus \lambda_b \mathsf{MG}_{\lambda_a \lambda_b}$
      iii. Compute the auxiliary keys: $\mathsf{AK} = \mathsf{lsb}(F_{w_a}(g\|0)) \oplus \mathsf{lsb}(F_{w_b}(g\|0)) \oplus \lambda_a \lambda_b$.
      iv. Compute the output color bit: $\lambda_c = \mathsf{AK} \oplus \lambda_a \mathsf{AG}_1 \oplus \lambda_b \mathsf{AG}_2$.
   (c) If g is a NOT gate with input wire a and output wire c, then set $k_c := k_a$ and $\lambda_c = \lambda_a$
3. For each output wire j in f, set $Y[j] := F_{w_j}(\text{out}\}\lambda_j)$
4. Return Y

Fig. 5: Evaluation algorithm

Procedure Decode(Y, d):

1. For $i = 1$ to $|Y|$:
   (a) If $Y[i] = d[i, 0]$, then $y[i] := 0$
   (b) Else, if $Y[i] = d[i, 1]$, then $y[i] := 1$
   (c) Else, return $\perp$
2. Return y

Fig. 6: Decoding algorithm

Thus, $\mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \oplus \mathsf{OC}_{11} = \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}$. In addition, When $\lambda_a = \lambda_b = 1$, the evaluator compute

$$\begin{aligned}
&\mathsf{AK}_{11} \oplus \mathsf{AG}_{01} \oplus \mathsf{AG}_{10} \\
&= \mathsf{AK}_{11} \oplus \mathsf{AK}_{01} \oplus \mathsf{OC}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{OC}_{10} \\
&= \mathsf{AK}_{01} \oplus \mathsf{AK}_{10} \oplus \mathsf{AK}_{11} \oplus \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \\
&= \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \oplus \mathsf{OC}_{11} \oplus \mathsf{OC}_{01} \oplus \mathsf{OC}_{10} \\
&= \mathsf{OC}_{11}
\end{aligned}$$

Thus, the output color bit is computation correctly.

Intuition for Security. The XOR and NOT gates follow the GLNP scheme, so we only need to prove the security of the AND gate. For any main key MK_{ij}(resp. auxiliary key AK_{ij}) of AND gate, this main key (resp. auxiliary key) can only be computed using wire labels w_a^i and w_b^j, and independent of other main keys (resp. auxiliary key). Thus, the scheme is security intuitively.

Theorem 5. *Our garbling scheme is secure (achieving privacy, obliviousness, and authenticity) if F is a family of pseudorandom functions.*

Proof. Due to page limits, the full proof is deferred to Appendix D.

References

1. ao, L.T.A.N.B., Peralta, R.: NIST First Call for Multi-Party Threshold Schemes (Initial Public Draft). NIST Internal Report, NIST IR 8214C ipd (2023), https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8214C.ipd.pdf
2. Barker, E., Roginsky, A., Davis, R.: Recommendation for Cryptographic Key Generation. NIST Special Publication 800-133, Revision 2 (2020), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
3. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: 2013 IEEE Symposium on Security and Privacy. pp. 478–492. IEEE Computer Society Press, Berkeley, CA, USA (May 19–22, 2013). https://doi.org/10.1109/SP.2013.39
4. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012: 19th Conference on Computer and Communications Security. pp. 784–796. ACM Press, Raleigh, NC, USA (Oct 16–18, 2012). https://doi.org/10.1145/2382196.2382279
5. Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.S.: On the security of the "free-XOR" technique. In: Cramer, R. (ed.) TCC 2012: 9th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 7194, pp. 39–53. Springer, Berlin, Heidelberg, Germany, Taormina, Sicily, Italy (Mar 19–21, 2012). https://doi.org/10.1007/978-3-642-28914-9_3
6. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 191–219. Springer, Berlin, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_7
7. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast garbling of circuits under standard assumptions. Journal of Cryptology **31**(3), 798–844 (Jul 2018). https://doi.org/10.1007/s00145-017-9271-y
8. Guo, C., Katz, J., Wang, X., Weng, C., Yu, Y.: Better concrete security for half-gates garbling (in the multi-instance setting). In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 793–822. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_28
9. Guo, C., Katz, J., Wang, X., Yu, Y.: Efficient and secure multiparty computation from fixed-key block ciphers. In: 2020 IEEE Symposium on Security and Privacy. pp. 825–841. IEEE Computer Society Press, San Francisco, CA, USA (May 18–21, 2020). https://doi.org/10.1109/SP40000.2020.00016

10. Kempka, C., Kikuchi, R., Suzuki, K.: How to circumvent the two-ciphertext lower bound for linear garbling schemes. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 967–997. Springer, Berlin, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53890-6_32
11. Kolesnikov, V., Mohassel, P., Rosulek, M.: FleXOR: Flexible garbling for XOR gates that beats free-XOR. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part II. Lecture Notes in Computer Science, vol. 8617, pp. 440–457. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014). https://doi.org/10.1007/978-3-662-44381-1_25
12. Kolesnikov, V., Rackoff, C.: Password mistyping in two-factor-authenticated key exchange. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II. Lecture Notes in Computer Science, vol. 5126, pp. 702–714. Springer, Berlin, Heidelberg, Germany, Reykjavik, Iceland (Jul 7–11, 2008). https://doi.org/10.1007/978-3-540-70583-3_57
13. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II. Lecture Notes in Computer Science, vol. 5126, pp. 486–498. Springer, Berlin, Heidelberg, Germany, Reykjavik, Iceland (Jul 7–11, 2008). https://doi.org/10.1007/978-3-540-70583-3_40
14. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology **22**(2), 161–188 (Apr 2009). https://doi.org/10.1007/s00145-008-9036-8
15. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce. pp. 129–139 (1999). https://doi.org/10.1145/336992.337028
16. National Institute of Standards and Technology: Advanced Encryption Standard (AES). NIST FIPS 197 (2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf
17. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 250–267. Springer, Berlin, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009). https://doi.org/10.1007/978-3-642-10366-7_15
18. Rosulek, M.: Improvements for gate-hiding garbled circuits. In: Patra, A., Smart, N.P. (eds.) Progress in Cryptology - INDOCRYPT 2017: 18th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 10698, pp. 325–345. Springer, Cham, Switzerland, Chennai, India (Dec 10–13, 2017). https://doi.org/10.1007/978-3-319-71667-1_17
19. Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part I. Lecture Notes in Computer Science, vol. 12825, pp. 94–124. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84242-0_5
20. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science. pp. 162–167. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986). https://doi.org/10.1109/SFCS.1986.25

21. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 220–250. Springer, Berlin, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_8

A Independence and Pseudorandomness of Wire Labels

In the proof framework of Y. Lindell and B. Pinkas, the simulator needs to forge the truth table of the garbled circuit such that the evaluator "sees" the correct row of the truth table. In contrast, the other three rows appear as independent random values. The approach involves using a real PRF to generate the row the evaluator "sees," while a truly random function f generates the other three rows. In the reduction proof, the indistinguishability between the "real truth table" and the "forged truth table" is reduced to the indistinguishability between a pseudorandom function and a truly random function. However, in the indistinguishability game between a pseudorandom function and a truly random function, the pseudorandom function must be selected independently from the key space. Therefore, to complete this reduction proof, the labels unknown to the evaluator must be independent and random relative to those known to the evaluator.

B Equivalence of Alternative Model

Recalling the proof of Theorem 1: if the main keys of different groups are correlated, privacy is leaked; if they are uncorrelated, at least $2s$ ciphertexts are needed, each with a length of κ/s. Thus, transmitting the labels requires at least 2κ bits of ciphertext. Additionally, it has been shown that transmitting the color bit requires at least 2 bits of ciphertext, so a total of at least $2\kappa+2$ bits of ciphertext are needed. If the label and permute bit are treated as a single entity, each ciphertext needs to be of length κ/s. If the keys from different groups are correlated, privacy is leaked; if they are uncorrelated, at least $2s$ ciphertexts are needed, each with a length of $(\kappa+1)/s$, meaning the communication cost remains at least $2\kappa+2$ bits of ciphertext (and there's no need to separately transmit the color bit in this case).

C Supplemental Materials for Section 4

C.1 Proof of Lemma 1

Proof. Recall that the vector $\mathbf{MK}_{ij}^{ab,u}$ has length $n_A^i + n_B^j + 2s$, with elements in $\mathrm{GF}(2^{k/s})$. We parse it into two parts: the first part consists of the first $n_A^i + s$ elements and is denoted by $\mathbf{MK}_{ij}^{ab,u,A}$, while the second part consists of the remaining $n_B^j + s$ elements and is denoted by $\mathbf{MK}_{ij}^{ab,u,B}$.

Recall that

$$\begin{aligned}\mathsf{MK}_{ij}^{u} &= \langle \mathbf{MK}_{ij}^{ab,u}, \mathbf{S}_{ij} \rangle \\ &= \langle \mathbf{MK}_{ij}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{ij}^{ab,u,B}, \mathbf{S}_{B_j} \rangle\end{aligned}$$

First, we claim that neither $\mathbf{MK}_{ij}^{ab,u,A}$ nor $\mathbf{MK}_{ij}^{ab,u,B}$ can be zero vectors. Then, based on this claim, we can verify that any three main keys are linearly independent. That is, the XOR of any one element, any two elements, or any three elements cannot result in a publicly fixed value. Here, we demonstrate that the XOR of any three elements cannot result in a publicly fixed value. Similarly, the cases of any two elements and any single element can be derived in the same way. Without loss of generality, we choose the three elements as MK_{00}^{u}, MK_{01}^{u}, and MK_{10}^{u}. Then

$$\begin{aligned}&\mathsf{MK}_{00}^{u} \oplus \mathsf{MK}_{00}^{u} \oplus \mathsf{MK}_{00}^{u} \\ &= \langle \mathbf{MK}_{00}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{00}^{ab,u,B}, \mathbf{S}_{B_j} \rangle \oplus \langle \mathbf{MK}_{01}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{01}^{ab,u,B}, \mathbf{S}_{B_j} \rangle \\ &\oplus \langle \mathbf{MK}_{10}^{ab,u,A}, \mathbf{S}_{A_i} \rangle \oplus \langle \mathbf{MK}_{10}^{ab,u,B}, \mathbf{S}_{B_j} \rangle \\ &= \langle \mathbf{MK}_{00}^{ab,u,A} + \mathbf{MK}_{01}^{ab,u,A}, \mathbf{S}_{A_0} \rangle \oplus \langle \mathbf{MK}_{00}^{ab,u,B} + \mathbf{MK}_{10}^{ab,u,B}, \mathbf{S}_{B_0} \rangle \\ &\oplus \langle \mathbf{MK}_{10}^{ab,u,A}, \mathbf{S}_{A_1} \rangle \oplus \langle \mathbf{MK}_{10}^{ab,u,B}, \mathbf{S}_{B_1} \rangle\end{aligned}$$

Note that $\mathbf{S}_{A_0}$ and $\mathbf{S}_{A_1}$, $\mathbf{S}_{B_0}$ and $\mathbf{S}_{B_1}$ are mutually independent and random. Since both $\mathbf{MK}_{10}^{ab,u,A}$ and $\mathbf{MK}_{10}^{ab,u,B}$ are non-zero vectors, the expression cannot result in a publicly fixed value.

Then, we demonstrate the intuitive correctness of our claim. Without loss of generality, we assume $\mathbf{MK}_{ij}^{ab,u,A}$ are zero vectors. In this case, MK_{ij}^{u} could be computed using only B_j without involving A_i.

Note that $\mathbf{MK}_{i\bar{j}}^{ab,u,A}$ only depends on the secret values of the permutation bits π_a and π_b. Therefore, the evaluator has a $1/4$ chance of correctly guessing $\mathbf{MK}_{ij}^{ab,u,A}$. If the evaluator possesses A_i and $B_{\bar{j}}$, they would have a $1/4$ chance of successfully guessing $\mathbf{MK}_{ij}^{ab,u,A}$ and subsequently computing MK_{ij}^{u}, which would compromise the security of the garbled circuit.

C.2 Proof of Lemma 3

Before proving this lemma, we first establish the following Lemma:

Lemma 7. *For any* $S \subseteq \{C_0^1, C_1^1, \ldots, C_0^s, C_1^s\}$ *and* $S \neq \emptyset$, $\bigoplus_{x \in S} x \neq \mathsf{pub}$

Proof. We divide S into two subsets. S_0 consists of all elements in S with a subscript of 0, i.e., $S_0 = \{C_0^i \mid C_0^i \in S\}$, and S_1 consists of all elements in S with a subscript of 1, i.e., $S_1 = \{C_1^i \mid C_1^i \in S\}$.

$$\bigoplus_{x \in S} x = \bigoplus_{x \in S_1} x \oplus \bigoplus_{x \in S_2} x$$

Recall that the evaluator has one label for every wire. We denote the evaluator has label C_v on wire c. If $\bigoplus_{x\in S} x = \mathsf{pub}$, then the evaluator can compute

$$\bigoplus_{x\in S_{\bar{v}}} x = \bigoplus_{x\in S_v} x \oplus \mathsf{pub}$$

Because the evaluator has label C_v, the evaluator has all the values in S_v. Then, $\bigoplus_{x\in S_{\bar{c}}} x = pub'$ for evaluator. Assume $S_{\bar{c}}$ has n elements. Then, the evaluator can guess the first $n-1$ element and compute the last element. Then, the evaluator can guess the $C_{\bar{v}}$ with probability $poly(\kappa)/(2^{(s-1)\kappa/s})$ rather than $poly(k)/2^{\kappa}$.

Proof. Then, we prove Lemma 3 by contradiction. If the main keys are linearly dependent, then for some $S \subseteq \{(i,j,u) \mid i,j \in \{0,1\}, u \in [s]\}, \bigoplus_{(i,j,c)\in S} \mathsf{MK}^c_{ij} = 0$.

We divide S into s smaller sets S_c, where each $S_c = \{(i,j,c) \in S\}$.

Recall that

$$\mathsf{OL}^c_{ij} = \mathsf{MK}^c_{ij} \oplus \widetilde{\mathsf{MG}}^c_{ij}$$

Thus

$$\begin{aligned}
\bigoplus_{(i,j,c)\in S_i, c\in[s]} \mathsf{OL}^c_{ij} &= \bigoplus_{(i,j,c)\in S} \mathsf{OL}^c_{ij} \\
&= \bigoplus_{(i,j,c)\in S} (\mathsf{MK}^c_{ij} \oplus \widetilde{\mathsf{MG}}^c_{ij}) \\
&= \bigoplus_{(i,j,c)\in S} \mathsf{MK}^c_{ij} \oplus \bigoplus_{(i,j,c)\in S} \widetilde{\mathsf{MG}}^c_{ij} \\
&= \bigoplus_{(i,j,c)\in S} \widetilde{\mathsf{MG}}^c_{ij}
\end{aligned}$$

Recall that OL^c_{ij} is either C^c_0 or C^c_1. Thus, $\bigoplus_{(i,j,c)\in S_i, c\in[s]} \mathsf{OL}^c_{ij} = 0$ according to Lemma 7. In other words, for any S_i, S_i is either empty or contains exactly two elements, both of which correspond to C^i_0.

Next, we prove that any S_i must be empty. Therefore, there are no main keys that are linearly independent.

Assume there exist some sets S_i that contain two elements, both of which correspond to C^i_0. Note that for the sets $\mathsf{OL}_{00}, \mathsf{OL}_{01}, \mathsf{OL}_{10}, \mathsf{OL}_{11}$, there is one that equals C^i_1 and three that equal C^i_0. Without loss of generality, we assume OL_{11} corresponds to C^i_1. Thus, S_i can only include two elements from $\mathsf{MK}_{00}, \mathsf{MK}_{01}, \mathsf{MK}_{10}$.

$$
\begin{aligned}
&\bigoplus_{(i,j,c)\in S} \mathsf{MK}_{ij}^{c} \\
&= \bigoplus_{c\in S_{00}} \mathsf{MK}_{00}^{c} \oplus \bigoplus_{c\in S_{01}} \mathsf{MK}_{01}^{c} \oplus \bigoplus_{c\in S_{10}} \mathsf{MK}_{10}^{c} \\
&= \bigoplus_{c\in S_{00}} \langle \mathbf{MK}_{00}^{ab,c}, S_{00}\rangle \oplus \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c}, S_{01}\rangle \oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c}, S_{10}\rangle \\
&= \bigoplus_{c\in S_{00}} \langle \mathbf{MK}_{00}^{ab,c,A}, S_{A_0}\rangle \oplus \bigoplus_{c\in S_{00}} \langle \mathbf{MK}_{00}^{ab,c,B}, S_{B_0}\rangle \\
&\oplus \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c,A}, S_{A_0}\rangle \oplus \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c,B}, S_{B_1}\rangle \\
&\oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c,A}, S_{A_1}\rangle \oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c,B}, S_{B_0}\rangle \\
&= \langle \bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,A} \oplus \bigoplus_{c\in S_{01}} \mathbf{MK}_{01}^{ab,c,A}, S_{A_0}\rangle \oplus \langle \bigoplus_{c\in S_{10}} \mathbf{MK}_{10}^{ab,c,A}, S_{A_1}\rangle \\
&\oplus \langle \bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,B} \oplus \bigoplus_{c\in S_{01}} \mathbf{MK}_{10}^{ab,c,B}, S_{B_0}\rangle \oplus \langle \bigoplus_{c\in S_{01}} \mathbf{MK}_{01}^{ab,c,B}, S_{B_1}\rangle
\end{aligned}
$$

Since S_{A_0}, S_{A_1}, S_{B_0}, and S_{B_1} are mutually independent, $\bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,A} \oplus \bigoplus_{c\in S_{01}} \mathbf{MK}_{01}^{ab,c,A}$, $\bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,B} \oplus \bigoplus_{c\in S_{10}} \mathbf{MK}_{10}^{ab,c,B}$, $\bigoplus_{c\in S_{01}} \mathbf{MK}_{10}^{ab,c,A}$ and $\bigoplus_{c\in S_{01}} \mathbf{MK}_{01}^{ab,c,B}$ must all be 0.

Then, we prove S_{00}, S_{01}, S_{10}, S_{11} are all empty.

$$
\begin{aligned}
&\bigoplus_{c\in S_{01}} \mathsf{MK}_{10}^{c} \\
&= \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{10}^{ab,c,A}, A_1\rangle \oplus \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{10}^{ab,c,B}, B_0\rangle \\
&= \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{10}^{ab,c,B}, A_1\rangle
\end{aligned}
$$

Then, if evaluator has A_1 and B_1, then he can have relationship on MK_{10}, which can help evauator guess MK_{10}, violating privacy. Similarly, S_{10} is also empty.

$$
\begin{aligned}
&\bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,A} \oplus \bigoplus_{c\in S_{01}} \mathbf{MK}_{10}^{ab,c,A} \\
&= \bigoplus_{c\in S_{00}} \mathbf{MK}_{00}^{ab,c,A}
\end{aligned}
$$

Thus, S_{00} is also empty. Thus, S is empty, too. Thus, the main keys are not linearly dependent.

C.3 Proof of Lemma 6

Proof. For every u, the dimension of MK_{ij}^{u} is 3 from Lemma 1, Then, we denote MK_{00}^{u}, MK_{01}^{u}, MK_{10}^{u} as the base of u-th main keys. If these bases are linearly independent, then the dimension of the main keys MK_{ij}^{u} is $3s$.

Then, we show these bases are linearly independent by contradiction. Suppose these bases are linearly dependent, then for some $S \in \{(i,j,u)|i,j \in \{(0,0),(0,1),(1,0)\}, u \in [s]\}$, such that $\bigoplus_{(i,j,c)\in S} \mathsf{MK}_{ij}^{c} = 0$. Recall that

$$\mathsf{MK}_{ij}^{c} \oplus \widetilde{\mathsf{MG}}_{ij}^{c} = \mathsf{OL}_{ij}^{c}$$

Thus

$$\begin{aligned}
\bigoplus_{(i,j,c)\in S_i, c\in[s]} \mathsf{OL}_{ij}^{c} &= \bigoplus_{(i,j,c)\in S} \mathsf{OL}_{ij}^{c} \\
&= \bigoplus_{(i,j,c)\in S} (\mathsf{MK}_{ij}^{c} \oplus \widetilde{\mathsf{MG}}_{ij}^{c}) \\
&= \bigoplus_{(i,j,c)\in S} \mathsf{MK}_{ij}^{c} \oplus \bigoplus_{(i,j,c)\in S} \widetilde{\mathsf{MG}}_{ij}^{c} \\
&= \bigoplus_{(i,j,c)\in S} \widetilde{\mathsf{MG}}_{ij}^{c}
\end{aligned}$$

Recall that OL_{ij}^{c} is either C_0^c or C_1^c. Thus, $\bigoplus_{(i,j,c)\in S_i, c\in[s]} \mathsf{OL}_{ij}^{c} = 0$ according to Lemma 5. In other words, for any S_i, S_i is either empty or contains exactly both OL_{01} and OL_{01}

Next, we prove that any S_i must be empty. Therefore, there are no main keys that are correlated.

$$\begin{aligned}
&\bigoplus_{(i,j,c)\in S} \mathsf{MK}_{ij}^{c} \\
&= \bigoplus_{c\in S_{01}} \mathsf{MK}_{ij}^{c} \oplus \bigoplus_{c\in S_{10}} \mathsf{MK}_{ij}^{c} \\
&= \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c}, S_{01} \rangle \oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c}, S_{10} \rangle \\
&= \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c,A}, S_{A_0} \rangle \oplus \bigoplus_{c\in S_{01}} \langle \mathbf{MK}_{01}^{ab,c,B}, S_{B_1} \rangle \\
&\oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c,A}, S_{A_1} \rangle \oplus \bigoplus_{c\in S_{10}} \langle \mathbf{MK}_{10}^{ab,c,B}, S_{B_0} \rangle
\end{aligned}$$

Since S_{A_0}, S_{A_1}, S_{B_0}, and S_{B_1} are mutually independent. Then, S_{01}, S_{10} are all empty.

C.4 The correctness of claims in Theorem 3

These are the two claims that need to be proven.

1. The computation of MK_{ij} includes at least two PRF calls, and the keys for these two PRF calls are A_i and B_j, respectively.
2. $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ includes at least two PRF calls, and the keys for these PRF calls are either A_0, A_1 or B_0, B_1.

First, we prove claim 1 using a proof by contradiction. Assume there exists MK_{ij} where its computation involves only one PRF call. Without loss of generality, we assume the key for this PRF is A_i. Recalling the proof of Lemma 1, $\mathbf{MK}_{ij}^{ab,u,B}$ cannot be the zero vector. Thus, $\mathsf{MK}_{ij}^{u} = \langle \mathbf{MK}_{ij}^{ab,u,A}, S_{A_i} \rangle \oplus \oplus_{x \in S} B_j^x$ where $S \subseteq \{1, \cdots, s\}$ and $S \neq \emptyset$. Additionally, let us consider the relationship between OL_{ij}^{u} and $\mathsf{OL}_{i\bar{j}}^{u}$. There are two cases to consider.

- Case 1: $\mathsf{OL}_{ij}^{u} = \mathsf{OL}_{i\bar{j}}^{u} = C_0^u$. In this case, an evaluator with A_i and B_j can compute $AAAA$. Since the evaluator also possesses A_i and $B_{\bar{j}}$, it can compute MK_{ij}^{u}. Moreover, $\mathbf{MK}_{ij}^{ab,u,A}$ depends on the permute bits π_a and π_b, meaning the evaluator has a $1/4$ chance of guessing $\mathbf{MK}_{ij}^{ab,u,A}$ correctly. In other words, the evaluator has a $1/4$ probability of obtaining a linear relation of $B_i^1, \cdots, B_i^s$. This would result in a loss of perfect security for the scheme.
- Case 2: If $\mathsf{OL}_{ij}^{u} \neq \mathsf{OL}_{i\bar{j}}^{u}$, then $\mathsf{OL}_{\bar{i}j}^{u} = \mathsf{OL}_{\bar{i}\bar{j}}^{u} = C_0^u$. From the analysis in Case 1, it follows that both $\mathsf{MK}_{\bar{i}j}^{u}$ and $\mathsf{MK}_{\bar{i}\bar{j}}^{u}$ must involve at least two PRF calls, which would lead to the evaluator's computation leaking information about the output label. For example, if the evaluator knows that MK_{ij} only involves one PRF call, they would infer that $\mathsf{OL}_{\bar{i}j}^{u} = \mathsf{OL}_{\bar{i}\bar{j}}^{u} = C_0^u$. This would result in compromising privacy.

Next, we show the correctness of claim 2. Recall that $\mathsf{OL}_{ij}^{u} = \mathsf{MK}_{ij}^{u} \oplus \widetilde{\mathsf{MG}}_{ij}^{u}$, thus

$$
\begin{aligned}
& C_0^u \oplus C_1^u \\
&= \mathsf{OL}_{00}^{u} \oplus \mathsf{OL}_{01}^{u} \oplus \mathsf{OL}_{10}^{u} \oplus \mathsf{OL}_{11}^{u} \\
&= \mathsf{MK}_{00}^{u} \oplus \widetilde{\mathsf{MG}}_{00}^{u} \oplus \mathsf{MK}_{01}^{u} \oplus \widetilde{\mathsf{MG}}_{01}^{u} \oplus \mathsf{MK}_{10}^{u} \oplus \widetilde{\mathsf{MG}}_{10}^{u} \oplus \mathsf{MK}_{11}^{u} \oplus \widetilde{\mathsf{MG}}_{11}^{u} \\
&= \mathsf{MK}_{00}^{u} \oplus \mathsf{MK}_{01}^{u} \oplus \mathsf{MK}_{10}^{u} \oplus \mathsf{MK}_{11}^{u} \oplus \widetilde{\mathsf{MG}}_{00}^{u} \oplus \widetilde{\mathsf{MG}}_{01}^{u} \oplus \widetilde{\mathsf{MG}}_{10}^{u} \oplus \widetilde{\mathsf{MG}}_{11}^{u}
\end{aligned}
$$

If $\mathsf{MK}_{00} \oplus \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$ only consist of one PRF calls. Then, it violate the requirement in step 5(d). Thus claim 2 is correct.

D Security Proof of Our Garbling Scheme

Our proof follows the high-level structure of Gueron et al. [7], with modifications as needed for our garbling scheme.

At first, we overview the experiment, called $\mathsf{2PRF}$ defined in [7]. In this experiment, the distinguisher/adversary is given access to four oracles divided into two pairs. The second and fourth oracles are always pseudorandom functions F_{k_1} and F_{k_2}, respectively. In contrast, the first and third oracles are either the same pseudorandom functions F_{k_1} and F_{k_2}, respectively, or independent truly

Exptment $\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}}(n,\sigma)$

1. Choose random keys $k_1, k_2 \leftarrow \{0,1\}^n$ for the pseudorandom function, and choose two truly random functions f^1, f^2. If $\sigma = 0$, set $(\mathcal{O}^1, \mathcal{O}^2, \mathcal{O}^3, \mathcal{O}^4) = (F_{k_1}, F_{k_1}, F_{k_2}, F_{k_2})$; else set $(\mathcal{O}^1, \mathcal{O}^2, \mathcal{O}^3, \mathcal{O}^4) = (f^1, F_{k_1}, f^2, F_{k_2})$
2. The adversary $\mathcal{A}$ is invoked upon input 1^n
3. When $\mathcal{A}$ makes a query j, x to its oracles with $j \in \{1,2,3,4\}$ and $x \in \{0,1\}^{n+1}$, answer as follows:
   - if $j \in \{1,2\}$ and x was already queried to $\{1,2\}\backslash\{j\}$, return $\perp$.
   - if $j \in \{3,4\}$ and x was already queried to $\{3,4\}\backslash\{j\}$, return $\perp$.
   - Otherwise, return $\mathcal{O}^j(x)$
4. $\mathcal{A}$ outputs a bit σ', and this is the output of the experiment.

Fig. 7: 2PRF experiment

random functions f_1 and f_3. If $\mathcal{A}$ can make the same query to the first and second oracle or the third and fourth oracle, then it can easily distinguish the cases. The security requirement is that it cannot distinguish the cases as long as it does not make such queries. The experiment is formally defined in Fig. 7, and 2PRF security is formalized in Definition 6.

Definition 6. *Let $\mathcal{F} = \{F_n\}_{n\in\{\mathbb{N}\}}$ be an efficient family of functions where for every n, $F_n : \{0,1\}^n \times \{0,1\}^{n+1} \to \{0,1\}^{n+1}$. Family $\mathcal{F}$ is a* 2PRF *if for every probabilistic polynomial time adversary $\mathcal{A}$, there exists a negligible function aa such that for every n*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}}(n,1)=1] - \Pr[\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}}(n,0)=1]\right| \leq u$$

The following lemma of Shay et al. [7] shows that the pseudorandomness of F_k is sufficient for it to be 2PRF as well.

Lemma 8. *If $\mathcal{F}$ is a family of pseudorandom functions, then it is a* 2PRF.

We begin by proving that our garbling scheme achieves privacy. Let $\mathcal{G}$ denote our garbling scheme. Our proof follows the high-level structure of [7], with modifications as needed for our garbling scheme.

Theorem 6. *If $\mathcal{F}$ is a family of pseudorandom functions, then the garbling scheme $\mathcal{G}$ achieves privacy.*

Proof. We begin by describing a simulator $\mathcal{S}$ for the privacy experiment. $\mathcal{S}$ is invoked with input $(1^n, f, f(x))$ and works as follows. As we will show, $\mathcal{S}$ will define an active label on every wire. This label will be the one that is obtained in the evaluation procedure. The other label is not active and is actually never explicitly defined. Rather, all the ciphertexts in the gates that are not "decrypted" in the evaluation are chosen at random.

1. For each input wire j in circuit f:
   (a) Choose an active label: $w_j \xleftarrow{\$} \{0,1\}^n$
   (b) Choose an active signal bit $\lambda_j \xleftarrow{\$} \{0,1\}$
   (c) Prepare the garbled input data: $X[j] = w_j \| \lambda_j$
2. In topological order, for each gate g in c:
   (a) If g is a XOR gate with input wires a, b and output wire c:
      i. Compute the active output wire signal bit: $\lambda_c = \lambda_a \oplus \lambda_b$
      ii. Compute a translated new label for wire a: $\tilde{w}_a = F_{w_a}(g \| \lambda_a)$
      iii. Compute a translated new label for wire b, set the garbled table and compute the output label:
         – $\tilde{w}_b = w_b$, set $C[g] \leftarrow \{0,1\}^n$ and set $w_c = w_a \oplus w_b \oplus \lambda_b T$.
   (b) If g is a AND gate with input wires a, b and output wire c:
      i. Choose the main ciphertext $\mathsf{MG}_1 \xleftarrow{\$} \{0,1\}^\kappa$, $\mathsf{MG}_2 \xleftarrow{\$} \{0,1\}^\kappa$.
      ii. Compute the main key $\mathsf{MK} = F_{w_a}(g \| 0)[1 \ldots \kappa] \oplus F_{w_b}(g \| 0)[1 \ldots \kappa] \oplus \lambda_b F_{w_a}(g \| 1)[1 \ldots \kappa]$.
      iii. Compute the output labels $w_c = \mathsf{MK} \oplus \lambda_a \mathsf{MG}_1 \oplus \lambda_b \mathsf{MG}_2$
      iv. Compute the auxiliary keys: $\mathsf{AK} = \mathsf{lsb}(F_{w_a}(g \| 0)) \oplus \mathsf{lsb}(F_{w_b}(g \| 0)) \oplus (\lambda_a \dot{\lambda}_b)$
      v. Compute the output color bit $\lambda_c = \mathsf{AK} \oplus (\lambda_a \cdot \mathsf{AG}_1) \oplus (\lambda_b \cdot \mathsf{AG}_2)$.
3. For each output wire j in c:
   (a) Prepare the decoding information: $d[j, f(x)_j] = F_{w_j}(out \| \lambda_j)$ and $d[j, \overline{f(x)_j}] \xleftarrow{\$} \{0,1\}^n$.
4. return (C, X, d).

We now show that the simulated garbled circuit is indistinguishable from a real garbled circuit by reduction to the $\mathsf{2PRF}$ experiment, which by Lemma 8 follows merely from the fact that F is a pseudorandom function. Let $\mathcal{A}$ be a probabilistic polynomial time adversary for privacy, and let g denote the number of gates in the circuit. We define a hybrid distribution $H_i(f, x)$ with $0 \leq i \leq g$ as the triple (F, X, d) generated in the following way (note that the procedure for generating $H_i(f, x)$ is given the circuit f and the real input x):

– *Garbling of gates*: The garbled circuit F is generated by garbling the first i gates in the topological order using the simulator garbling procedure, while gates $i+1, \cdots, g$ are garbled using the real garbling scheme. Observe that the simulator generates only a single label per wire; specifically, it generates the active label $w_j^{v_j}$. Therefore, the first step in the hybrid is to choose an additional label $w_j^{\overline{v}_j}$ for every wire j, which is the input wire of a gate that is garbled according to the real scheme.
– *Encoding information* X: For each circuit input wire j that enters a gate g, if g is garbled using the real scheme (i.e., the gate's index $> i$), then $X[j]$ is the garbled value that was chosen to represent the jth bit of the input (recall that in experiment Expt^{priv}, the adversary knows the input string x and so can choose the correct encoding for x). Otherwise, if g is garbled using the simulator procedure (i.e., the gates index $\leq i$), then $X[j]$ is the garbled value chosen for that wire's active label.

– *Decoding information* d: For each output wire j that exits from a gate g, if g was garbled using the real scheme, then there are two garbled values on a wire j, and $d[j,\cdot]$ is generated exactly as in the Garble procedure. Otherwise, if g is garbled using the simulator instructions, then there is only one garbled value on j and $d[j,]$ is generated exactly as in the simulator procedure.

Note that the hybrid $H_0(x)$ is a real garbled circuit (and is distributed as (C, X, d) in $\mathsf{Expt}^{\mathsf{priv}}_{\mathcal{G},\mathcal{A},\mathcal{S}}$ in the case that $\beta = 0$), while $H_\ell(x)$ is the output of the simulator $\mathcal{S}$ (and is distributed as (C, X, d) in $\mathsf{Expt}^{\mathsf{priv}}_{\mathcal{G},\mathcal{A},\mathcal{S}}$ in the case that $\beta = 1$). Next, for each $0 \le i \le \ell$, we define $\mathcal{A}_i$ to be a probabilistic polynomial time adversary for $\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}_i}(n,\sigma)$ experiment. $\mathcal{A}_i$ is given access to four oracles:

$$(\mathcal{O}^1(\cdot),\mathcal{O}^2(\cdot),\mathcal{O}^3(\cdot),\mathcal{O}^4(\cdot)) = (f^1(\cdot) \text{ or } F_{w_1}(\cdot), F_{w_1}(\cdot), f^2(\cdot) \text{ or } F_{w_2}(\cdot), F_{w_2}(\cdot))$$

Adversary $\mathcal{A}_i$ runs $\mathsf{Expt}^{\mathsf{priv}}$ with adversary $\mathcal{A}$. First, it invokes $\mathcal{A}$ and receives (f, x). Then, as we will see, it constructs a garbled circuit, which will either be distributed according to H_{i-1} or H_i, depending on the oracles it received. Thus, as we will show, if A can succeed in Expt^{priv} with the probability that is non-negligibly greater than $1/2$, then A_i will distinguish in the 2PRF experiment with non-negligible probability.

Formally, adversary $\mathcal{A}_i$ constructs a garbled circuit by generating the first $i-1$ gates in topological order using the simulator procedure and generating the gates indexed by $i+1, \cdots, m$ using the real Garble instructions (with subroutines GbXOR and GbAND). However, for the ith gate, $\mathcal{A}_i$ will use its oracles to generate a garbled table that is garbled as in the real scheme or as in the simulator code, depending on whether it received an oracle access to pseudorandom or to random functions. Assume the input wires of the ith gate are a, b, and the output wire is c. In addition, assume that the active labels on the input wires are associated with the bits v_a, v_b (recall that $\mathcal{A}_i$ knows the input to the circuit and thus v_a, v_b are known to it). Knowing w_a^v and w_b^v, adversary $\mathcal{A}$ will (implicitly) use the secrets k_1, k_2 that were chosen for the pseudorandom function in $\mathsf{Expt}^{\mathsf{2PRF}}$ as $w_a^{\overline{v}_a}$ and $w_b^{\overline{v}_b}$, respectively. Thus, whenever $\mathcal{A}_i$ needs to compute $F_{w_a^{\overline{v}_a}}(x)$ or $F_{w_b^{\overline{v}_b}}(x)$ for some x, it will send x to its oracles $\mathcal{O}^1$ or $\mathcal{O}^3$, respectively. (recall that these are either also F_{w_1}, F_{w_2}, or are random functions f^1, f^2). We remark that $\mathcal{O}^2$ and $\mathcal{O}^4$ are used to garble gates $\ell > i$ that use wires a, b as well; this will be described after we present the method for garbling the i th gate. Since the XOR gate follows the GLNP-GRR1 scheme, we only need to consider the case where the i-th gate is an AND gate.

As before, for wire a and b, $\mathcal{A}_i$ has two labels w_a^v, w_b^v, two signal bits λ_a, λ_b and the bits v_a, v_b that are on the wires. Then, it does the following:

1. Compute the values $\tilde{w}_a^{v_a} = F_{w_a^{v_a}}(g\|0\lambda_a)$, $\tilde{w}_a^{\overline{v}_a} = \mathcal{O}^1(g\|0\overline{\lambda}_a)$, $\tilde{w}_b^{v_b} = F_{w_b^{v_b}}(g\|0\lambda_b)$, $\tilde{w}_b^{\overline{v}_b} = \mathcal{O}^3(g\|0\overline{\lambda}_b)$, $\hat{w}_a^{v_a} = F_{w_a^{v_a}}(g\|1\lambda_a)[1\cdots n]\|\lambda_a$, $\hat{w}_a^{\overline{v}_a} = \mathcal{O}^1(g\|1\overline{\lambda}_a)[1\cdots n]\|\overline{\lambda}_a$.
2. Compute the offset $R = \hat{w}_a^{v_a} \oplus \hat{w}_a^{\overline{v}_a}$.
3. $T_G = \tilde{w}_a^{v_a} \oplus \tilde{w}_a^{\overline{v}_a} \oplus (v_b \oplus \lambda_b) \cdot R$, $T_E = \tilde{w}_b^{v_b} \oplus \tilde{w}_b^{\overline{v}_b} \oplus \overline{v}_a \hat{w}_a^{v_a} \oplus v_a \hat{w}_a^{\overline{v}_a}$

4. $W_G^0 = \tilde{W}_a^0 \oplus (v_a \oplus \lambda_a)T_G$, $W_E^0 = \tilde{W}_b^0 \oplus (\lambda_b \oplus v_b)(T_E \oplus \hat{W}_a^0)$
5. $w_c^0 \| \pi_c = W_G^0 \oplus W_E^0$, $w_c^1 = w_c^0 \oplus R[1 \cdots n]$

1. Compute the permute bit $\pi_a = v_a \oplus \lambda_a$, $\pi_b = v_b \oplus \lambda_b$.
2. Compute the PRF:

$$\begin{aligned}
\tilde{w}_a^{v_a} &= F_{w_a^{v_a}}(g\|0), \tilde{w}_a^{\bar{v}_a} = f(g\|0) \\
\tilde{w}_b^{v_b} &= F_{w_b^{v_b}}(g\|0), \tilde{w}_b^{\bar{v}_b} = f(g\|0) \\
\hat{w}_a^{v_a} &= F_{w_a^{v_a}}(g\|1), \hat{w}_a^{\bar{v}_a} = f(g\|1)
\end{aligned}$$

3. Set $\tilde{w}_a^{\pi_a}$, $\tilde{w}_a^{\bar{\pi}_a}$, $\hat{w}_a^{\pi_a}$, $\hat{w}_a^{\bar{\pi}_a}$
   - If $\lambda_a = 0$: $\tilde{w}_a^{\pi_a} = \tilde{w}_a^{v_a}$, $\tilde{w}_a^{\bar{\pi}_a} = \tilde{w}_a^{\bar{v}_a}$, $\hat{w}_a^{\pi_a} = \hat{w}_a^{\pi_a}$, $\hat{w}_a^{\bar{\pi}_a} = \hat{w}_a^{\bar{v}_a}$.
   - Else: $\tilde{w}_a^{\pi_a} = \tilde{w}_a^{\bar{v}_a}$, $\tilde{w}_a^{\bar{\pi}_a} = \tilde{w}_a^{v_a}$, $\hat{w}_a^{\pi_a} = \hat{w}_a^{\bar{\pi}_a}$, $\hat{w}_a^{\bar{\pi}_a} = \hat{w}_a^{v_a}$.
4. Set $\tilde{w}_b^{\pi_b}$, $\tilde{w}_b^{\bar{\pi}_b}$
   - If $\lambda_b = 0$: $\tilde{w}_b^{\pi_b} = \tilde{w}_b^{v_b}$, $\tilde{w}_b^{\bar{\pi}_b} = \tilde{w}_b^{\bar{v}_b}$.
   - Else: $\tilde{w}_b^{\pi_b} = \tilde{w}_b^{\bar{v}_b}$, $\tilde{w}_b^{\bar{\pi}_b} = \tilde{w}_b^{v_b}$.
5. Compute the main keys

$$\begin{aligned}
\mathsf{MK}_{00} &= (\tilde{w}_a^{\pi_a} \oplus \tilde{w}_b^{\pi_b})[1 \cdots n] \\
\mathsf{MK}_{01} &= (\tilde{w}_a^{\pi_a} \oplus \tilde{w}_b^{\bar{\pi}_b} \oplus \hat{w}_a^{\pi_a})[1 \cdots n] \\
\mathsf{MK}_{10} &= (\tilde{w}_a^{\bar{\pi}_a} \oplus \tilde{w}_b^{\pi_b})[1 \cdots n] \\
\mathsf{MK}_{11} &= (\tilde{w}_a^{\bar{\pi}_a} \oplus \tilde{w}_b^{\bar{\pi}_b} \oplus \hat{w}_a^{\bar{\pi}_a})[1 \cdots n]
\end{aligned}$$

6. Compute the location of '1' in the truth table: $s = 2\pi_i + \pi_j$
7. Compute the output labels
   (a) If $s = 0$: $w_c^0 = \mathsf{MK}_{00}$ and $w_c^1 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$.
   (b) Else: $w_c^1 = \mathsf{MK}_{00}$ and $w_c^0 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$.
8. Compute MG_1 and MG_2:
   (a) If $s = 0$: $\mathsf{MG}_1 = \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$, $\mathsf{MG}_2 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{11}$
   (b) If $s = 1$: $\mathsf{MG}_1 = \mathsf{MK}_{10} \oplus \mathsf{MK}_{11}$, $\mathsf{MG}_2 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{10}$
   (c) If $s = 2$: $\mathsf{MG}_1 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{01}$, $\mathsf{MG}_2 = \mathsf{MK}_{01} \oplus \mathsf{MK}_{11}$
   (d) If $s = 3$: $\mathsf{MG}_1 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{01}$, $\mathsf{MG}_2 = \mathsf{MK}_{00} \oplus \mathsf{MK}_{10}$
9. Compute the auxiliary keys:

$$\begin{aligned}
\mathsf{AK}_{00} &= \mathsf{lsb}(\tilde{w}_a^{\pi_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\pi_b}) \\
\mathsf{AK}_{01} &= \mathsf{lsb}(\tilde{w}_a^{\pi_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\bar{\pi}_b}) \\
\mathsf{AK}_{10} &= \mathsf{lsb}(\tilde{w}_a^{\bar{\pi}_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\pi_b}) \\
\mathsf{AK}_{11} &= \mathsf{lsb}(\tilde{w}_a^{\bar{\pi}_a}) \oplus \mathsf{lsb}(\tilde{w}_b^{\bar{\pi}_b}) \oplus 1
\end{aligned}$$

10. Compute the output permute bit :
    - If $s = 0$: $\pi_c = \mathsf{AK}_{00}$; Else: $\pi_c = \mathsf{AK}_{00} \oplus 1$.
11. Compute the auxiliary ciphertexts:
    (a) If $s = 0$: $\mathsf{AG}_1 = \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}$, $\mathsf{AG}_2 = \mathsf{AK}_{01} \oplus \mathsf{AK}_{11}$
    (b) If $s = 1$: $\mathsf{AG}_1 = \mathsf{AK}_{10} \oplus \mathsf{AK}_{11}$, $\mathsf{AG}_2 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{10}$
    (c) If $s = 2$: $\mathsf{AG}_1 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{01}$, $\mathsf{AG}_2 = \mathsf{AK}_{01} \oplus \mathsf{AK}_{11}$

(d) If $s = 3$: $\mathsf{AG}_1 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{01}$, $\mathsf{AG}_2 = \mathsf{AK}_{00} \oplus \mathsf{AK}_{10}$

As in the previous case, when $\sigma = 0$, it's easy to see that the code is identical to the real garbling scheme. In step 2, compute the offset $R = \hat{w}_a^0 \oplus \hat{w}_a^1$. In step 3, compute the $T_G = \tilde{w}_a^0 \oplus \tilde{w}_a^1 \oplus (\pi_b \cdot R)$ and $T_E = \tilde{w}_b^0 \oplus \tilde{w}_b^1 \oplus \hat{w}_a^0$. In step 4, compute the

When $\sigma = 1$, the answers of the oracles are random strings, and therefore, all the rows in the garbled table are random as well. Additionally, apart from replacing $F_{w_a^{v_a}}$ and $F_{w_b^{v_b}}$ with random functions, the rest of the code remains unchanged. Therefore, the evaluation function of the real scheme is still valid. Moreover, since the simulator is simulated based on the evaluator, it can be verified that the code and the simulator are consistent when $\sigma = 1$.

We conclude that when $\sigma = 0$, the ith gate is garbled as in the real garbling scheme, while when $\sigma = 1$, the ith gate is garbled as in the simulator procedure. However, to complete the garbled circuit construction, $\mathcal{A}_i$ needs to construct all the gates $\ell > i$. For a gate $\ell > i$, with input wires that are output from the ith gate and greater, $\mathcal{A}_i$ has both labels on the wires and can compute the gate just like in the real garbling procedure. If a gate $\ell > i$ has an input wire that is output from a gate $j < i$ that does not equal a or b, then $\mathcal{A}_i$ chooses the (inactive) label at random, like in the hybrid definition.

If a gate $\ell > i$ has an input wire that is a, then $\mathcal{A}_i$ just computes the gate just like in the real garbling procedure but using oracle $\mathcal{O}^2$ when $F_{w_a^{\overline{v}_a}}$ is needed. If a gate $\ell > i$ has an input wire that is b, then $\mathcal{A}_i$ just computes the gate just like in the real garbling procedure but using oracle $\mathcal{O}^2$ when $F_{w_a^{\overline{v}_a}}$ is needed if ith gate is an AND gate or ith gate is an XOR gate and $\lambda_b = 0$. If a gate $\ell > i$ has an input wire that is b and ith gate is an XOR gate and $\lambda_b = 0$, $\mathcal{A}_i$ has both labels on the wires ($\tilde{w}_b^{\overline{v}_b}$ is chosen) and can compute the gate just like in the real garbling procedure.

Concluding the proof, when $\sigma = 0$, $\mathcal{A}_i$ construct the hybrid $H_{i-1}(x)$, while when $\sigma = 1$, $\mathcal{A}_i$ constructs the hybrid $H_i(x)$. We, therefore, construct a single adversary $\mathcal{A}'$ for $\mathsf{2PRF}$ who chooses a random i and then runs $\mathcal{A}_i$ with adversary $\mathcal{A}$. By a standard hybrid argument, if $\mathcal{A}$ succeeds with non-negligible probability in $\mathsf{Expt}^{\mathsf{priv}}$ then $\mathcal{A}'$ distinguishs between $\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}}(n, 0)$ and $\mathsf{Expt}^{\mathsf{2PRF}}_{\mathcal{F},\mathcal{A}}(n, 1)$, with non-negiliable probability. This contradicts the assumption that $\mathcal{F}$ is a family of pseudorandom functions.

D.1 Obliviousness and Authenticity

The proofs of obliviousness and authenticity are similar to those of Gueron et al. [7]. For completeness, we present them here.

To satisfy the obliviousness requirement, we construct a simulator that outputs (F, X) given only circuit f as an input. Note that the simulator $\mathcal{S}$ completed above for the privacy requirement outputs the triple (F, X, d). However, $\mathcal{S}$ uses circuit f only for generating (F, X), particularly the output $f(x)$, used only for generating d. Thus, we can remove the generation of the decoding information from $\mathcal{S}$'s instruction and obtain a simulator that produces only (F, X) as

required. Proving that this simulator's output is indistinguishable from (F, X) generated by the real scheme is the same as in the proof of privacy.

Regarding authenticity, we need to show that an adversary $\mathcal{A}$ that is given (F, X) as input can output $\tilde{Y}$ such that $\mathsf{Decode}\ (\tilde{Y}, d \neq \{f(x), \perp\})$ with at negligible probability. If we give $\mathcal{A}$ the pair (F, X) generated by our simulator, it can succeed only with probability at most 2^{-n}. This is because in the simulated garbled circuit, for each output wire j corresponding to the jth output bit, $d[j, \overline{f(x)_j}]$ is a random string. Now, if given the real (F, X), the adversary can output such a $\tilde{Y}$ with non-negligible probability, then it could be used by an adversary given (F, X, d) to break the privacy property. Observe that since the adversary in the privacy experiment is given all of the decoding information d, it can efficiently verify if $\mathcal{A}$ output a $\tilde{Y}$ with the property that $\mathsf{Decode}(\tilde{Y}, d) \notin \{f(x), \perp\}$.