Column-wise Garbling, and How to Go Beyond the Linear Model*

Lei Fan
Shanghai Jiao Tong University
fanlei@sjtu.edu.cn

Zhenghao Lu
Shanghai Jiao Tong University
zhenghao.lu.sh@gmail.com

Hong-Sheng Zhou
Virginia Commonwealth University
hszhou@vcu.edu

May 20, 2024

Abstract

In the linear garbling model introduced by Zahur, Rosulek, and Evans (Eurocrypt 2015), garbling an AND gate requires at least 2κ bits of ciphertext, where κ is the security parameter. Though subsequent works, including those by Rosulek and Roy (Crypto 2021) and Acharya et al. (ACNS 2023), have advanced beyond these linear constraints, a more comprehensive design framework is yet to be developed.

Our work offers a novel, unified, and arguably simple perspective on garbled circuits. We introduce a hierarchy of models that captures all existing practical garbling schemes. By determining the lower bounds for these models, we elucidate the capabilities and limits of each. Notably, our findings suggest that simply integrating a nonlinear processing function or probabilistic considerations does not break the 2κ lower bound by Zahur, Rosulek, and Evans. However, by incorporating column correlations, the bound can be reduced to $(1+1/w)\kappa$, where $w \geq 1$. Additionally, we demonstrate that a straightforward extension of Rosulek and Roy's technique (Crypto 2021) does not yield improved results. We also present a methodology for crafting new models and for exploring further extensions of both the new and the existing models.

Our new models set the course for future designs. We introduce three innovative garbling schemes based on a common principle called "majority voting." The third construction performs on par with the state-of-the-art.

*We thank Tomer Ashur, Carmit Hazay, and Rahul Satish, for the discussions about their results in [AHS24]. We thank Taechan Kim for the discussions about his results in [Kim24, BK24]. We remark that, these results [AHS24, Kim24, BK24] and the results in this writeup are concurrently and independently discovered.

Contents

1 Introduction

Yao's garbled circuits [Yao86] are well received in both theory and practice. Today, garbled circuits have become a fundamental tool for constructing constant-round secure multi-party computation. In addition, garbled circuits have been used for constructing cryptographic primitives such as verifiable computation [GGP10], zero-knowledge proofs [JKO13], functional encryption [GKP+13], key-dependent message security [BHHI10], and many more. Please also refer to the great survey by Applebaum [App17].

New insights have also been developed. For example, Ishai et al. [IK00, AIK04] view garbled circuits as randomized encodings of functions; later, Bellare et al. [BHR12b] formalize garbled circuits as a cryptographic primitive. Over the years, huge efforts have been made to improve the performance of garbled circuits. Notably, a long sequence of results (e.g., [BMR90, NPS99, KS08, PSSW09, KMR14, ZRE15, GLNP15, RR21, AAC+23]) have been developed with the goal of minimizing the concrete size of garbled circuits. The state-of-the-art construction [RR21] requires only a ciphertext of 1.5κ bits to garble an AND gate, while garbling an XOR gate is "free."

Zahur et al.'s linear garbling. We must highlight the remarkable work of Zahur, Rosulek and Evans [ZRE15]. Before their contributions, numerous *ad hoc* but practical solutions to garbled circuits have been constructed [BMR90, NPS99, KS08, PSSW09]. Zahur et al. for the first time introduced a *unified* design framework, called the *linear garbling model*, aiming to capture all practical garbling schemes at the time using the so-called "standard techniques"[1]. Very surprisingly, Zahur et al. were able to establish a *lower bound*, stating that any garbling scheme within their linear garbling model must use a ciphertext of at least 2κ bits for garbling an AND gate. The formulation of such model as well as proving a lower bound are truly signifiant: to achieve a smaller ciphertext size, novel constructions must transcend the boundaries of this model.

Beyond linear garbling. Indeed, a few subsequent works have moved beyond Zahur et al.'s linear garbling model. For instance, Rosulek and Roy [RR21] provided a construction requiring merely 1.5κ bits to garble an AND gate. While it preserves most of the "standard techniques" in the linear garbling model, it introduces an innovative method termed "slicing and dicing." Acharya et al. offered a markedly different design approach [AAC+23]. Rather than the conventional row-by-row encryption, it handles the garbling of each gate as a whole. Although the construction in [AAC+23] does not break the 2κ lower bound, the insights in their designs carry significant weight.

Searching for classes of practical solutions to garbled circuits. In this paper, we aim to search for practical solutions to garbled circuits. Our goal is rather ambitious: instead of developing *ad hoc* constructions, we attempt to propose a **methodology**. This includes defining a hierarchy of design models, establishing lower bounds for these models, and introducing new constructions within them. Specifically, we can develop new models on top of existing ones and establish a lower bound for each new model. This will arguably enhance our understanding of the limits and the power of the design techniques in the new model.

We point out that, we are not the first to provide a systematic, instead of an *ad hoc*, treatment for garbling schemes; As discussed above, Zahur et al. [ZRE15] have already demonstrated a beautiful example of linear garbling model. However, we are the first to launch a comprehensive search for many, not just one, design models. Ultimately, our goal is to craft a (nearly) complete vision for advancing solutions in the realm of garbled circuits.

1.1 Our Contributions

In Yao's garbling, we first represent an (efficiently computable) function f as a boolean circuit C, and then garble each gate G_i of the boolean circuit. It seems that we are taking a detour. However, we remark that representing a more complex function f as a less complex boolean circuit C is essential: the boolean circuit can be decomposed into multiple boolean gates $\{\mathsf{G}_i\}$, and each gate can be enumerated through a **constant-size** mapping table between its input and output; this constant-size table can be efficiently "encrypted" or garbled.[2] Is it possible to further "decompose" a boolean gate? If yes, then we may be able to obtain a better understanding of garbling.

[1] The term "standard techniques" specifically means that, aside from the non-linear operations of point-permute and calls to the random oracle, all other operations in the garbling designs are linear. Detailed elaboration is available in Section 3.

[2] In contrast, without this "detour," directly garbling the function f is infeasible: an exponential size table is needed for mapping the input to its output of f; unfortunately, we cannot deal with this huge table!

A new perspective. We introduce a new, fine-grained approach to examining garbled circuits: the **column-wise** perspective[3]. This offers a more unified and arguably simpler understanding of garbling schemes. In the linear garbling model by Zahur et el. [ZRE15], output labels are derived by XORing random oracle outputs, input labels, and ciphertext. This XOR operation can be further refined. The process of obtaining the output label breaks down into κ operations, each requiring a sub-ciphertext[4] and producing a single bit of the output label. While this seems redundant in the linear garbling model, it sets the stage for more complex constructions and for a comprehensive approach to garbled circuits. In Sections 3.1 and 6.1, we will introduce this perspective in detail.

New models and new lower bounds. We propose several extensible models that cover all known practical constructions and establish the corresponding lower bounds.

- MODEL-1: By employing the **column-wise** perspective, we present a redefined linear garbling model. Building on MODEL-1 for additional extensions, like integrating more non-linear operations, is fairly straightforward. For a detailed discussion, refer to Section 3.
- MODEL-2: In MODEL-1, every operation producing a single bit of the output label is linear, and can informally be represented by matrix multiplication, vector inner products, or similar forms. Moving to MODEL-2, we do not specify the operation, allowing for arbitrary non-linearity. However, we demonstrate that such flexibility does not break the lower bound of 2κ when garbling an AND gate. Moreover, MODEL-2 encompasses all constructions that comply with MODEL-1. See Section 4 for further details.
- MODEL-3: In earlier models, each operation produces one bit of the output label. In MODEL-3, we relax this, allowing operations to possibly reject, meaning some might not yield an output bit. While before, κ operations give κ output bits, we might now need more than κ operations. Even so, garbling an AND gate in MODEL-3 still requires at least 2κ-bit ciphertext. Notably, if an operation employs just a 1-bit sub-ciphertext, any construction adhering to MODEL-3 will have an operation rejection probability over $5/8$. MODEL-3 covers all constructions fitting MODEL-2 and the construction in [AAC+23] (where rejection probability is $3/4$). For more details, refer to Section 5.
- MODEL-3′: Expanding on MODEL-2 and adding further correlations, we can create more efficient garbling schemes. Unlike earlier models, MODEL-3′ aims to obtain w bits of the output label from a single operation, with $w \geq 1$. We utilize the connections between different bit positions for a shorter ciphertext. A favorable lower bound is also established in this model. Specifically, garbling an AND gate requires at least $(2 - \upsilon/w) \cdot \kappa$ bits, with υ tied to the correlations and $\upsilon < w$. For the construction in [RR21], $w = 2$ and $\upsilon = 1$. This model offers a broad understanding of the sliced garbling presented in [RR21]. Moreover, MODEL-3′ covers MODEL-2 and the construction in [RR21]. For a deeper dive, see Section 6.
- Other variants: Continuing this approach, we extend the aforementioned models and obtain several intriguing results. For instance, we demonstrate that garbling an AND gate with n inputs and one output necessitates at least $n\kappa$ bits (where $n \geq 2$) in a new model that allows a high fan-in. Additionally, we investigate potential extensions of the model and outline a methodology for such expansions. A thorough explanation of these variants is available in Section 7.

In summary, the models mentioned above follow this relationship: MODEL-1 $\subset$ MODEL-2 $\subset$ MODEL-3, MODEL-3′. Interestingly, our results indicate that simply adopting the concept from [AAC+23] doesn't necessarily lead to a better construction, while it is still important not to overlook its potential. The ideas in [RR21] seem to offer a more promising path towards enhanced constructions. Moreover, we proof that a straightforward extension of the technique presented in [RR21], employing 1.5κ bits to garble an AND gate, is already optimal within a more constrained version of MODEL-3′.

New techniques for establishing lower bounds. Our methods for establishing lower bounds are innovative. Using our **column-wise** perspective, we can view each operation as a mapping from some basic inputs (e.g. bits from random oracle outputs) and a sub-ciphertext, to a single bit of the output label.

In proving the lower bound for MODEL-2, we discover that a succinct mapping table aptly describes the

[3]The term **column-wise** will be clarified in Section 3.1.

[4]A sub-ciphertext can be viewed as a part of the ciphertext used for garbling a gate.

correctness requirement of the garbing scheme. For the privacy requirement, we interpret it as a need for the randomness of the mapping's output distribution. If the sub-ciphertext length is too short, the mapping's potential outputs become limited, challenging the desired output randomness. Hence, we derive a lower bound for the ciphertext length.

In the proof of the lower bound for MODEL-3, we examine the mapping more closely due to the challenges introduced by rejection. We divide the mapping into three categories: certain to fail, successful in some cases, and certain to succeed. By evaluating the probabilities for these categories and integrating them with the desired probability distribution of the output bit, we establish a lower bound.

The proof of the lower bound for MODEL-3′ is more complicated because it introduces certain correlations in the garbling scheme design, not formally discussed before. Therefore, for the first time, we introduce a new concept, **column correlation**, to describe these potential correlations. We find that using **column correlation** can reduce uncertainty in the mapping, and we obtain a more favorable lower bound.

Our proof techniques are completely different from those in [ZRE15]. We also remark that, in some recent and concurrent work [AHS24, Kim24, BK24], the authors also showed some interesting lower bounds for garbled circuits employing different techniques. More details can be found in Section 6.3.2. Moreover, the techniques used in our proofs can inspire new constructions, due to the "visibility" of the correctness and privacy requirements.

New constructions. Along the way, we construct several new garbling schemes. Note that, new ideas have been introduced: we develop a strategy, termed **majority voting**; with a simple "bit-flipping" mechanism, we can obtain efficient garbling schemes; details can be found in Sections 4.3 and 6.4. All our constructions are summarized in Table 1.

Table 1: Our new constructions using **majority voting**.

Constructions	Ciphertext length for garbling an AND gate	Compatibility with free-XOR	Corresponding model
Construction #1	2κ	$\times$	MODEL-2
Construction #2	2κ	✓	MODEL-2
Construction #3	1.5κ	✓	MODEL-3′

To the best of our knowledge, our Construction #1 and Construction #2 are the only garbling schemes that are under MODEL-2, but not MODEL-1. Further, Construction #2 is compatible with free-XOR technique. Interestingly, we discover a mechanism to make non-free-XOR-compatible constructions support free-XOR. Finally, our Construction #3, has the same performance as [RR21], but utilizing more non-linear operations. We remark that, while our constructions might not outperform the state-of-the-art results (e.g., [RR21]), they utilize different ideas. This may allow us to find novel pathways for future optimization and applications of garbled circuits.

Organization. In Section 2, we present the preliminaries. In Section 3, we introduce the column-wise perspective and redefine the linear garbling model as MODEL-1. In Section 4, we define MODEL-2 and explore the role of non-linear mappings. In Section 5, we introduce MODEL-3 and examine the potential of mapping with rejection for better constructions. In Section 6, we define MODEL-3′ and demonstrate the power of introducing correlations. In Section 7, we discuss more variants and potential extensions. Related works can be found in Section 8. Finally, supplemental materials for Sections 3, 4, 5, 6, and 7 can be found in Appendices A, B, C, D, and E respectively.

2 Preliminaries

Notations. In this paper, the security parameter is denoted by κ. The set $\{1, 2, ..., n\}$ is represented by $[n]$. Vectors are shown in bold, like $\boldsymbol{A}$, 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 $\times$. Logarithms default to base 2

unless stated otherwise. The term $\alpha\beta$, for $\alpha, \beta \in \{0,1\}$, means the concatenation of two bits, not multiplication. The symbol $\leftarrow$ indicates uniform random sampling. For instance, $R \leftarrow \mathrm{GF}(2^\kappa)$ means R is randomly chosen from the field $\mathrm{GF}(2^\kappa)$.

2.1 Garbling Schemes

In the realm of garbled circuits, two primary roles exist: the garbler and the evaluator. The garbler encrypts the circuit gate-by-gate and sends the resulting ciphertext to the evaluator. Armed with the input labels of the circuit and each gate's ciphertext, the evaluator decrypts the circuit following its topological order. When dealing with two-party computation, the evaluator obtains the labels corresponding to her inputs through the oblivious transfer protocol.

We use the garbling scheme definition from [RR21], which represents a slight modification of the definitions found in [BHR12b].

Definition 1. *A garbling scheme, denoted as $\mathcal{GC} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$, is composed of four polynomial time algorithms:*

- *Garbling algorithm* Gb*: On input $(1^\kappa, f)$, outputs (F, e, d), where f is a circuit, F is a garbled circuit, e is an input encoding set, and d is an output decoding set.*
- *Encoding algorithm* En*: On input (e, x), outputs X, where x is an input for f, and X is a garbled input.*
- *Evaluation algorithm* Ev*: On input (F, X), outputs Y, where Y is a garbled output.*
- *Decoding algorithm* De*: On input (d, Y), outputs y, where y is a plain output.*

Furthermore, these four algorithms need to satisfy the following properties:

- **Correctness:** For any circuit f and any input x, if we obtain $(F, e, d) \leftarrow \mathsf{Gb}(1^\kappa, f)$, then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x)$ with overwhelming probability.
- **Privacy:** For any circuit f and any input x, a simulator Sim must exist which takes as input $(1^\kappa, f, f(x))$ and produces an output (F, X, d) that is indistinguishable from the set generated in the usual manner.
- **Obliviousness:** For any circuit f and any input x, a simulator Sim must exist which takes as input $(1^\kappa, f)$ and produces an output (F, X) that is indistinguishable from the set generated in the usual manner.
- **Authenticity:** For any PPT adversary, given the input (F, X, d), it should be infeasible to generate a $\tilde{Y}$ distinct from $\mathsf{Ev}(F, X)$ such that $\mathsf{De}(d, \tilde{Y}) \neq \perp$, except with negligible probability.

3 Column-wise Garbling: A New Perspective

In this section, we present a **column-wise** perspective on garbling schemes. Based on this new perspective, we redefine the linear garbling model by Zahur et al. [ZRE15] as MODEL-1. This redefinition is pivotal for understanding our new design models and constructions in following sections.

We focus on garbling an AND gate, which has two inputs and one output. As depicted in Figure 1, each wire is associated with two labels. The two input wires carry labels (A_0, A_1) and (B_0, B_1), while the output wire has labels (C_0, C_1). All these labels are κ-bit strings.

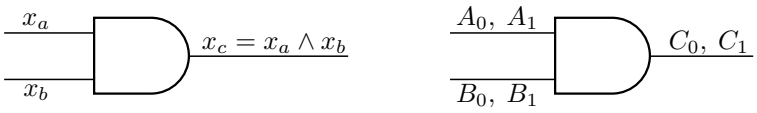

Figure 1: An AND gate and its garbled version.

As stressed by Zahur et al. [ZRE15], constructions in the linear garbling model use "standard techniques." Specifically, these constructions utilize the point-permute optimization and make calls to the random oracle, while other operations are linear.

In the point-permute optimization, the garbler secretly selects a "permute bit" for each wire. To elucidate, the garbler chooses two random bits, a and b, as the permute bits for two input wires. In this context, A_a and B_b correspond to the actual input value of FALSE. The subscript of each label denotes the "color bit," disclosed to the evaluator. However, the evaluator cannot determine the permute bit for each wire. That is, when given the input labels A_α and B_β, the evaluator is aware of α and β but not a and b. As a result, the evaluator cannot identify the actual input values $x_a = \alpha \oplus a$ and $x_b = \beta \oplus b$.

Another non-linear "standard technique" is the use of the random oracle. All constructions under the linear garbling model utilize queries to H, regarded as a random oracle[5]. This oracle accepts arbitrary input and outputs a κ-bit random string. In practice, block ciphers like AES can be used to instantiate the random oracle, ensuring fast computation.

Apart from these, "standard techniques" operations are linear, typically appearing as bit-wise XOR.

3.1 A Column-wise View of Garbling Schemes

Starting with the "standard techniques," we provide a fresh perspective on garbled circuits.

Classical Yao Scheme. We first examine the classical Yao scheme that now includes the point-permute optimization. Using the input labels A_0, A_1, B_0, B_1 and the random oracle, the garbler generates a base for garbling an AND gate, ordered by color bits:

$$\begin{array}{c} \overbrace{\phantom{\mathsf{H}(A_0,B_0)}}^{\textbf{base}} \\ \mathsf{H}(A_0,B_0) \\ \mathsf{H}(A_0,B_1) \\ \mathsf{H}(A_1,B_0) \\ \mathsf{H}(A_1,B_1) \end{array} = \overbrace{\begin{array}{ccccccc} 1 & 1 & \cdots & \boxed{1} & \cdots \\ 0 & 0 & \cdots & \boxed{0} & \cdots \\ 1 & 0 & \cdots & \boxed{0} & \cdots \\ 0 & 0 & \cdots & \boxed{1} & \cdots \end{array}}^{\text{in total, } \kappa \text{ columns}} \quad \underbrace{\phantom{\boxed{1}}}_{t\text{-th column}}$$

The base consists of four rows, each κ bits long. Considering these bits column-wise, the t-th column comprises four bits, where $1 \le t \le \kappa$. Next, the garbler randomly generates two κ-bit output labels, C_0 and C_1, with C_0 corresponding to FALSE. We assume, without loss of generality, that the third row of the base, where the color bits are $(1, 0)$, yields the output label C_1 (this convention persists in subsequent examples). The garbler's task is to map each column on the left to its corresponding column on the right, as illustrated below:

$$\begin{array}{ccccccc} \mathsf{H}(A_0,B_0) & = & 1 & 1 & \cdots & \boxed{1} & \cdots \\ \mathsf{H}(A_0,B_1) & = & 0 & 0 & \cdots & \boxed{0} & \cdots \\ \mathsf{H}(A_1,B_0) & = & 1 & 0 & \cdots & \boxed{0} & \cdots \\ \mathsf{H}(A_1,B_1) & = & 0 & 0 & \cdots & \boxed{1} & \cdots \end{array} \quad \underbrace{\phantom{\boxed{1}}}_{t\text{-th column}} \quad \longrightarrow \quad \begin{array}{ccccccc} 0 & 0 & \cdots & \boxed{1} & \cdots & = & C_0 \\ 0 & 0 & \cdots & \boxed{1} & \cdots & = & C_0 \\ 1 & 0 & \cdots & \boxed{0} & \cdots & = & C_1 \\ 0 & 0 & \cdots & \boxed{1} & \cdots & = & C_0 \end{array} \quad \underbrace{\phantom{\boxed{1}}}_{t\text{-th column}}$$

The four bits in the t-th column on the left side are denoted as $\boldsymbol{M}[t] = (M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$ and those in the t-th column on the right side as $\boldsymbol{Z}[t] = (Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$. In the classical Yao scheme, the garbler uses a 4-bit sub-ciphertext $\boldsymbol{G}[t] = (G_{00}[t], G_{01}[t], G_{10}[t], G_{11}[t])$ to achieve the mapping function MAP shown below:

$$\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} M_{00}[t] \oplus G_{00}[t] \\ M_{01}[t] \oplus G_{01}[t] \\ M_{10}[t] \oplus G_{10}[t] \\ M_{11}[t] \oplus G_{11}[t] \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix} = \boldsymbol{Z}[t], \text{ where } Z_{00}[t] = Z_{01}[t] = Z_{11}[t].$$

Since each column requires a 4-bit sub-ciphertext and there are κ columns, the total ciphertext length is 4κ.

For the evaluator, given input labels A_α and B_β, she can reconstruct the $(2\alpha + \beta + 1)$-th row of the base. Utilizing ciphertext $\boldsymbol{G}$, the t-th bit of the output label can be calculated as $M_{\alpha\beta}[t] \oplus G_{\alpha\beta}[t]$.

[5] We view H as a random oracle for now. In later discussions about ensuring the security of our constructions, H can be treated as a certain circular hash function.

Several observations arise from the aforementioned process. Firstly, the structure of the base doesn't depend on which row corresponds to the output label C_1. Secondly, the function MAP used to complete the mapping is also independent of which row yields C_1. Moreover, MAP operates on a single column of the base, generating one bit of the output label per operation. While the classical Yao scheme uses row-wise XOR operations, our view may initially appear redundant. However, we will later show that it offers substantial flexibility.

Row Reduction (GRR3). In the classical scheme above, the garbler first chooses output labels and then determines the ciphertext for each column to finalize the mapping. Randomly selecting two output labels introduces two degrees of freedom. GRR3 removes one, reducing the ciphertext length to 3κ.

Initially, the garbler creates a base (identical to the previous one). Consistent with the previous example, we assume the third row of the base yields C_1. However, this time, the garbler does not randomly select C_0. Instead, the t-th bit from the first row of the base becomes the t-th bit of C_0. Specifically, for the t-th column, the garbler selects a 4-bit sub-ciphertext $\boldsymbol{G}[t] = (0, G_{01}[t], G_{10}[t], G_{11}[t])$ to finalize the mapping as follows:

$$\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} M_{00}[t] \oplus 0 \\ M_{01}[t] \oplus G_{01}[t] \\ M_{10}[t] \oplus G_{10}[t] \\ M_{11}[t] \oplus G_{11}[t] \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix} = \boldsymbol{Z}[t], \text{ where } Z_{00}[t] = Z_{01}[t] = Z_{11}[t].$$

Since the first bit of $\boldsymbol{G}[t]$ is always 0, it need not be transmitted, reducing the ciphertext to 3κ bits.

Half Gates. The half-gates garbling scheme further exploits the degrees of freedom in the output labels. First, the garbler generates a base as follows:

$$\begin{array}{l} \mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \\ \mathsf{H}(A_0) \oplus \mathsf{H}(B_1) \oplus A_0 \\ \mathsf{H}(A_1) \oplus \mathsf{H}(B_0) \\ \mathsf{H}(A_1) \oplus \mathsf{H}(B_1) \oplus A_1 \end{array}$$

For the t-th column, the garbler purposefully chooses a sub-ciphertext $\boldsymbol{G}[t] = (0, G_{01}[t], G_{10}[t], G_{01}[t] \oplus G_{10}[t])$ to meet the equation below:

$$\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} M_{00}[t] \oplus 0 \\ M_{01}[t] \oplus G_{01}[t] \\ M_{10}[t] \oplus G_{10}[t] \\ M_{11}[t] \oplus G_{01}[t] \oplus G_{10}[t] \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix} = \boldsymbol{Z}[t], \text{ where } Z_{00}[t] = Z_{01}[t] = Z_{11}[t].$$

3.2 MODEL-1: Column-wise Garbling with Linear Mapping

In our perspective, as outlined in the previous section, the essence of constructing a garbling scheme lies in identifying a suitable base and a mapping function MAP. Each operation both generates one bit of the output label and ensures the scheme's correctness and privacy.

Considering the original linear garbling model, from the evaluator's viewpoint, obtaining the output label entails computing a linear combination of ciphertext, input labels, and some random oracle outputs, based on the color bits. The combined input labels and random oracle outputs can be regarded as the base. Performing a bit-wise XOR between the base and the ciphertext can be thoroughly reduced to a column-wise XOR operation, which we term the mapping function MAP.

Our goal is to reshape the linear garbling model through our lens. Building on the original definition, a detailed understanding of it is pivotal before moving forward. While we provide brief explanations here, for further details, readers are encouraged to refer to Appendix A.1. The linear garbling model, serving as a framework, possesses numerous parameters. Any particular construction within this framework instantiates these parameters. The linear combination of input labels and random oracle outputs can be depicted as a vector inner product.

We redefine the linear garbling model as MODEL-1, a starting point for exploring more advanced models.

- Garbling algorithm Gb:
  Parameterized by integers m, r, q, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{M}_{ij} \mid i,j \in \{0,1\}\}$, matrices $\{\boldsymbol{\mathcal{P}}[u] \mid u \in [\kappa]\}$, random oracle RO and mapping function MAP. Each vector is of length $r+q$ with entries in $\mathrm{GF}(2^\kappa)$. Each matrix is $4 \times m$ in size with entries in $\mathrm{GF}(2)$. The RO is mapped as $\{0,1\}^* \to \mathrm{GF}(2^\kappa)$. The MAP breaks down into two linear sub-functions: $(\mathsf{MAP}_1, \mathsf{MAP}_2)$, where MAP_1 performs matrix multiplication in $\mathrm{GF}(2)$, while MAP_2 carries out a bit-wise XOR operation.

  1. For $i \in [r]$, choose $R_i \leftarrow \mathrm{GF}(2^\kappa)$.
  2. Make q distinct queries to the RO, which can be determined based on the R_i values. Let $Q_1, \ldots, Q_q$ denote the responses to these queries, and define $\boldsymbol{S} := (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$.
  3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.
  4. Compute $A_0 := \langle \boldsymbol{A}_0, \boldsymbol{S} \rangle$, $A_1 := \langle \boldsymbol{A}_1, \boldsymbol{S} \rangle$, $B_0 := \langle \boldsymbol{B}_0, \boldsymbol{S} \rangle$, $B_1 := \langle \boldsymbol{B}_1, \boldsymbol{S} \rangle$. Then $A_0||0$ and $A_1||1$ are labels for one input wire, and $B_0||0$ and $B_1||1$ for the other. Subscripts indicate the color bits, with A_a and B_b representing FALSE.
  5. For $i,j \in \{0,1\}$, compute $M_{ij} := \langle \boldsymbol{M}_{ij}, \boldsymbol{S} \rangle$. Then $(M_{00}, M_{01}, M_{10}, M_{11})^\top$ is the base. The t-th column of the base, represented by $\boldsymbol{M}[t]$, is composed of 4 bits and can be interpreted as $(M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$.
  6. For $t \in [\kappa]$, find m-bit sub-ciphertext $\boldsymbol{G}[t]$ (which can be considered as a m-dimensional vector, with entries in $\mathrm{GF}(2)$), such that the following three conditions are satisfied:
     (a) $\mathsf{MAP}_1(\boldsymbol{\mathcal{P}}[t], \boldsymbol{G}[t]) = \boldsymbol{\mathcal{P}}[t] \times \boldsymbol{G}[t]^\top = \widetilde{\boldsymbol{G}}[t]^\top$;
     (b) $\mathsf{MAP}_2(\boldsymbol{M}[t], \widetilde{\boldsymbol{G}}[t]) = \begin{pmatrix} M_{00}[t] \oplus \widetilde{G}_{00}[t] \\ M_{01}[t] \oplus \widetilde{G}_{01}[t] \\ M_{10}[t] \oplus \widetilde{G}_{10}[t] \\ M_{11}[t] \oplus \widetilde{G}_{11}[t] \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix}$;
     (c) $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$[6].
     Then use $Z_{ab}[t]$ as the t-th bit of the output label C_0, and $Z_{(a\oplus 1)(b\oplus 1)}[t]$ as the t-th bit of the output label C_1. Here, C_0 corresponds to FALSE. The sub-ciphertext $\boldsymbol{G}[t]$, with a length of m, represents the t-th part of the whole ciphertext $\boldsymbol{G}$.

- Encoding algorithm En:
  Given inputs $x_a, x_b \in \{0,1\}$, 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 Ev:
  Parameterized by integers q, vectors $\{\boldsymbol{V}_{\alpha\beta} \mid \alpha,\beta \in \{0,1\}\}$, matrices $\{\boldsymbol{\mathcal{P}}[u] \mid u \in [\kappa]\}$, random oracle RO and mapping function MAP. The length of each $\boldsymbol{V}_{\alpha\beta}$ is $2+q$, with entries in $\mathrm{GF}(2^\kappa)$.

  1. The inputs are wire labels $A_\alpha||\alpha$, $B_\beta||\beta$ and the ciphertext $\boldsymbol{G}$.
  2. Make q distinct queries to the RO, which can be determined based on the input wire labels. Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, and define $\boldsymbol{T} := (A_\alpha, B_\beta, Q'_1, \ldots, Q'_q)$.
  3. Compute $V_{\alpha\beta} := \langle \boldsymbol{V}_{\alpha\beta}, \boldsymbol{T} \rangle$, and let $V_{\alpha\beta}[t]$ denote the t-th bit of $V_{\alpha\beta}$.
  4. For $t \in [\kappa]$, compute $(\widetilde{G}_{00}[t], \widetilde{G}_{01}[t], \widetilde{G}_{10}[t], \widetilde{G}_{11}[t])^\top := \mathsf{MAP}_1(\boldsymbol{\mathcal{P}}[t], \boldsymbol{G}[t])$, then use $Z_{\alpha\beta}[t] := V_{\alpha\beta}[t] \oplus \widetilde{G}_{\alpha\beta}[t]$ as the t-th bit of the output label.

3.3 Reflection and Discussion

Our MODEL-1, compared to Zahur et al.'s [ZRE15] original definition of the linear garbling model, is a bit more complex. This intricacy arises from our endeavor to capture the "linear" constraints required of the mapping function MAP. But when certain constraint is relaxed, we arrive at a more elegant model, detailed in the succeeding section.

[6] Recall that the subscript denotes the concatenation of two bits, rather than multiplication. In addition, in this paper, we study the garbling of an AND gate, so the output of MAP_2 needs to satisfy this equation. Indeed, we can take the gate type as a parameter to generalize the model's definition, rather than hardcoding it into the model.

From the evaluator's execution of the Ev algorithm, we can clearly see the connection between our MODEL-1 and the original linear garbling model. As noted earlier, the evaluator derives the output label from a linear combination of input labels, random oracle outputs, and ciphertext. In MODEL-1, this computation can be divided into two steps. First, the evaluator forms a linear combination of input labels and random oracle outputs, effectively equating to a row in the base. Then, for the t-th bit of this row, an XOR operation with a linear transformation (dictated by $\boldsymbol{\mathcal{P}}[t]$) of the sub-ciphertext $\boldsymbol{G}[t]$ yields the corresponding bit of the output label.

In our MODEL-1, we can consider $\boldsymbol{G}[t]$ as a combination of the t-th bits from $G_1, \ldots, G_m$ in the original definition. However, we do not claim our MODEL-1 captures the linear garbling model, or vice versa. More explicitly, if the coefficient multiplied by the ciphertext in the linear garbling model during evaluation belongs to $\mathrm{GF}(2)$, then our MODEL-1 includes the linear garbling model. Among all known constructions, only the scheme in [PSSW09] fails to meet this criterion, due to the coefficient belonging to $\mathrm{GF}(2^\kappa)$, a result of the polynomial interpolation method[7]. This minor flaw will be addressed and in Section 6.1. Notably, all existing schemes under the linear garbling model, except for [PSSW09], find their counterparts in our MODEL-1.

In the linear garbling model, garbling an AND gate requires a 2κ-bit ciphertext. This lower bound also holds for our MODEL-1, which will naturally emerge in the next section.

As can be seen, our MODEL-1 effectively "slices" the linear garbling model, transforming its row-wise processing into single-bit operations. This detailed design model provides more opportunities for further extensions:

1. In MODEL-1, the mapping function MAP blends matrix multiplication with bitwise XOR, giving MAP a somewhat "linear" character. However, it's possible to design a garbling scheme that introduces non-linear operations on each column. This leads us to the question:

   Without constraints on the mapping function MAP*, could we achieve shorter ciphertext?*

2. In MODEL-1, the goal is to determine a sub-ciphertext for each column that, when processed through the function MAP, yields one bit of the output label. However, schemes such as [AAC+23] offer the flexibility to bypass certain columns that do not have a suitable sub-ciphertext for successful mapping. This prompts the question:

   Is a better construction possible if the mapping function MAP *allows rejection?*

3. In MODEL-1, an initial base is generated. Within this base, any two columns are independent, which makes it challenging to leverage correlations between columns. Indeed, [RR21] circumvents this limitation by introducing two bases and exploiting the correlations between them. This approach reduces the ciphertext length for an AND gate to 1.5κ. Therefore, the question arises:

   How much improvement can be achieved by introducing additional bases*?*

In the following Sections 4, 5 and 6, we will answer these three questions.

4 Column-wise Garbling: Dealing with Non-linear Mapping

In this section, we will address the first question:

Without constraints on the mapping function MAP*, could we achieve shorter ciphertext?*

[7] Indeed, polynomial interpolation is impractical. [GLNP15] introduced another GRR2 method, which primarily uses XOR operations and is simple to implement.

4.1 MODEL-2: Column-wise Garbling with Unconstraint Mapping

In MODEL-1, the mapping function MAP for each column is "linear." When we remove this constraint and place no restrictions on the MAP, we introduce a more refined model, MODEL-2. It's important to note that MODEL-2 encompasses MODEL-1. Therefore, constructions under MODEL-1 are also included in MODEL-2. This encompasses classical schemes like Yao's (with point-permute optimization), GRR3 [NPS99], free-XOR [KS08], GRR2 [GLNP15], half-gates [ZRE15], among others. In subsequent sections, we will introduce some new constructions exclusive to MODEL-2 and not part of MODEL-1.

The key difference between MODEL-2 and MODEL-1 is in the 6-th step of the Gb algorithm and the 4-th step of the Ev algorithm. In MODEL-2, MAP is not confined to linear operations such as matrix multiplication or bitwise XOR; it can be arbitrary function. Specifically, for the t-th column of the base, the gabrler needs to find a m-bit sub-ciphertext $\boldsymbol{G}[t]$ such that the following two conditions are satisfied:

1. $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}_{00}(M_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{01}(M_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{10}(M_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{11}(M_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix}$;
2. $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$.

We decompose the mapping function MAP into four sub-functions: $(\mathsf{MAP}_{00}, \mathsf{MAP}_{01}, \mathsf{MAP}_{10}, \mathsf{MAP}_{11})$. In MODEL-1, these four sub-functions simply perform bit-wise XOR to some extent. There are no constraints on the MAP here, making MODEL-2 more concise than MODEL-1. The formal definition of MODEL-2 is provided in Appendix B.1.

4.2 A Lower Bound

In the work by Zahur et al. [ZRE15], they established an intriguing lower bound: any garbling scheme conforming to the linear garbling model must have at least 2κ-bit ciphertext to garble an AND gate. We aim to establish a similar lower bound for schemes under MODEL-2.

As already noted by Zahur et al. [ZRE15], to derive a meaningful bound, a fundamental methodology is imperative. We make the assumption that the design of a garbling scheme is solely based on symmetric-key primitives. Moreover, we do not limit the computational resources of the participating parties, assuming only that they will make polynomial queries to the random oracle. In other words, we derive the lower bound in the world of Minicrypt [Imp95]. We also adopt the definition of *ideal security* from [ZRE15], which states that if a garbling scheme has *ideal security*, then no adversary can win the security game with an advantage greater than $\mathrm{poly}(\kappa)/2^{\kappa}$.

Now, we formally state the following theorem:

Theorem 1. *Any ideally secure garbling scheme for an* AND *gate conforming to* MODEL-2 *must satisfy* $|\boldsymbol{G}| \geq 2\kappa$.

In other words, the use of non-linear operations for each column in MODEL-2 does not break the lower bound set by the linear garbling model.

Proof Sketch. Each operation maps a column from the base to four bits using the function MAP. For a fixed $\boldsymbol{M}[t]$, the number of potential values that $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t]) = \mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ can assume is limited by $2^{|\boldsymbol{G}[t]|} = 2^m$, where m is the length of $\boldsymbol{G}[t]$. Out of these four bits, at least three must be the same, depending on the selected permute bits. For instance, with permute bits $(a, b) = (0, 0)$, the fourth row of the base will correspond to the output TRUE, making the first three of the four bits identical. According to the permute bits (a, b), the four following conditions must each be met:

$$\begin{aligned} Z_{00}[t] = Z_{01}[t] = Z_{10}[t], \text{ when } (a, b) = (0, 0) \\ Z_{00}[t] = Z_{01}[t] = Z_{11}[t], \text{ when } (a, b) = (0, 1) \\ Z_{00}[t] = Z_{10}[t] = Z_{11}[t], \text{ when } (a, b) = (1, 0) \\ Z_{01}[t] = Z_{10}[t] = Z_{11}[t], \text{ when } (a, b) = (1, 1). \end{aligned}$$

Using a proof by contradiction, we assume $m = 1$. Given this, for a fixed $\boldsymbol{M}[t]$, the vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ can only take two distinct values. However, these values must cover the four cases where

permute bits (a, b) take different values. This means that one of the two values must be either $(0, 0, 0, 0)$ or $(1, 1, 1, 1)$ which is suitable for all the cases. This results in an increased probability that the bits at the same position in C_0 and C_1 are identical, violating privacy. Therefore, we conclude that $m \geq 2$.

A complete proof can be found in Appendix B.2. □

Remark 1. The proof for the lower bound in MODEL-1 can be viewed as a special case of the aforementioned proof. We note that merely incorporating non-linearity in column-wise operations does not lead to a construction with a reduced ciphertext length. This observation suggests us to define later models. It is also crucial to underline that our proof technique serves as a strong inspiration for designing new schemes. While it is easy to ensure correctness, the main challenge is to find a mapping function MAP whose output distribution meets the privacy requirement.

4.3 New Constructions

To the best of our knowledge, there currently exists no construction that belongs to MODEL-2 but not to MODEL-1. Here, we introduce two novel constructions that employ a non-linear function MAP for each column's processing. Both constructions pivot around a core principle, which we term as **majority voting**. The first construction necessitates a 2κ-bit ciphertext to garble an AND gate and does not support free-XOR. The second construction is not only compatible with free-XOR but also requires a ciphertext length of merely 2κ bits, matching the efficiency of half gates.

Throughout this section, we maintain the assumption that labels A_0, B_1, and C_0 correspond to FALSE.

4.3.1 Construction #1: 2κ without Free-XOR

We now start from the definition of MODEL-2 and aim to obtain a succinct scheme. In the Gb algorithm, the first step is to construct a base. We won't go into depth about the base here, postponing that discussion until the next construction. Currently, we assume it takes the following straightforward form:

$$\begin{array}{c} \mathsf{H}(A_0, B_0) \\ \mathsf{H}(A_0, B_1) \\ \mathsf{H}(A_1, B_0) \\ \mathsf{H}(A_1, B_1) \end{array}$$

We need to devise a function MAP to map each column of this base to a vector $\boldsymbol{Z}[t]$. To ensure correctness, the vector $\boldsymbol{Z}[t] = (Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ obtained from $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ must satisfy $Z_{00}[t] = Z_{01}[t] = Z_{11}[t]$. For each column $\boldsymbol{M}[t] = (M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$, if $M_{00}[t] = M_{01}[t] = M_{11}[t]$ already holds, then there's no need for ciphertext and MAP can be an identity mapping from $\boldsymbol{M}[t]$ to $\boldsymbol{Z}[t]$. However, the probability that $\mathsf{H}(A_0, B_0) = \mathsf{H}(A_0, B_1) = \mathsf{H}(A_1, B_1)$ is negligible, so a simple identity map is evidently insufficient.

In fact, among $M_{00}[t]$, $M_{01}[t]$ and $M_{11}[t]$, at least two bits are the same. Therefore, the garbler can simply flip the differing bit to obtain three identical bits. Below is a detailed description of the operation to be performed by the garbler and the evaluator.

Garbled Circuit Generation. For the t-th column, the garbler expects that $M_{00}[t]$, $M_{01}[t]$, and $M_{11}[t]$ can be flipped, or corrected to the same value. Using an idea similar to **majority voting**, correction can be achieved using just two bits per column. Specifically, for the t-th column, the garbler will find that at least two of the bits in $(M_{00}[t], M_{01}[t], M_{11}[t])$ are the same. If there is a bit different from the majority, then the garbler simply points out the position of that bit using the corresponding color bits. For example, if $(M_{00}[t], M_{01}[t], M_{11}[t]) = (0, 1, 0)$, then the garbler uses ciphertext $(0, 1)$ to indicate that $M_{01}[t]$ should be flipped. After this correction, the garbler has $M_{00}[t] = M_{01}[t] \oplus 1 = M_{11}[t]$, and he uses this value as the t-th bit of C_0. If all three bits are the same, then the garbler chooses to flip the bit $M_{10}[t]$, i.e. using color bits $(1, 0)$ to indicate this position - doing so is meaningless in terms of correctness, but privacy is ensured.

In summary, the garbler uses two bits per column to denote which position requires flipping. These two bits are part of the ciphertext $\boldsymbol{G}$ that should be sent to the evaluator. Since there are κ columns in total,

the length of $\boldsymbol{G}$ is 2κ. Table 2 details how the garbler constructs the sub-ciphertext based on the value of $\boldsymbol{M}[t] = (M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$[8].

Table 2: Sub-ciphertext generation for the t-th column when garbling an AND gate and $(a, b) = (0, 1)$.

$M_{00}[t]$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$M_{01}[t]$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$M_{10}[t]$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$M_{11}[t]$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\boldsymbol{G}[t]$	10	11	10	11	01	00	01	00	00	01	00	01	11	10	11	10

Garbled Circuit Evaluation. Next, we turn our attention to the evaluator. For an AND gate, assume the evaluator possesses the ciphertext $\boldsymbol{G}$, input labels A_α and B_β, and is aware of their subscripts (the color bits of input labels). Then, the evaluator makes a random oracle query $\mathsf{H}(A_\alpha, B_\beta)$. For each position, if $\boldsymbol{G}[t] = (\alpha, \beta)$, then the evaluator uses $M_{\alpha\beta}[t] \oplus 1$ as the t-th bit of the output label $C_{\alpha\wedge(\beta\oplus 1)}$. Otherwise, she directly adopts $M_{\alpha\beta}[t]$ as the t-th bit of $C_{\alpha\wedge(\beta\oplus 1)}$.

Informally, the function $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ flips a bit in $\boldsymbol{M}[t]$ as indicated by $\boldsymbol{G}[t]$. If $\boldsymbol{G}[t] = (\alpha, \beta)$, then the $(2\alpha + \beta + 1)$-th bit of $\boldsymbol{M}[t]$ should be flipped. It's evident that MAP is a non-linear operation, which cannot be expressed as a fixed matrix multiplication, vector inner product, or bit-wise XOR. Consequently, this construction falls outside the scope of MODEL-1. Indeed, our majority voting method is only barely non-linear since it can be represented by a quadratic function. However, MODEL-2 is significantly more general than MODEL-1, as it accommodates any required mapping function MAP, whereas MAP in MODEL-1 must be linear.

A notable advantage of our approach is its scalability, allowing it to be applied to garble gates of any type, including those with multiple inputs and outputs. In Appendix B.3, we provide a brief introduction on how to garble an XOR gate using the bit-flipping method.

4.3.2 Construction #2: 2κ with Free-XOR

In the free-XOR setting [KS08], every wire has a global secret offset Δ between its two labels, such as $A_0 \oplus A_1 = \Delta$. In this setting, it becomes evident that the previous construction doesn't meet the requirements, as there is no inherent relationship between C_0 and C_1 when they are obtained by flipping bits. Recall that in the previous construction, we generated the base in a straightforward manner. However, by designing a more intricate base, we can achieve compatibility with free-XOR.

In the **majority voting** method, we perform a bit flip for each column. Specifically, given $\boldsymbol{M}[t] = (M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$, after undergoing a particular bit flip, the resulting $\boldsymbol{Z}[t] = (Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ will satisfy the following relationship:

$$Z_{00}[t] \oplus Z_{01}[t] \oplus Z_{10}[t] \oplus Z_{11}[t] = M_{00}[t] \oplus M_{01}[t] \oplus M_{10}[t] \oplus M_{11}[t] \oplus 1.$$

Considering all κ columns together and perform a bitwise XOR on the four rows of the base, we arrive at the following equation:

$$C_0 \oplus C_0 \oplus C_1 \oplus C_0 = C_0 \oplus C_1 = M_{00} \oplus M_{01} \oplus M_{10} \oplus M_{11} \oplus 1^\kappa.$$

Therefore, our task is to find a base with four rows that satisfy $M_{00} \oplus M_{01} \oplus M_{10} \oplus M_{11} = \Delta \oplus 1^\kappa$. By achieving this, we ensure that $C_0 \oplus C_1 = \Delta$. Locating such a base is relatively trivial. For instance, it can

[8]For brevity, in Table 2, $\boldsymbol{G}[t]$ is denoted as xy, signifying (x, y).

take the following form[9]:

$$\begin{array}{l}\mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \oplus A_0 \oplus 1^\kappa \\ \mathsf{H}(A_0) \oplus \mathsf{H}(B_1) \\ \mathsf{H}(A_1) \oplus \mathsf{H}(B_0) \oplus A_1 \\ \mathsf{H}(A_1) \oplus \mathsf{H}(B_1)\end{array}$$

Given the relationship $A_0 \oplus A_1 = \Delta$, it can be confirmed that the bitwise XOR of the four rows mentioned above is $\Delta \oplus 1^\kappa$. Indeed, the structure of the base is not unique, and we can also use a construction like the one below:

$$\begin{array}{l}\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus A_0 \oplus 1^\kappa \\ \mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \\ \mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus A_1 \\ \mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_1)\end{array}$$

The mapping function MAP remains consistent with the one discussed in Construction #1, employing the **majority voting** method to flip a specific bit. Further elaboration is not provided here. In Appendix B.4, we provide a formal description and security proof for this construction.

Remark 2. In comparison to the previous construction, this approach is compatible with free-XOR, implying that garbling an XOR gate is cost-free. However, we refrain from asserting that this construction is superior. The rationale behind this is that free-XOR relies on stronger security assumptions and may be more vulnerable to practical attacks, including side-channel attacks [LH23].

Interestingly, the GRR2 construction in [GLNP15] and the half gates in [ZRE15] have a relationship similar to that between our Construction #1 and Construction #2. That is, they employ the same mapping function MAP but with different bases.

5 Column-wise Garbling via "Mapping with Rejection"

We observe that even without constraints on the function MAP, MODEL-2 cannot capture Ashur et al.'s construction [AAC+23]. In this section, we introduce a new model that incorporates probability and address the following question:

Is a better construction possible if the mapping function MAP allows rejection?

5.1 MODEL-3: Column-wise Garbling via "Mapping with Rejection"

Revisiting the Techniques in [AAC+23]. We first revisit Ashur et al.'s construction [AAC+23] from a column-wise perspective. In their construction, the garbler generates a simple base as follows (not in the free-XOR setting):

$$\begin{array}{l}\mathsf{H}(A_0, B_0) \\ \mathsf{H}(A_0, B_1) \\ \mathsf{H}(A_1, B_0) \\ \mathsf{H}(A_1, B_1)\end{array}$$

For the t-th column of the base, the garbler aims to map it to $\boldsymbol{Z}[t] = (Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ using the function MAP. We once again assume that labels A_0, B_1, and C_0 correspond to FALSE, which implies that $\boldsymbol{Z}[t]$ must satisfy $Z_{00}[t] = Z_{01}[t] = Z_{11}[t]$. In Ashur et al.'s construction [AAC+23], the function MAP acts as an identity map to some extent. If the four bits of the t-th column, represented as $\boldsymbol{M}[t] = (M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$, already meet the condition $M_{00}[t] = M_{01}[t] = M_{11}[t]$, then $\boldsymbol{Z}[t]$ is set to $\boldsymbol{M}[t]$. Otherwise, this column is skipped and the next column is processed. The garbler uses a 1-bit sub-ciphertext $\boldsymbol{G}[t]$ to inform the evaluator whether to skip the t-th column. A value of $\boldsymbol{G}[t] = 1$ indicates that the column achieves the mapping and contributes one bit to the output label. If $\boldsymbol{G}[t] = 0$, the column should be skipped.

[9] Technically, within our definition of MODEL-2, the base should not include specific constants like 1^κ. Nevertheless, for the sake of clarity, we've included 1^κ in the first row of the base here. This inclusion is inconsequential, as we can encompass the act of XORing the first row with 1^κ within the sub-function MAP_{00}.

For any given $\boldsymbol{M}[t]$, the probability that it satisfies $M_{00}[t] = M_{01}[t] = M_{11}[t]$ is $1/4$. Therefore, the probability of successfully processing a column is $1/4$. Each successful process contributes one bit to the output label. To obtain an output label of κ bits, an average of 4κ columns are required. This means the average length of the base (output length of the random oracle) should be 4κ, and the average ciphertext length is also 4κ.

Building on the above observation, we incorporate it into MODEL-3. Here, each column is successfully processed with a probability p. In MODEL-2, every column can be processed successfully, representing a special case of MODEL-3 with $p = 1$. Therefore, MODEL-3 can cover all known constructions that conform to MODEL-2, as well as the construction in [AAC+23].

The main distinction between MODEL-3 and MODEL-2 lies in the 6-th step of the Gb algorithm and the 4-th step of the Ev algorithm. Specifically, for the garbler, the processing of the t-th column will fall into two cases:

1. *With a probability of* p, successfully find m-bit sub-ciphertext $\boldsymbol{G}[t]$ such that the following two conditions are satisfied:
   (a) $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}_{00}(M_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{01}(M_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{10}(M_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{11}(M_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix}$;
   (b) $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$.
2. *With a probability of* $1 - p$, the desired mapping does not exist. Using a particular $\boldsymbol{G}[t]$ to indicate that this column is skipped.

In the first case, one bit of the output label will be successfully retrieved, while in the second case, it cannot be obtained. We present a detailed definition of MODEL-3 in Appendix C.1.

5.2 A Lower Bound

Among existing constructions, only the one in [AAC+23] employs a mapping with rejection, where $p < 1$. Nonetheless, the investigation into the role of probability in garbling schemes is intriguing and vital. However, from our analysis, simply introducing probability in the processing of a single column does not appear to yield better results.

Theorem 2. *Any ideally secure garbling scheme for an* AND *gate conforming to* MODEL-3 *must satisfy* $|\boldsymbol{G}| \geq 2\kappa$ *on average.*

Proof Sketch. To prove this, we only need to demonstrate the following: If $p = 1$, then $m \geq 2$; If $m = 1$, then $p \leq 1/2$, where m is the length of the sub-ciphertext. The former can be directly derived from Theorem 1, so we only need to prove the latter.

Given $m = 1$, the sub-ciphertext $\boldsymbol{G}[t]$ can only assume two distinct values. One of these values indicates the failure of the mapping. Therefore, for a fixed $\boldsymbol{M}[t]$, $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ can produce a maximum of one valid output (which should have three identical bits according to the selected permute bits).

Next, we classify different values of $\boldsymbol{M}[t]$. Suppose x potential values of $\boldsymbol{M}[t]$ that map to outputs in the set $\{(0,0,0,0),(1,1,1,1)\}$, then these values are valid for any case (4 different permute bit choices) and make the t-th bit of C_0 and the t-th bit of C_1 be identical. Assume y values that map to other valid outputs, then they cover only one case and lead to the t-th bit of C_0 differing from the t-th bit of C_1. The remaining values, totaling $16 - x - y$, don't map to valid outputs. To ensure privacy, the probability that the t-th bit of C_0 and C_1 are identical should be $1/2$, resulting in $y = 4x$. Based on this relationship, we can conclude that $p < 1/2$.

We provide a complete proof in Appendix C.2. □

Remark 3. While allowing mapping with rejection does not seem to result in a better construction, this concept is still insightful. It is possible that through more sophisticated constructions that go beyond the constraints of MODEL-3, the full potential of this approach can be unlocked. In Section 7.3, we will delve deeper into this idea.

6 Multi-Column Garbling

Now, it is time to answer the final question:

How much improvement can be achieved by introducing additional bases?

6.1 From Single- to Multi-Column Garbling

What we are aiming for is to surpass the lower bound of MODEL-2 or MODEL-3. The construction in [RR21] appears to have already achieved this. Before delving into a detailed review of their work, we first consider a straightforward approach.

A Straightforward Approach. In the definition of MODEL-2 or MODEL-3, we process one column at a time, aiming to obtain one bit of the output label. An intuitive extension would involve processing multiple columns at once with a single ciphertext to obtain several bits of the output label. This method indeed offers a more favorable lower bound.

In Appendix D.1, we briefly describe a potential approach that might come close to a ciphertext length of $\log(3)\cdot\kappa$ by simultaneously processing multiple columns of the base. However, existing construction [RR21] has already achieved an overhead of 1.5κ for garbling an AND gate, noting that $1.5 < \log 3$. Consequently, simply processing multiple columns will not be able to cover the construction in [RR21] as a special case. In the definition of MODEL-2, the form of the base is some linear combinations of random oracle outputs and random numbers, as exemplified below:

$$\begin{aligned}&\mathsf{H}(A_0 \oplus B_0) \oplus A_0 \oplus \dots\\ &\mathsf{H}(A_0 \oplus B_1) \oplus A_0 \oplus \dots\\ &\mathsf{H}(A_1 \oplus B_0) \oplus A_1 \oplus \dots\\ &\mathsf{H}(A_1 \oplus B_1) \oplus A_1 \oplus \dots\end{aligned}$$

Given the randomness of the random oracle outputs, any two columns, such as $\boldsymbol{M}[t]$ and $\boldsymbol{M}[t+1]$, are independent of each other. Therefore, processing multiple columns simultaneously does not yield a significant reduction in ciphertext length, rendering our first attempt unsuccessful. In fact, we can view the construction in [PSSW09] as handling multiple columns of the base together. However, despite this, it does not result in a better construction.

An appealing direction is to introduce correlation to bypass the lower bound. Indeed, this can be accomplished by incorporating additional bases.

Revisiting the "Slicing and Dicing" Technique. In [RR21], Rosulek and Roy develop a novel technique called "slicing and dicing" to circumvent the lower bound. Specifically, the "slicing" pertains to the introduction of two bases, as detailed below (S^v_{ij} denotes a specific combination derived from the input labels, where $i,j \in \{0,1\}$ and $v \in \{1,2\}$):

$$\begin{aligned}&\overbrace{\begin{aligned}&\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^1_{00}\\ &\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^1_{01}\\ &\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^1_{10}\\ &\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_1) \oplus S^1_{11}\end{aligned}}^{\text{the left base}} \qquad \overbrace{\begin{aligned}&\mathsf{H}(B_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^2_{00}\\ &\mathsf{H}(B_1) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^2_{01}\\ &\mathsf{H}(B_0) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^2_{10}\\ &\mathsf{H}(B_1) \oplus \mathsf{H}(A_1 \oplus B_1) \oplus S^2_{11}\end{aligned}}^{\text{the right base}}\end{aligned}$$

In their construction, the garbler uses the left base to produce the left half of the output label and the right base for the right half. This approach can be understood in terms of column-wise processing. Specifically, during the t-th iteration, both the t-th columns of the left and right bases are chosen. These columns are combined using the sub-ciphertext $\boldsymbol{G}[t]$ to yield two bits for the output label. If each $\boldsymbol{G}[t]$ requires only 3 bits, then the total ciphertext length achieved is 1.5κ.

The remaining question is why $\boldsymbol{G}[t]$ requires only 3 bits. The answer lies in the correlation between the two bases. For example, when suitable S^v_{ij} 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[10]

$$\begin{aligned} &\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^1_{00} \quad \oplus \quad \mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^1_{01} \\ &= \mathsf{H}(B_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^2_{00} \quad \oplus \quad \mathsf{H}(B_0) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^2_{10}, \end{aligned}$$

then once the t-th column in the left base is determined, the number of the potential values for the t-th column in the right base will be reduced. For example, without this correlation, the column in the right base might have 16 potential values. However, with the left base column fixed, the correlation will help reduce the potential values for the right column to just 8. As a result, processing the right column requires less ciphertext since the reduced uncertainty dictates a shorter ciphertext.

However, the specific correlation required between the two bases varies based on the permute bits. Therefore, revealing all values of S^v_{ij} ($i, j \in \{0, 1\}$ and $v \in \{1, 2\}$) directly to the evaluator would compromise privacy. To address this, [RR21] employs a technique termed "dicing;" note that, the "dicing" idea was first introduced in [KKS16]. 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 certain 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 is restricted to determining only the values of $S^1_{\alpha\beta}$ and $S^2_{\alpha\beta}$. This marginal view of the bases enables the evaluator to compute the output label while ensuring privacy.

6.2 MODEL-3′: Multi-Column Garbling

Building on the observation from the previous section, we now consider a garbling scheme that includes w bases, where $w \geq 1$. In every operation, the t-th column is extracted from each base, resulting in a total of w columns. Using the mapping function MAP and the sub-ciphertext $\boldsymbol{G}[t]$, these w columns are jointly processed to produce w bits of the output label. The process of each operation is illustrated in Figure 2.

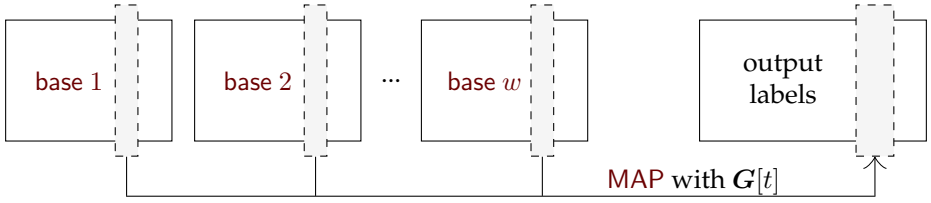

Figure 2: Procedure of the t-th operation.

Our MODEL-3′ builds on MODEL-2. For clarity and simplicity, we've opted not to expand this definition based on MODEL-3, reserving such extension for future works. That is, every column will be processed successfully in our MODEL-3′. Our MODEL-3′ includes all constructions under MODEL-2, as well as [RR21].

Here, we briefly describe the differences in the definition of MODEL-3′ compared to MODEL-2:

- In MODEL-2, the base is not influenced by permute bits. In contrast, in MODEL-3′, permute bits can affect the generation of each base. Therefore, in MODEL-3′, the parameters used to generate the bases are associated with the permute bits (a, b).
- In MODEL-3′, there are w bases with $w \geq 1$. For the u-th base, the mapping function is MAP^u, and the t-th column is $\boldsymbol{M}^u[t]$, where $1 \leq u \leq w$. For every operation, find m-bit sub-ciphertext $\boldsymbol{G}[t]$ such that the following two conditions are satisfied for all $u \in [w]$:
  1. $\mathsf{MAP}^u(\boldsymbol{M}^u[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}^u_{00}(M^u_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{01}(M^u_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{10}(M^u_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{11}(M^u_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z^u_{00}[t] \\ Z^u_{01}[t] \\ Z^u_{10}[t] \\ Z^u_{11}[t] \end{pmatrix}$;
  2. $Z^u_{ab}[t] = Z^u_{(a\oplus 1)b}[t] = Z^u_{a(b\oplus 1)}[t]$.

[10] In the free-XOR setting, we have $\mathsf{H}(A_0 \oplus B_1) = \mathsf{H}(A_1 \oplus B_0)$. For the equation to hold, it is necessary that $S^1_{00} \oplus S^1_{01} = S^2_{00} \oplus S^2_{10}$. By choosing the appropriate S^v_{ij}, this condition can be satisfied.

- In MODEL-3′, some additional ciphertext needs to be generated. The evaluator can decrypt only a portion of them and reconstructs a specific row of each base using these decrypted bits.

In Appendix D.2, we provide a detailed definition of MODEL-3′.

6.3 Lower Bounds

In Section 6.3.1, as before, we demonstrate the lower bound on the ciphertext size required for garbling an AND gate in MODEL-3′. In Section 6.3.2, we attempt to directly extend the technique presented by [RR21] in a model with more constraints than MODEL-3′. However, we find that this approach leads to an impossibility result.

6.3.1 A Lower Bound for MODEL-3′

In this section, we assume that every column of each base would have 16 potential values, regardless of the choice of permute bits. If the base remains unaffected by permute bits, as in MODEL-2 and MODEL-3, then as demonstrated in Theorem 2, each column should encompass 16 potential values. However, according to the definition of MODEL-3′, the generation of each base might be influenced by permute bits, so we cannot discount the possibility of having fewer potential values of each column. To the best of our knowledge, nearly all existing constructions adhere to this assumption, with the exception of [BMR16, KKS16, WmM17]. Both [KKS16] and [WmM17] explicitly exploit the relationships among input labels to ensure two rows in the base are identical, reducing the number of potential column values to 8. Similarly, [BMR16] also makes implicit use of this feature. They leverage the Hamming weight of inputs, noting that the Hamming weight remains consistent for inputs $(0,1)$ and $(1,0)$. When garbling an AND gate, all these constructions necessitate ciphertext of only κ bits. However, these constructions can only garble a single AND gate in isolation, and cannot work for general circuits. Therefore, we exclude them from our consideration[11]. In other words, our lower bound in this section isn't applicable to them.

Coming back to our primary focus, our goal is to reduce the ciphertext length by leveraging the correlations between multiple bases. But the pressing questions remain: What precisely are these correlations? And how can we effectively harness them? Tackling these questions is pivotal in setting a substantial lower bound.

Definition 2 (Column correlation). *Consider a garbling scheme with w bases, where $w > 1$. For any two distinct bases $\boldsymbol{M}^i, \boldsymbol{M}^j$, where $i, j \in [w]$, we say $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$ are **column-correlated**, if there exist unique non-zero vectors $\boldsymbol{P}^i, \boldsymbol{P}^j$ of lengths 4 with entries in $\mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})$ such that, $\langle \boldsymbol{P}^i, \boldsymbol{M}^i \rangle = \langle \boldsymbol{P}^j, \boldsymbol{M}^j \rangle$ is always satisfied. Furthermore, if $i < j$, then we say $\boldsymbol{M}^i$ is an **ancestor** of $\boldsymbol{M}^j$.*

We will use the term "**column-correlated**" to describe the correlation between two bases. We emphasize that this definition is justified. Each base is generated through a linear combination of random oracle outputs and input labels, so it is reasonable to define the correlation between two bases using the inner product. Using the construction in [RR21] as an example, we assume that the garbler selects permute bits $(a, b) = (0, 0)$. The garbler generates two bases, namely $\boldsymbol{M}^1 = (M^1_{00}, M^1_{01}, M^1_{10}, M^1_{11})$ and $\boldsymbol{M}^2 = (M^2_{00}, M^2_{01}, M^2_{10}, M^2_{11})$, where all elements are in $\mathrm{GF}(2^{\lceil \frac{\kappa}{2} \rceil})$. There exists a unique $\boldsymbol{P}^1 = (1, 1, 0, 0)$ and $\boldsymbol{P}^2 = (1, 0, 1, 0)$ such that for all potential values of $\boldsymbol{M}^1$ and $\boldsymbol{M}^2$, $\langle \boldsymbol{P}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle$ holds.

If two bases are **column-correlated**, then once the value of the first base is determined, the potential values of the second base will decrease.

Lemma 1. *In a garbling scheme with w bases adhering to MODEL-3′, suppose two bases $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$ (where $1 \le i < j \le w$) are **column-correlated**. Once $\boldsymbol{M}^i$ is determined, the number of potential values for each column of $\boldsymbol{M}^j$ is halved, reducing from 16 to 8.*

The proof of Lemma 1 can be found in Appendix D.3.

In MODEL-3′, the garbler selects one column from each of the w bases to process and derives w bits of the output label. For any two bases satisfying **column-correlated**, the column taken from the first base

[11] Even so, this observation is quite intriguing. The question of how to halve the potential values of a column from 16 to 8 while retaining self-composability remains an open question.

has $2^4 = 16$ potential values. As demonstrated in the proof of Theorem 1, this necessitates at least a 2-bit ciphertext. Lemma 1 elucidates that the column extracted from the second base has only 8 potential values.

In the definition of MODEL-3′, the ciphertext consists of two parts: the main ciphertext $\boldsymbol{G}$ and the additional ciphertext. Given that the length of the additional ciphertext remains constant, our primary focus lies on the length of $\boldsymbol{G}$. We now proceed to establish the lower bound.

Theorem 3. *Consider an ideally secure garbling scheme in* MODEL-3′ *with* w bases. *Assume that each* base *has at most one* ***ancestor****. If* υ *pairs of* bases *are* ***column-correlated****, then for garbling an* AND *gate, the ciphertext length* $|\boldsymbol{G}|$ *must satisfy* $|\boldsymbol{G}| \geq (2 - \upsilon/w) \cdot \kappa$, *where* $\upsilon \leq w - 1$.

If $\upsilon = w - 1$, then a secure garbling scheme requires at least $(1 + 1/w) \cdot \kappa$ bits of ciphertext to handle an AND gate. Below we provide a proof sketch. The detailed proof can be found in Appendix D.4.

Proof Sketch. From the given conditions, there are $w - \upsilon$ bases without **ancestor**, and υ bases with **ancestor**. For the column taken from the bases without **ancestor**, there are 16 potential values. For the column taken from the bases with **ancestor**, there are 8 possible values.

For the former case, at least 2-bit sub-ciphertext is needed as in Theorem 1. For the latter case, at least 1-bit sub-ciphertext is required, based on the intuition that the column is correlated with its **ancestor**. We can conclude that each processing requires at least $(2 \cdot (w - \upsilon) + 1 \cdot \upsilon)$ bits of ciphertext. And a total of κ/w operations are required, so $|\boldsymbol{G}| \geq (2 \cdot (w - \upsilon) + 1 \cdot \upsilon) \cdot \kappa/w = (2 - \upsilon/w) \cdot \kappa$.

□

Remark 4. The understanding of the "sliced" technique in general remains an open question. Essentially, increasing slices can be seen as introducing more bases. The obtained lower bound suggests that incorporating additional bases could potentially lead to improved constructions. Therefore, a straightforward extension of the sliced technique in [RR21] may help us develop constructions with ciphertext less than 1.5κ bits.

6.3.2 Limitation of the Rosulek-Roy Technique

As mentioned, a natural idea is to extend the approach of [RR21] (or jumping ahead, our Construction #3 in the next section) by introducing multiple bases and leveraging the correlations between them, thereby surpassing the optimal result of 1.5κ bits for garbling an AND gate. However, surprisingly, this method is not feasible. In other words, if we impose some reasonable constraints on MODEL-3′ (such as compatibility with free-XOR and the mapping function MAP being linear), using 1.5κ bits to garble an AND gate is already optimal. We remark that, in a concurrent work [BK24], Baek and Kim also presented a lower bound proof using an algebraic approach.

Recall that in [RR21], the authors introduce two bases and use the correlation between them to reduce the ciphertext length. Specifically, for a column in the first base, 2-bit sub-ciphertext is needed, and only 1-bit sub-ciphertext is needed for the corresponding column in the second base, since these two bases are **column-correlated**. Therefore, an intuitive extension is to introduce three bases, assuming that the first base and second base are **column-correlated**, as well as the first base and third base. According to Theorem 3, this would require only $4(= 2+1+1)$ bits of sub-ciphertext to handle three columns, with a total ciphertext length of $4\kappa/3$ bits. Unfortunately, such an extension will leak privacy.

Informally, we find that, building upon the technique outlined in [RR21], where a garbling scheme is depicted as a linear equation system[12], introducing more bases (slices) does not yield constructions requiring fewer than 1.5κ-bit ciphertext for garbling an AND gate.

Theorem 4. *If a secure garbling scheme can be represented by a linear equation system, and the correlation between* bases *can be expressed using Definition 2, then this scheme requires at least* $3/2 \cdot \kappa + O(1)$ *bits of ciphertext length for garbling an* AND *gate.*

A proof sketch can be found below. In Appendix D.5, we provide a detailed proof.

[12] More details about this notion can be found in Appendix D.5.

Proof Sketch. To demonstrate this impossibility result, we initially establish that the introduction of three bases necessitates a ciphertext length of at least $3\kappa/2$ bits. In other words, we will show that, if a garbling scheme employs three bases and the ciphertext length is less that $3\kappa/2$ bits, it violates the *privacy* requirement.

In [RR21], Rosulek and Roy proposed a garbling scheme by transforming the problem into solving a linear equation system with compatibility of free-XOR. Following their technical approach, we divide the proof into three steps:

1. Determine the form of hash queries (represented by the matrix $\boldsymbol{\mathcal{M}}$) and the form of the linear mapping function MAP (represented by the matrix $\boldsymbol{\mathcal{V}}$). To ensure the solvability of the linear equation system, the column spaces of matrices $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{V}}$ must be identical.

2. Establish correlations within each base and among the three bases. These correlations arise from three aspects. Firstly, to accommodate free-XOR, as observed in our Construction #2, the XOR of four rows within a base must equal Δ. Secondly, to achieve a ciphertext length of less than 1.5κ, there must be at least two correlations among the three bases. For example, assuming the first base and the second base are **column-correlated**, and the first base and the third base are **column-correlated**. Lastly, the linear equation system must be solvable. These correlations result in multiple relational equations involving the 12 rows originating from the three bases.

3. Deduce partial information about the true values of wires based on the marginal view. Specifically, if the evaluator obtains A_0 and B_0, she can reconstruct the first row of each base. (Recall that we use the "dicing" technique, so the evaluator can only learn the content of the first row of each base.) However, we find that the evaluator, relying on her marginal view and the relational equations established in the previous step, can infer partial information about the true values of wires, violating the privacy requirement.

Therefore, if we use three bases, it is impossible to garble an AND gate with a ciphertext length of less than $3\kappa/2$. As a corollary, we can extend this impossibility to multiple bases. Again, please see the full proof in Appendix D.5. □

In a concurrent work, Ashur et al. [AHS24] introduced an intriguing method to extend the approach in [RR21], asserting the achievement of garbling an AND gate with only $4\kappa/3$ bits. However, our proof technique highlights a gap between their construction and security proof, suggesting that their scheme is not secure. In another recent independent work, Kim [Kim24] also identified the security issue in Ashur et al.'s scheme. More details can be found in Appendix D.6. As a result, the question of surpassing the 1.5κ bound is still open.

Remark 5. Theorem 4 suggests that introducing more bases and correlating them would not lead to better constructions, implying that the lower bound we proved in Theorem 3 might be too conservative. However, this is not the case. Firstly, our MODEL-3′ allows non-linear operations, making it impossible for representation through a linear equation system. Secondly, we aren't confined to specific hash queries or the free-XOR setting as in [RR21] (which can be found in Appendix D.5 in detail). Therefore, it is still possible that a garbling scheme meets our lower bound in Theorem 3.

6.4 New Construction #3: 1.5κ with Free-XOR

Following the idea of **majority voting**, weobtained a scheme that requires 2κ bits to garble an AND gate, maintaining compatibility with free-XOR. Utilizing the same idea, we can further optimize our construction to necessitate only 1.5κ bits for garbling an AND gate. This puts our performance on par with [RR21].

From the previous section, it's clear that to achieve a reduced ciphertext size, leveraging correlations between multiple bases is crucial. Consequently, we introduce two bases here, where the left base is derived

directly from the Construction #2 in Section 4.3:

$$\overbrace{\begin{array}{l}\mathsf{H}(A_0)\oplus\mathsf{H}(B_0)\oplus S^1_{00}\\ \mathsf{H}(A_0)\oplus\mathsf{H}(B_1)\oplus S^1_{01}\\ \mathsf{H}(A_1)\oplus\mathsf{H}(B_0)\oplus S^1_{10}\\ \mathsf{H}(A_1)\oplus\mathsf{H}(B_1)\oplus S^1_{11}\end{array}}^{\text{the left base}} \qquad \overbrace{\begin{array}{l}\mathsf{H}(A_0)\oplus\mathsf{H}(A_0\oplus B_0)\oplus S^2_{00}\\ \mathsf{H}(A_0)\oplus\mathsf{H}(A_0\oplus B_1)\oplus S^2_{01}\\ \mathsf{H}(A_1)\oplus\mathsf{H}(A_1\oplus B_0)\oplus S^2_{10}\\ \mathsf{H}(A_1)\oplus\mathsf{H}(A_1\oplus B_1)\oplus S^2_{11}\end{array}}^{\text{the right base}}$$

Here, each S^u_{ij} (where $i,j\in\{0,1\}$, $u\in\{1,2\}$) represents some values derived from the input labels, which are undetermined.

We emphasize our primary construction idea here. From the left base, we choose a column and employ the **majority voting** approach, necessitating a 2-bit sub-ciphertext as before, to specify which position should be flipped. Next, from the right base, we pick another column. By carefully determining the composition of each S^u_{ij}, we can establish a correlation between the two bases. As a result, **majority voting** can be performed using only a 1-bit sub-ciphertext for the second column.

However, unlike the first two constructions, in this construction, the generation of each base depends on permute bits, leading to several technical challenges. To address these, we introduce a unified flipping method. For the specific details and the security proof for this construction, refer to Appendices D.7 and D.8.

Remark 6. Compared to the state-of-the-art ([RR21]), our construction incurs the same overhead. However, it exhibits more non-linearity, which might better resist certain attacks, such as side-channel attacks [LH23]. These potential advantages are left to be tested by future works.

7 Toward a Unified Framework

In this section, we will explore more variants of the design models.

7.1 MODEL-0: Garbling with Random Output Labels

In the definition of MODEL-2, there are no constraints on the relationship between output and input labels. In the classical garbling scheme, output labels are selected independently, meaning they are picked randomly before generating the ciphertext. Reducing the degrees of freedom in output label choice can decrease the ciphertext size. However, the ability to freely select output labels has its benefits. For instance, it improves the garbler's efficiency when generating the garbled circuits by permitting parallel processing of each gate. Discussing this from a theoretical viewpoint is also significant.

First, we introduce a variant of MODEL-2, referred to as MODEL-0. In this variant, output labels are selected independently. For brevity, we won't explore the detailed definition of MODEL-0. We assume that MODEL-0 and MODEL-2 are identical in all aspects, except for the independent selection of output labels in MODEL-0.

Theorem 5. *Any ideally secure garbling scheme for an* AND *gate that satisfies the* MODEL-0 *must have* $|\boldsymbol{G}|\geq 4\kappa$.

The proof of Theorem 5 can be found in Appendix E.1. Interestingly, by processing multiple columns together, we can achieve an even lower bound.

Lemma 2. *For ℓ columns of the output labels, the total number of potential values is given by $2^{2\ell+2}-3\cdot 2^{\ell}$, where $1\leq\ell\leq\kappa$.*

We provide the proof of Lemma 2 in Appendix E.2.

From Lemma 2, we can infer that if ℓ columns are processed simultaneously, then the required ciphertext length is greater than $\log(2^{2\ell+2}-3\cdot 2^{\ell})$. When we amortize this length over each column, it becomes $\log(2^{2\ell+2}-3\cdot 2^{\ell})/\ell$. Hence, the total ciphertext length is $(\log(2^{2\ell+2}-3\cdot 2^{\ell})/\ell)\cdot\kappa>2\kappa$. This suggests that by processing multiple columns simultaneously, we could potentially design a garbling scheme with a ciphertext length approaching 2κ.

7.2 MODEL-3″: Garbling an AND Gate with Multiple Inputs

Up to now, we have considered an AND gate with two input wires and one output wire. However, for an AND gate with n inputs and one output, where $n \geq 2$, the current optimal approach is to break it down into $n-1$ two-input AND gates. Then, the optimal garbling technique for a two-input AND gate is applied. While constructing a better scheme remains a challenge, we can establish a lower bound using a similar proof technique.

We introduce a variant of MODEL-2, denoted as MODEL-3″. The main difference between MODEL-3″ and MODEL-2 is that MODEL-3″ doesn't limit the number of inputs. Given its intuitive nature, we won't delve into the detailed definition of MODEL-3″. It's worth mentioning that the base in MODEL-3″ has 2^n rows.

Theorem 6. *Any ideally secure garbling scheme for an* AND *gate with n input wires that satisfies the* MODEL-3″ *must have $|\boldsymbol{G}| \geq n\kappa$.*

The proof of Theorem 6 can be found in Appendix E.3.

7.3 Summary and Extensions

In this work, we provide a detailed exploration of MODEL-0, MODEL-1, MODEL-2, MODEL-3, MODEL-3′, and MODEL-3″, along with their respective lower bounds. First, we summarize these models and discuss the reasons behind their development.

MODEL-0 is the closest to the classical Yao scheme (that is why we named it MODEL-0). This model has the following characteristics:

1. It handles a single binary gate, specifically a two-input, one-output AND gate.
2. The mapping function MAP cannot be rejected.
3. It only allows a single base, so there is no correlation between bases.
4. The mapping function MAP is linear, meaning it can be represented using matrix multiplication or vector inner product.
5. Input and output labels are chosen independently, meaning output labels do not depend on input labels.

In this case, we find that we need at least 4κ bits of ciphertext to garble an AND gate. Interestingly, the features listed above can also be considered as *constraints*. Therefore, removing some of these constraints may allow us to achieve a more favorable lower bound, suggesting more efficient constructions.

For example, we can develop MODEL-1 by removing the fifth constraint. In this model, output labels may depend on input labels. Thus, by reducing the degree of freedom, we can achieve a better lower bound and more efficient construction. Specifically, we only need 2κ bits to garble an AND gate.

Building on MODEL-1, if we remove the fourth constraint, we arrive at MODEL-2. However, we find that even with non-linear mappings allowed, the 2κ lower bound still applies. Going further, by removing the third constraint from MODEL-2, we develop MODEL-3′, which achieves a bound of $(1+1/w)\cdot\kappa$ and results in a construction that requires only 1.5κ bits for garbling an AND gate.

In Figure 3, we summarize all the models discussed in this work. We remark that a more appropriate way to name models might be to use the gray numbers from Figure 3, rather than simply naming them as MODEL-0, MODEL-1, etc.

Indeed, we can imagine and propose even more variants, as illustrated in Figure 4.

We provide brief descriptions of these potential models:

- MODEL-2′: Retaining the linear operations on each column as in MODEL-1, we introduce additional bases to exploit the correlations among them, thereby reducing the ciphertext length. Strictly speaking, [RR21] falls under MODEL-2′, but the construction we introduce in Section 6.4 does not belong to this model.
- MODEL-4: This model represents a combination of MODEL-3 and MODEL-3′. That is to say, in this model, we allow for multiple bases and mapping with rejection. Moreover, we can make even more interesting extensions, such as allowing each base to have distinct rejection probability.

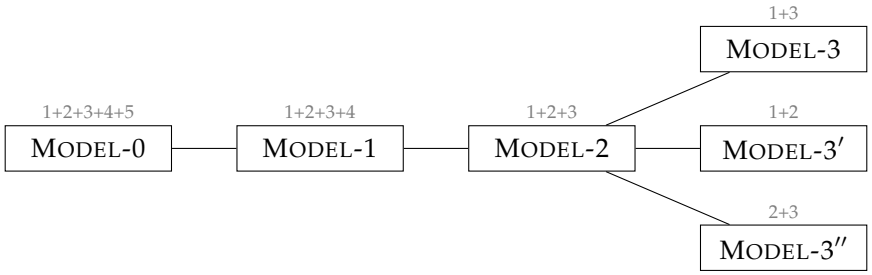

Figure 3: A hierarchy of garbling models discussed in this work. The numbers in gray on each box represent the constraints for each model. The models on the right strictly include those on the left.

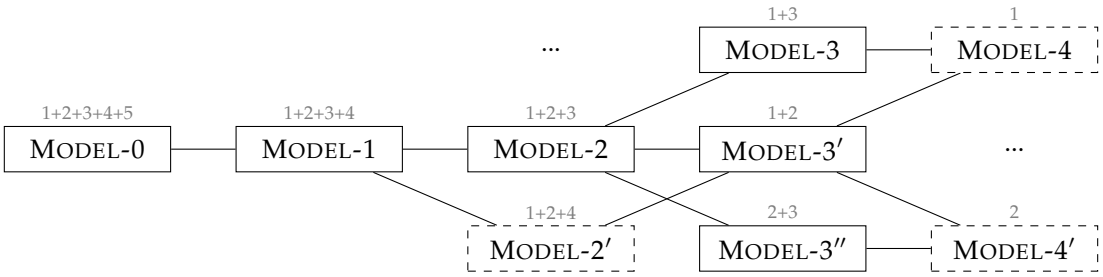

Figure 4: More potential variants of garbling models. Solid boxes represent the models discussed in this work, while dashed boxes indicate potential extensions. If a line connects two boxes, it indicates that the model in the right box strictly encompasses the model in the left box.

- MODEL-4′: By combining MODEL-3′ and MODEL-3″, we obtain this new model. In other words, we allow the presence of multiple bases to garble a multi-input AND gate. Recent work by Ashur et al. [AHS23] attempted to find new constructions within this model, but their efforts were unsuccessful.
- ...

Can more efficient constructions be derived in the aforementioned variants? What are their lower bounds? These remain open questions, which we leave for future work.

8 Related Works

Following Yao's introduction of garbled circuits [Yao86], Lindell and Pinkas [LP09] provided a comprehensive description and security proof for Yao's protocol. Bellare et al. [BHR12b] later abstracted garbled circuits into a primitive, giving them a flexible syntax and multiple security definitions.

In optimizing garbled circuits, much effort has been on enhancing the efficiency of garbling a single gate against semi-honest adversaries. Beaver et al. [BMR90] introduced the point-permute technique, which lets the evaluator choose the right ciphertext using visible "color bits" without decrypting each one. Naor et al. [NPS99] utilized the freedom of randomly choosing output labels to present the row reduction technique, reducing the ciphertext length for garbling an AND gate from 4κ to 3κ. Subsequent optimizations by [PSSW09] and [GLNP15] brought this down to 2κ. Kolesnikov and Schneider's free-XOR technique [KS08] leveraged linear relationships to make the garbling of an XOR gate cost-free. However, its security relies on circular security, not standard assumptions, as detailed in [CKKZ12, GKWY20]. The fleXOR by Kolesnikov et al. [KMR14] expanded on free-XOR, offering several advantages. Zahur et al. [ZRE15] introduced the half gates, which require only 2κ bits for an AND gate and are compatible with free-XOR. Acharya et al. [AAC+23] took a novel approach by conceptualizing gate functionality as a unified entity rather than encrypted rows, though it lacked efficiency benefits. Currently, the state-of-the-art method is by Rosulek and

Roy [RR21], which uses just 1.5κ-bit ciphertext for an AND gate and remains compatible with free-XOR.

Some methods [BMR16, KKS16, WmM17] achieved κ bits for an AND gate, but they are restricted to garbling an individual AND gate in isolation and lack self-composability, thus falling outside the scope of our work.

Other research has pursued security against malicious adversaries [WRK17, KRRW18] or focused on adaptive adversaries [BHR12a, HJO+16]. As garbling gate-by-gate optimization opportunities wane, studies increasingly target improving larger circuits [HK20, HK21] or devising arithmetic garbling schemes [AIK11, BMR16, BLLL23].

References

[AAC+23] Anasuya Acharya, Tomer Ashur, Efrat Cohen, Carmit Hazay, and Avishay Yanai. A new approach to garbled circuits. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, volume 13906 of *Lecture Notes in Computer Science*, pages 611–641. Springer, 2023.

[AHS23] Tomer Ashur, Carmit Hazay, and Rahul Satish. Garbling 3-input and gates. In *CFAIL 2023: The 5th Conference for Failed Approaches and Insightful Losses in Cryptology*, 2023. https://www.cfail.org/_files/ugd/6a3e24_425474ea0b834ac7aed0b3371acaec60.pdf.

[AHS24] Tomer Ashur, Carmit Hazay, and Rahul Satish. On the feasibility of sliced garbling. Cryptology ePrint Archive, Paper 2024/389, 2024. https://eprint.iacr.org/2024/389.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, Heidelberg, May / June 2010.

[BHR12a] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

[BHR12b] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

[BK24] Chunghun Baek and Taechan Kim. Can We Beat Three Halves Lower Bound? (Im)Possibility of Reducing Communication Cost for Garbled Circuits. 2024. Manuscript.

[BLLL23] Marshall Ball, Hanjun Li, Huijia Lin, and Tianren Liu. New ways to garble arithmetic circuits. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 3–34. Springer, Heidelberg, April 2023.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[BMR16] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for Boolean and arithmetic circuits. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 565–577. ACM Press, October 2016.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 39–53. Springer, Heidelberg, March 2012.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841. IEEE Computer Society Press, May 2020.

[GLNP15] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 567–578. ACM Press, October 2015.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, August 2016.

[HK20] David Heath and Vladimir Kolesnikov. Stacked garbling - garbled circuit proportional to longest execution path. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 763–792. Springer, Heidelberg, August 2020.

[HK21] David Heath and Vladimir Kolesnikov. One hot garbling. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 574–593. ACM Press, November 2021.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, November 2013.

[Kim24] Taechan Kim. Slice more? It leaks: Analysis on the paper "On the Feasibility of Sliced Garbling". *IACR Cryptol. ePrint Arch.*, 2024. Manuscript.

[KKS16] Carmen Kempka, Ryo Kikuchi, and Koutarou Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 967–997. Springer, Heidelberg, December 2016.

[KMR14] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, Heidelberg, August 2014.

[KRRW18] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 365–391. Springer, Heidelberg, August 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.

[LH23] Itamar Levi and Carmit Hazay. Garbled circuits from an SCA perspective free XOR can be quite expensive. . *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):54–79, 2023.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In Stuart I. Feldman and Michael P. Wellman, editors, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 129–139. ACM, 1999.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, Heidelberg, December 2009.

[RR21] Mike Rosulek and Lawrence Roy. Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 94–124, Virtual Event, August 2021. Springer, Heidelberg.

[WmM17] Yongge Wang and Qutaibah m. Malluhi. Reducing garbled circuit size while preserving circuit gate privacy. Cryptology ePrint Archive, Report 2017/041, 2017. https://eprint.iacr.org/2017/041.

[WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 21–37. ACM Press, October / November 2017.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, April 2015.

A Supplemental Materials for Section 3

A.1 Linear Garbling Model

Zahur et al. introduced the linear garbling model in [ZRE15]. The authors observed that in all existing constructions at the time, operations are linear with the exception of the point-permute and queries to the random oracle. Formally, they considered a garbling scheme to be linear if it takes the following form:

- Garbling algorithm Gb:
  Parameterized by integers m, r, q, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{C}_{ab,i} \mid a,b,i \in \{0,1\}\}$, $\{\boldsymbol{G}^i_{ab} \mid a,b \in \{0,1\}, i \in [m]\}$, random oracle RO. Each vector is of length $r+q$ with entries in $\mathrm{GF}(2^\kappa)$. The RO is defined as $\{0,1\}^* \to \mathrm{GF}(2^\kappa)$.

1. For $i \in [r]$, choose $R_i \leftarrow \mathrm{GF}(2^\kappa)$.
2. Make q distinct queries to the RO, which can be determined based on the R_i values. Let $Q_1, \ldots, Q_q$ denote the responses to these queries, and define $\boldsymbol{S} := (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$.
3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.
4. Compute $A_0 := \langle \boldsymbol{A}_0, \boldsymbol{S} \rangle$, $A_1 := \langle \boldsymbol{A}_1, \boldsymbol{S} \rangle$, $B_0 := \langle \boldsymbol{B}_0, \boldsymbol{S} \rangle$, $B_1 := \langle \boldsymbol{B}_1, \boldsymbol{S} \rangle$, $C_0 := \langle \boldsymbol{C}_{ab,0}, \boldsymbol{S} \rangle$, $C_1 := \langle \boldsymbol{C}_{ab,1}, \boldsymbol{S} \rangle$. Then $A_0||0$ and $A_1||1$ are labels for one input wire, and $B_0||0$ and $B_1||1$ for the other. Subscripts indicate the color bits, with A_a and B_b representing FALSE. C_0 and C_1 are the output wire labels with C_0 corresponding to FALSE.
5. For $i \in [m]$, compute $G_i := \langle \boldsymbol{G}^i_{ab}, \boldsymbol{S} \rangle$, and the values $G_1, \ldots, G_m$ comprise the ciphertext of garbled circuit.

- Encoding algorithm En:
  Given inputs $x_a, x_b \in \{0,1\}$, 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 Ev:
  Parameterized by integers m, q, vectors $\{\boldsymbol{V}_{\alpha\beta} \mid \alpha, \beta \in \{0,1\}\}$, random oracle RO. The length of each vector is $m + q + 2$.
  1. The inputs are wire labels $A_\alpha \| \alpha, B_\beta \| \beta$ and the ciphertext $G_1, \ldots, G_m$.
  2. Make q distinct queries to the RO, which can be determined based on the input wire labels. Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, and define $\boldsymbol{T} := (A_\alpha, B_\beta, Q'_1, \ldots, Q'_q, G_1, \ldots, G_m)$.
  3. Compute $\langle \boldsymbol{V}_{\alpha\beta}, \boldsymbol{T} \rangle$ as the output label.

Based on the above model, they derived the following theorem.

Theorem 7 (Theorem 3 in [ZRE15])**.** *Every ideally secure garbling scheme for* AND *gates that is linear in the above sense must have $m \geq 2$. That is, the garbled gate consists of at least 2κ bits.*

B Supplemental Materials for Section 4

B.1 Formal Definition of MODEL-2

We formally define MODEL-2 as follows:

- Garbling algorithm Gb:
  Parameterized by integers m, r, q, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{M}_{ij} \mid i, j \in \{0,1\}\}$, random oracle RO and mapping function MAP. Each vector is of length $r + q$ with entries in $\mathrm{GF}(2^\kappa)$. The RO is mapped as $\{0,1\}^* \to \mathrm{GF}(2^\kappa)$. The MAP is defined as $\mathrm{GF}(2^4) \times \mathrm{GF}(2^m) \to \mathrm{GF}(2^4)$, and it can be decomposed into four sub-functions ($\mathsf{MAP}_{00}, \mathsf{MAP}_{01}, \mathsf{MAP}_{10}, \mathsf{MAP}_{11}$).
  1. For $i \in [r]$, choose $R_i \leftarrow \mathrm{GF}(2^\kappa)$.
  2. Make q distinct queries to the RO, which can be determined based on the R_i values. Let $Q_1, \ldots, Q_q$ denote the responses to these queries, and define $\boldsymbol{S} := (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$.
  3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.
  4. Compute $A_0 := \langle \boldsymbol{A}_0, \boldsymbol{S} \rangle$, $A_1 := \langle \boldsymbol{A}_1, \boldsymbol{S} \rangle$, $B_0 := \langle \boldsymbol{B}_0, \boldsymbol{S} \rangle$, $B_1 := \langle \boldsymbol{B}_1, \boldsymbol{S} \rangle$. Then $A_0||0$ and $A_1||1$ are labels for one input wire, and $B_0||0$ and $B_1||1$ for the other. Subscripts indicate the color bits, with A_a and B_b representing FALSE.
  5. For $i, j \in \{0,1\}$, compute $M_{ij} := \langle \boldsymbol{M}_{ij}, \boldsymbol{S} \rangle$. Then $(M_{00}, M_{01}, M_{10}, M_{11})^\top$ is the base. The t-th column of the base, represented by $\boldsymbol{M}[t]$, is composed of 4 bits and can be interpreted as $(M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$.
  6. For $t \in [\kappa]$, find m-bit ciphertext $\boldsymbol{G}[t]$ such that the following two conditions are satisfied:

(a) $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}_{00}(M_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{01}(M_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{10}(M_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{11}(M_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix}$;

(b) $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$.

Then use $Z_{ab}[t]$ as one bit of the output label C_0 and $Z_{(a\oplus 1)(b\oplus 1)}[t]$ as one bit of the output label C_1, where C_0 corresponds to FALSE. The sub-ciphertext $\boldsymbol{G}[t]$, with a length of m, represents the t-th part of the whole ciphertext $\boldsymbol{G}$.

- Encoding algorithm En:
  Given inputs $x_a, x_b \in \{0,1\}$, 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 Ev:
  Parameterized by integers q, vectors $\{\boldsymbol{V}_{\alpha\beta} \mid \alpha, \beta \in \{0,1\}\}$, random oracle RO and mapping function MAP. The length of each $\boldsymbol{V}_{\alpha\beta}$ is $2+q$, with entries in $\mathrm{GF}(2^\kappa)$.
  1. The inputs are wire labels $A_\alpha || \alpha$, $B_\beta || \beta$ and the ciphertext $\boldsymbol{G}$.
  2. Make q distinct queries to the RO, which can be determined based on the input wire labels. Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, and define $\boldsymbol{T} := (A_\alpha, B_\beta, Q'_1, \ldots, Q'_q)$.
  3. Compute $V_{\alpha\beta} := \langle \boldsymbol{V}_{\alpha\beta}, \boldsymbol{T} \rangle$, and use $V_{\alpha\beta}[t]$ to denote the t-th bit of $V_{\alpha\beta}$.
  4. For $t \in [\kappa]$, use $Z_{\alpha\beta}[t] := \mathsf{MAP}_{\alpha\beta}(V_{\alpha\beta}[t], \boldsymbol{G}[t])$ as the t-th bit of the output label.

B.2 Proof of Theorem 1

Theorem 1. *Any ideally secure garbling scheme for an* AND *gate conforming to* MODEL-2 *must satisfy* $|\boldsymbol{G}| \geq 2\kappa$.

Proof. In the definition of MODEL-2, we perform column-wise operations, producing one bit of the output label for each operation. Each operation requires a sub-ciphertext $\boldsymbol{G}[t]$ with a length of m, where $1 \leq t \leq \kappa$. To prove $|\boldsymbol{G}| \geq 2\kappa$, we need only demonstrate that $m \geq 2$.

To ensure the correctness of garbling an AND gate, the target vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ in the computation of $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ should satisfy the condition $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$. For permute bits $a, b \in \{0,1\}$, there are four cases. In each case, the vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ has four potential values, detailed in Table 3. For example, in the case where $(a,b) = (0,0)$, A_0 and B_0 are set to FALSE. Therefore, the vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ should belong to the set $\{(0,0,0,0), (0,0,0,1), (1,1,1,1), (1,1,1,0)\}$.

Table 3: The potential values of $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ in different cases.

(a,b)	potential values of $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$
$(0,0)$	$\mathcal{S}_{00} = \{(0,0,0,0),\ (0,0,0,1),\ (1,1,1,1),\ (1,1,1,0)\}$
$(0,1)$	$\mathcal{S}_{01} = \{(0,0,0,0),\ (0,0,1,0),\ (1,1,1,1),\ (1,1,0,1)\}$
$(1,0)$	$\mathcal{S}_{10} = \{(0,0,0,0),\ (0,1,0,0),\ (1,1,1,1),\ (1,0,1,1)\}$
$(1,1)$	$\mathcal{S}_{11} = \{(0,0,0,0),\ (1,0,0,0),\ (1,1,1,1),\ (0,1,1,1)\}$

All potential output values in Table 3 compose a set $\mathcal{S} = \mathcal{S}_{00} \cup \mathcal{S}_{01} \cup \mathcal{S}_{10} \cup \mathcal{S}_{11}$ which can be divided into two subsets. The first subset is $\mathcal{S}_1 = \{(0,0,0,0),\ (1,1,1,1)\}$, and the second subset is $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$. For a given vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$, if it belongs to $\mathcal{S}_1$, then the t-th bit of C_0 and the t-th bit of C_1 will be the same, otherwise they will differ. To ensure privacy, the bitwise XOR of the two output labels, namely $C_0 \oplus C_1$, should appear random. In other words, the probability that the same bit position in C_0 and C_1 is identical should be $1/2$. Therefore, we have:

$$\Pr\left[\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) \in \mathcal{S}_1\right] = \Pr\left[\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) \in \mathcal{S}_2\right] = \frac{1}{2}. \tag{1}$$

Recall that in the MODEL-2 definition, the generation of the base is independent of the choice of permute bits. Similarly, the function MAP doesn't depend on these bits either. For a fixed $\boldsymbol{M}[t]$, changing $\boldsymbol{G}[t]$ should let the output of $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ cover all four cases. This means that the set of possible outputs from $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ when changing $\boldsymbol{G}[t]$, should not only be a subset of $\mathcal{S}$ but also intersect with each $\mathcal{S}_{ab}$ for $a, b \in \{0, 1\}$.

We use a proof by contradiction, starting with the assumption $m = 1$. If this is the case, then $\boldsymbol{G}[t]$ can have only two potential values. Therefore, for each $\boldsymbol{M}[t]$, the mapping function $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ can produce, at most, two unique values, which we denote as the set $\mathcal{W}$. To ensure correctness, regardless of the values of permute bits $a, b \in \{0, 1\}$, the function $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ must yield a correct output. If $\mathcal{W} \cap \mathcal{S}_1$ is empty, then $\mathcal{W}$ cannot cover all rows in Table 3.

Therefore, among the four possible values of (a, b), in at least three of them, $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ will be in $\mathcal{S}_1$. This violates the probability in Equation 1. Consequently, the assumption $m = 1$ is untenable, leading to the conclusion that $m \geq 2$. This completes the proof. □

B.3 Garbling an XOR Gate

We briefly outline the procedure for garbling an XOR gate, which also requires a 2-bit sub-ciphertext for each column. For the t-th column, the garbler aims to ensure that $M_{00}[t]$ and $M_{11}[t]$, as well as $M_{01}[t]$ and $M_{10}[t]$, have the same value after flipping. This can be achieved using a 2-bit ciphertext. Specifically, the first bit indicates whether $M_{00}[t]$ should be flipped, while the second bit indicates whether $M_{01}[t]$ should be flipped. Table 4 provides a detailed construction of the sub-ciphertext $\boldsymbol{G}[t]$.

Table 4: Sub-ciphertext generation for the t-th column when garbling an XOR gate.

$M_{00}[t]$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$M_{01}[t]$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$M_{10}[t]$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$M_{11}[t]$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\boldsymbol{G}[t]$	00	10	01	11	01	11	00	10	10	00	11	01	11	01	10	00

B.4 Security Proof of Construction #2

We present a formal description and security proof for the second construction in Section 4.3. Based on this, one can directly derive a formal description and security proof for the first construction.

Formal Description. Our garbling scheme is depicted in Figures 5, 6, 7, and 8. The boolean circuit function f can be decomposed into four parts: $(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type})$. The 'inputs' refers to the indices of the input wires of f, while 'outputs' provides the indices of the output wires. The 'in' specifies the indices of the two input wires of a gate, and 'type' indicates the type of a gate. The i-th wire has two labels, $(W_i, W_i \oplus \Delta)$, and a permute bit π_i. We assume that the color bit of each label is its least significant bit, and the color bit of W_i is $\mathsf{lsb}(W_i) = 0$, while that of $W_i \oplus \Delta$ is $\mathsf{lsb}(W_i \oplus \Delta) = 1$. Additionally, $W_i \oplus \pi_i \Delta$ is the label representing false.

We use the function $\mathsf{H} : \{0, 1\}^\kappa \times \mathbb{Z} \to \{0, 1\}^\kappa$ to denote a hash function suitable for this scheme, which accepts the index of the gate as a tweak. We did not specify this detail in the previous informal description.

We introduce a private algorithm, MajVote, to represent the majority voting process. For each AND gate, the garbler generates a base by querying the oracle H with input labels. Then, column by column, each operation produces one bit of the output label and two bits of the ciphertext. Each column contains four bits, and any of these bits could be flipped, yielding four possible flip cases.

For every output wire of f, the garbler computes hashes of the wire labels. These hashes serve as decoding information. We employ $2|f| + i$ as the tweak for these queries, where i is the index of the output wire. This ensures there's no duplication of the oracle queries used in generating the base.

For the evaluator, she processes each gate in sequence. We assume that for an AND gate, she has two "active" input labels A and B. The color bits of these labels are $\alpha = \mathsf{lsb}(A)$ and $\beta = \mathsf{lsb}(B)$ respectively. As

Garbling algorithm $\mathsf{Gb}(1^\kappa, f)$:

$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$\Delta \leftarrow \{0,1\}^{\kappa-1}$
$\Delta = \Delta||1$
for $i = 1$ to inputs :
  $W_i \leftarrow \{0,1\}^{\kappa-1}$
  $W_i = W_i||0$
  $\pi_i \leftarrow \{0,1\}$
for $i = \mathsf{inputs} + 1$ to $|f|$:
  $(A_0, B_0) := (W_{\mathsf{in}_1(i)}, W_{\mathsf{in}_2(i)})$
  $(\pi_A, \pi_B) := (\pi_{\mathsf{in}_1(i)}, \pi_{\mathsf{in}_2(i)})$
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    $W_i := A_0 \oplus B_0$
    $\pi_i := \pi_A \oplus \pi_B$
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $(C, F_i) := \mathsf{MajVote}(A_0, B_0, \Delta, \pi_A, \pi_B, i)$
    $\pi_i := \mathsf{lsb}(C)$
    $W_i := C \oplus \pi_i \Delta$
for $i \in \mathsf{outputs}, j \in \{0,1\}$:
  $d_i^j := \mathsf{H}(W_i \oplus (j \oplus \pi_i)\Delta, 2|f| + i)$
return $(F,\ e = (\Delta, W_{[1,\mathsf{inputs}]}, \pi_{[1,\mathsf{inputs}]}),\ d)$

Private algorithm $\mathsf{MajVote}(A_0, B_0, \Delta, \pi_A, \pi_B, i)$:

$M_{00} := \mathsf{H}(A_0, 2i-2) \oplus \mathsf{H}(B_0, 2i-1) \oplus A_0 \oplus 1^\kappa$
$M_{01} := \mathsf{H}(A_0, 2i-2) \oplus \mathsf{H}(B_0 \oplus \Delta, 2i-1)$
$M_{10} := \mathsf{H}(A_0 \oplus \Delta, 2i-2) \oplus \mathsf{H}(B_0, 2i-1) \oplus A_0 \oplus \Delta$
$M_{11} := \mathsf{H}(A_0 \oplus \Delta, 2i-2) \oplus \mathsf{H}(B_0 \oplus \Delta, 2i-1)$
for $t = 1$ to κ :
  if $M_{\pi_A||\pi_B}[t] = M_{\pi_A \oplus 1||\pi_B}[t] = M_{\pi_A||\pi_B \oplus 1}[t]$:
    $(C[t], G[t]) := (M_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1||\pi_B \oplus 1)$
  else if $M_{\pi_A||\pi_B}[t] = M_{\pi_A \oplus 1||\pi_B}[t]$:
    $(C[t], G[t]) := (M_{\pi_A||\pi_B}[t],\ \pi_A||\pi_B \oplus 1)$
  else if $M_{\pi_A||\pi_B}[t] = M_{\pi_A||\pi_B \oplus 1}[t]$:
    $(C[t], G[t]) := (M_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1||\pi_B)$
  else if $M_{\pi_A||\pi_B \oplus 1}[t] = M_{\pi_A \oplus 1||\pi_B}[t]$:
    $(C[t], G[t]) := (M_{\pi_A||\pi_B}[t] \oplus 1,\ \pi_A||\pi_B)$
return (C, G)

Figure 5: The Gb algorithm of the Construction #2.

Evaluation algorithm $\mathsf{Ev}(F, X)$:

for $i = \mathsf{inputs} + 1$ to $|f|$:
  $(A, B) := (X_{\mathsf{in}_1(i)}, X_{\mathsf{in}_2(i)})$
  $(\alpha, \beta) := (\mathsf{lsb}(A), \mathsf{lsb}(B))$
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    $X_i := A \oplus B$
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $M := \mathsf{H}(A, 2i-2) \oplus \mathsf{H}(B, 2i-1) \oplus (\beta \oplus 1)A$
    if $\alpha||\beta = 00$:
      $M = M \oplus 1^\kappa$
    for $t = 1$ to κ :
      if $F_i[t] = \alpha||\beta$:
        $X_i[t] := M[t] \oplus 1$
      else
        $X_i[t] := M[t]$
for $i \in \mathsf{outputs}$:
  $Y_i := X_i$
return Y

Figure 6: The Ev algorithm of the Construction #2.

Encoding algorithm $\mathsf{En}(e = (\Delta, W, \pi), x)$:

for $i = 1$ to inputs :
  $X_i := W_i \oplus (x_i \oplus \pi_i)\Delta$
return X

Figure 7: The En algorithm of the Construction #2.

Decoding algorithm $\mathsf{De}(d, Y)$:

for $i \in \mathsf{outputs}$:
  if $\exists\ j$ such that $d_i^j = \mathsf{H}(Y_i, 2|f| + i)$:
    $y_i := j$
  else
    abort
return y

Figure 8: The De algorithm of the Construction #2.

a result, the evaluator can reconstruct the $(2\alpha+\beta+1)$-th row of the base. She then processes each bit of this row. If the pair (α, β) aligns with the corresponding ciphertext, the bit is flipped; otherwise, it remains the same.

Security Proof. First, we define the security property that the hash function H should satisfy. We adopt the tweakable circular correlation robustness (TCCR) as defined in [GKWY20]. A hash function with this property is sufficient to demonstrate the security of our scheme. For a hash function H, consider an oracle defined as $\mathcal{O}_\Delta^{tccr}(x,t,b) = \mathsf{H}(x \oplus \Delta, t) \oplus b\Delta$. Roughly speaking, if H is a TCCR, then the result of any query to this oracle $\mathcal{O}_\Delta^{tccr}$ is indistinguishable from the output of a random oracle, provided that the distinguisher never queries both $(x,t,0)$ and $(x,t,1)$ for any x,t.

However, as noted in [GKWY20], to prove the security of garbled circuits, TCCR is overkill. What we require is the "tweakable circular correlation robustness for naturally derived keys," which is a weaker variant of TCCR where the adversary does not possess full control over the queries made to the oracle $\mathcal{O}_\Delta^{tccr}$.

Definition 3. *Let $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{Z} \to \{0,1\}^\kappa$ be a hash function, $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\kappa$ be a random oracle, and let $\mathcal{R}$ be a distribution over $\{0,1\}^\kappa$. Say that a sequence of operations $\mathcal{Q} = (Q_1, \ldots, Q_q)$ is* natural *if each operation is one of the following:*

- $x_i \leftarrow \{0,1\}^\kappa$.
- $x_i := x_{i_1} \oplus x_{i_2}$, *where* $i_1 < i_2 < i$.
- $x_i := \mathsf{H}(x_{i_1}, t)$, *where* $i_1 < i$ *and* $t \in \mathbb{Z}$.
- $x_i := \mathcal{O}(x_{i_1}, t, b)$, *where* $i_1 < i$, $t \in \mathbb{Z}$, *and* $b \in \{0,1\}$.

We fix a natural sequence $\mathcal{Q}$ with q operations. Then we define two experiments:

1. *The real-world experiment $\mathsf{Real}_{\mathsf{H},\mathcal{Q},\mathcal{R}}$: sample $\Delta \leftarrow \mathcal{R}$, set oracle query $\mathcal{O}(x,t,b)$ to $\mathcal{O}_\Delta^{tccr}(x,t,b) = \mathsf{H}(x \oplus \Delta, t) \oplus b\Delta$.*
2. *The ideal-world experiment $\mathsf{Ideal}_{\mathsf{RO},\mathcal{Q}}$: set oracle $\mathcal{O}$ to RO.*

Each experiment defines a distribution over values $x_1, \ldots, x_q$, based on the sequential execution of operations in $\mathcal{Q}$, which are then produced as the experiment's output.

We say a H is TCCR for naturally derived keys if, for any PPT distinguisher D that never repeats an oracle query to $\mathcal{O}_\Delta^{tccr}$ on the same (x,t),

$$\left| \Pr_{\{x_i\} \leftarrow \mathsf{Real}_{\mathsf{H},\mathcal{Q},\mathcal{R}}}[D(\{x_i\}) = 1] - \Pr_{\{x_i\} \leftarrow \mathsf{Ideal}_{\mathsf{RO},\mathcal{Q}}}[D(\{x_i\}) = 1] \right| \textit{ is negligible.}$$

Now, we formally prove the security of our scheme.

Theorem 8. *Our construction described in Figures 5, 6, 7, and 8 is a secure garbling scheme with any H satisfies Definition 3.*

Proof. We will prove the four properties that a secure garbling scheme needs to satisfy.

Correctness. This property is obvious, and we provide a brief description here. We require that a base has three rows that can produce the same output label, while another row produces a different output label. We use the majority voting method to operate column by column. One column has four bits. If any of the three bits corresponding to the same output label are not equal, we flip that bit, thus ensuring their consistency. For the evaluator, she can reconstruct a row of the base, determine bit by bit whether to flip, and obtain the corresponding output label.

Privacy. We need to prove that, for any f and any x, there must exist a simulator Sim that takes the input $(1^\kappa, f, f(x))$ and outputs a (F, X, d) that is indistinguishable from the one generated in the usual way. The simulator Sim we constructed is shown in Figure 9. We prove this by hybrid arguments.

Hybrid1: In this hybrid, we will track all "active" wire labels. In the definition of the Gb algorithm, for each wire, we assume that the color bit of label W_i is 0, and use this label to track each wire. In Hybrid1, we

rewrite the Gb algorithm, using the "active" label to track each wire. The so-called "active" refers to the labels activated when evaluating the circuit, assuming the input x is known. Note that only one of the two labels for each wire will be activated.

In Figure 10, we provide a detailed description of this hybrid. The main differences between Hybrid1 and the real garbling are as follows:

$\underline{\mathsf{Sim}_{priv}(1^\kappa, f, y = f(x))}$:
$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$(F, X) \leftarrow \mathsf{Sim}_{obliv}(1^\kappa, f)$
$Y := \mathsf{Ev}(F, X)$
for $i \in \mathsf{outputs}$:
  $d_i^{y_i} := \mathsf{H}(Y_i, 2|f| + i)$
  $d_i^{y_i \oplus 1} \leftarrow \{0,1\}^\kappa$
return $(F,\ X,\ d)$

$\underline{\mathsf{Sim}_{obliv}(1^\kappa, f)}$:
$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
for $i = 1$ to inputs :
  $E_i \leftarrow \{0,1\}^\kappa$
for $i = \mathsf{inputs} + 1$ to $|f|$:
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    continue
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $F_i \leftarrow \{0,1\}^{2\kappa}$
return $(F, X = E)$

Figure 9: The algorithms of simulator Sim.

- For the i-th input wire, E_i is randomly selected as its active label, and E_i corresponds to the truth value x_i. The permute bit of this wire is $\pi_i = x_i \oplus \mathsf{lsb}(E_i)$.
- For majority voting, we use an equivalent representation. Assume that the two active input labels of the i-th AND gate are A and B, and their color bits are $\alpha = \mathsf{lsb}(A)$ and $\beta = \mathsf{lsb}(B)$. For the algorithm MajVote′, its inputs A and B are no longer the labels with color bits of 0. We represent the base using the following compact matrix representation:

$$\begin{bmatrix} M_{00} \\ M_{01} \\ M_{10} \\ M_{11} \end{bmatrix} := \begin{bmatrix} \alpha \oplus 1 & \beta \oplus 1 & \alpha & \beta \\ \alpha \oplus 1 & \beta & \alpha & \beta \oplus 1 \\ \alpha & \beta \oplus 1 & \alpha \oplus 1 & \beta \\ \alpha & \beta & \alpha \oplus 1 & \beta \oplus 1 \end{bmatrix} \times \begin{bmatrix} \mathsf{H}(A, 2i-2) \\ \mathsf{H}(B, 2i-1) \\ \mathsf{H}(A \oplus \Delta, 2i-2) \\ \mathsf{H}(B \oplus \Delta, 2i-1) \end{bmatrix} + \begin{bmatrix} \alpha \\ 0 \\ \alpha \oplus 1 \\ 0 \end{bmatrix} \times \Delta + \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} A \\ 1^\kappa \end{bmatrix}$$

We focus on three rows of this base: $M_{\pi_A||\pi_B}$, $M_{\pi_A \oplus 1||\pi_B}$, and $M_{\pi_A||\pi_B \oplus 1}$. Therefore, we represent these three rows separately as follows:

$$\begin{bmatrix} M_{\pi_A||\pi_B} \\ M_{\pi_A \oplus 1||\pi_B} \\ M_{\pi_A||\pi_B \oplus 1} \end{bmatrix} = \begin{bmatrix} (\pi_A \oplus 1) \cdot (\pi_B \oplus 1) & (\pi_A \oplus 1) \cdot \pi_B & \pi_A \cdot (\pi_B \oplus 1) & \pi_A \cdot \pi_B \\ \pi_A \cdot (\pi_B \oplus 1) & \pi_A \cdot \pi_B & (\pi_A \oplus 1) \cdot (\pi_B \oplus 1) & (\pi_A \oplus 1) \cdot \pi_B \\ (\pi_A \oplus 1) \cdot \pi_B & (\pi_A \oplus 1) \cdot (\pi_B \oplus 1) & \pi_A \cdot \pi_B & \pi_A \cdot (\pi_B \oplus 1) \end{bmatrix} \times \begin{bmatrix} M_{00} \\ M_{01} \\ M_{10} \\ M_{11} \end{bmatrix}$$

Next, during the majority voting process, we compare each bit of these three rows. Equivalently, we can use the bitwise XOR between different rows to represent the relationship between their corresponding bits. For example, if the t-th bit of $M_{\pi_A||\pi_B} \oplus M_{\pi_A \oplus 1||\pi_B}$ is 0, then it means $M_{\pi_A||\pi_B}[t] = M_{\pi_A \oplus 1||\pi_B}[t]$. After substituting the first equation into the second one, and taking the XOR of the first and second rows, as well as the XOR of the first and third rows from the second equation, we obtain the following equation:

$$\begin{bmatrix} M_{\pi_A||\pi_B} \oplus M_{\pi_A \oplus 1||\pi_B} \\ M_{\pi_A||\pi_B} \oplus M_{\pi_A||\pi_B \oplus 1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \mathsf{H}(A, 2i-2) \\ \mathsf{H}(B, 2i-1) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \mathsf{H}(A \oplus \Delta, 2i-2) \\ \mathsf{H}(B \oplus \Delta, 2i-1) \end{bmatrix} + \begin{bmatrix} \pi_B \oplus 1 \\ \pi_A \oplus \alpha \end{bmatrix} \times \Delta + \begin{bmatrix} 0 & \pi_B \oplus 1 \\ 1 & \pi_A \oplus 1 \end{bmatrix} \times \begin{bmatrix} A \\ 1^\kappa \end{bmatrix}$$

In MajVote′, we use T_0 to represent $M_{\pi_A||\pi_B} \oplus M_{\pi_A \oplus 1||\pi_B}$ and T_1 to represent $M_{\pi_A||\pi_B} \oplus M_{\pi_A||\pi_B \oplus 1}$. If $T_0[t] = T_1[t] = 0$, then it indicates that the t-th bit of the aforementioned three rows are equal. As a result, we should flip the t-th bit of the remaining row. The logic follows similarly for other scenarios.
- In the real garbling, the ciphertext and the output labels of each gate are generated together. In Hybrid1, we split this process. That is to say, the MajVote′ algorithm only needs to return the ciphertext. Then, using the input labels A and B, along with the ciphertext, the gate is evaluated to get the active output label. The correctness of the scheme ensures that this change is valid.

This hybrid is just some equivalent modifications of the real garbling, so the distributions of their outputs are identical.

$\underline{\mathsf{Hybrid1}(1^\kappa, f, x)}$:
$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$\Delta \leftarrow \{0,1\}^{\kappa-1}$
$\Delta = \Delta || 1$
for $i = 1$ to inputs :
  $E_i \leftarrow \{0,1\}^\kappa$
for $i = \mathsf{inputs} + 1$ to $|f|$:
  $(A, B) := (E_{\mathsf{in}_1(i)}, E_{\mathsf{in}_2(i)})$
  $(\alpha, \beta) := (\mathsf{lsb}(A), \mathsf{lsb}(B))$
  $(x_A, x_B) := (x_{\mathsf{in}_1(i)}, x_{\mathsf{in}_2(i)})$
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    $E_i := A \oplus B$
    $x_i := x_A \oplus x_B$
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $F_i := \mathsf{MajVote}'(A, B, \Delta, \alpha \oplus x_A, \beta \oplus x_B, i)$
    $M := \mathsf{H}(A, 2i-2) \oplus \mathsf{H}(B, 2i-1) \oplus (\beta \oplus 1)A$
    if $\alpha || \beta = 00$:
      $M = M \oplus 1^\kappa$
    for $t = 1$ to κ :
      if $F_i[t] = \alpha || \beta$:
        $E_i[t] := M[t] \oplus 1$
      else
        $E_i[t] := M[t]$
    $x_i := x_A \wedge x_B$
for $i \in \mathsf{outputs}, j \in \{0,1\}$:
  $d_i^{x_i} := \mathsf{H}(E_i, 2|f| + i)$
  $d_i^{x_i \oplus 1} := \mathsf{H}(E_i \oplus \Delta, 2|f| + i)$
return $(F,\ e = E_{[1,\mathsf{inputs}]},\ d)$

Private algorithm $\underline{\mathsf{MajVote}'(A, B, \Delta, \pi_A, \pi_B, i)}$:
$(\alpha, \beta) := (\mathsf{lsb}(A), \mathsf{lsb}(B))$
$T_0 := \mathsf{H}(A, 2i-2) \oplus \mathsf{H}(A \oplus \Delta, 2i-2) \oplus (\pi_B \oplus 1)\Delta \oplus (\pi_B \oplus 1)1^\kappa$
$T_1 := \mathsf{H}(B, 2i-1) \oplus \mathsf{H}(B \oplus \Delta, 2i-1) \oplus (\pi_A \oplus \alpha)\Delta \oplus A \oplus (\pi_A \oplus 1)1^\kappa$
for $t = 1$ to κ :
  if $T_0[t] = 0$ and $T_1[t] = 0$:
    $G[t] := \pi_A \oplus 1 || \pi_B \oplus 1$
  else if $T_0[t] = 0$ and $T_1[t] = 1$:
    $G[t] := \pi_A || \pi_B \oplus 1$
  else if $T_0[t] = 1$ and $T_1[t] = 0$:
    $G[t] := \pi_A \oplus 1 || \pi_B$
  else if $T_0[t] = 1$ and $T_1[t] = 1$:
    $G[t] := \pi_A || \pi_B$
return G

Figure 10: Hybrid1 for proof of privacy.

Hybrid2: In this hybrid, we employ the property of TCCR for naturally derived keys, as defined earlier, for hash queries in the form of $\mathsf{H}(\cdot \oplus \Delta, \cdot)$. In Hybrid1, the hash queries in this format appears in the following three places:

- $T_0 := \mathsf{H}(A, 2i-2) \oplus \mathsf{H}(A \oplus \Delta, 2i-2) \oplus (\pi_B \oplus 1)\Delta \oplus (\pi_B \oplus 1)1^\kappa$
- $T_1 := \mathsf{H}(B, 2i-1) \oplus \mathsf{H}(B \oplus \Delta, 2i-1) \oplus (\pi_A \oplus \alpha)\Delta \oplus A \oplus (\pi_A \oplus 1)1^\kappa$
- $d_i^{x_i \oplus 1} := \mathsf{H}(E_i \oplus \Delta, 2|f| + i)$

We represent it in the form of $\mathcal{O}_\Delta^{tccr}(x, t, b) = \mathsf{H}(x \oplus \Delta, t) \oplus b\Delta$:

- $T_0 := \mathsf{H}(A, 2i-2) \oplus \mathcal{O}_\Delta^{tccr}(A, 2i-2, \pi_B \oplus 1) \oplus (\pi_B \oplus 1)1^\kappa$
- $T_1 := \mathsf{H}(B, 2i-1) \oplus \mathcal{O}_\Delta^{tccr}(B, 2i-1, \pi_A \oplus \alpha) \oplus A \oplus (\pi_A \oplus 1)1^\kappa$
- $d_i^{x_i \oplus 1} := \mathcal{O}_\Delta^{tccr}(E_i, 2|f| + i, 0)$

As can be seen, we have moved all references to Δ into the oracle $\mathcal{O}_\Delta^{tccr}$. By using its security property, replacing the results of these oracle queries with random values will only result in a negligible effect.

Hybrid3: In this hybrid, we let the ciphertext of each AND gate be randomly generated; that is, $F_i \leftarrow \{0,1\}^{2\kappa}$. Recall that in Hybrid2, T_0 and T_1 become two independent random values. As a result, for each position, the pair $(T_0[t], T_1[t])$ uniformly falls into the set $\{(0,0),(0,1),(1,0),(1,1)\}$. Consequently, the position of the bit that needs to be flipped is uniformly random. Therefore, Hybrid3 and Hybrid2 have the same distribution.

Since the simulator Sim knows the circuit's output $f(x)$, he can construct the decoding information as shown in Figure 9. It can be seen that the distribution of Sim_{priv}'s output is identical to the output of Hybrid3.

Obliviousness. The Sim_{priv} we defined calls the Sim_{obliv} as a subroutine. The (F, X) generated by the Sim_{obliv} is indistinguishable from the (F, X) generated by the real garbling, which can be directly derived from the proof of privacy.

Authenticity. We need to proof that for any PPT adversary, given the input (F, X, d), it should be impossible to produce a $\tilde{Y}$ such that $\mathsf{De}(d, \tilde{Y}) \notin \{y = f(x), \perp\}$, except with negligible probability. To compromise this property, the adversary needs to identify at least one output wire such that $d_i^{y_i \oplus 1} = \mathsf{H}(\tilde{Y}_i, 2|f| + i)$. However, in the proof of privacy, we demonstrated that $d_i^{y_i \oplus 1}$ can be substituted with a random value. Therefore, the probability of such an event occurring is $2^{-\kappa}$, which is negligible. □

C Supplemental Materials for Section 5

C.1 Formal Definition of MODEL-3

The formal definition of MODEL-3 is as follows:

- Garbling algorithm Gb:
  Parameterized by integers m, r, q, n, d, ℓ, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{M}_{ij} \mid i, j \in \{0,1\}\}$, probability p, functions Extend and Contract, random oracle RO and mapping function MAP. The length of $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$ is $r + d$, with entries in $\mathrm{GF}(2^\kappa)$. The length of each $\boldsymbol{M}_{ij}$ is $n + q$, with entries in $\mathrm{GF}(2^\ell)$. The Extend is defined as $\mathrm{GF}(2^\kappa)^r \to \mathrm{GF}(2^\ell)^n$. The Contract is defined as $\mathrm{GF}(2^\ell)^q \to \mathrm{GF}(2^\kappa)^d$. The RO is defined as $\{0,1\}^* \to \mathrm{GF}(2^\ell)$. The function MAP is defined as $\mathrm{GF}(2^4) \times \mathrm{GF}(2^m) \to \mathrm{GF}(2^4)$, and MAP can be decomposed into four sub-functions ($\mathsf{MAP}_{00}, \mathsf{MAP}_{01}, \mathsf{MAP}_{10}, \mathsf{MAP}_{11}$).

  1. For $i \in [r]$, choose $R_i \leftarrow \mathrm{GF}(2^\kappa)$, and compute $(N_1, \ldots, N_n) := \mathsf{Extend}(R_1, \ldots, R_r)$.
  2. Make q distinct queries to the RO, which can be determined based on the R_i values. Let $Q_1, \ldots, Q_q$ denote the responses to these queries, and compute $(D_1, \ldots, D_d) := \mathsf{Contract}(Q_1, \ldots, Q_q)$. Define $\boldsymbol{S}_1 := (R_1, \ldots, R_r, D_1, \ldots, D_d)$ and $\boldsymbol{S}_2 := (N_1, \ldots, N_n, Q_1, \ldots, Q_q)$.
  3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.

4. Compute $A_0 := \langle \boldsymbol{A}_0, \boldsymbol{S}_1 \rangle$, $A_1 := \langle \boldsymbol{A}_1, \boldsymbol{S}_1 \rangle$, $B_0 := \langle \boldsymbol{B}_0, \boldsymbol{S}_1 \rangle$, $B_1 := \langle \boldsymbol{B}_1, \boldsymbol{S}_1 \rangle$. Then $A_0||0$ and $A_1||1$ are labels for one input wire, and $B_0||0$ and $B_1||1$ for the other. Subscripts indicate the color bits, with A_a and B_b representing FALSE.
5. For $i, j \in \{0,1\}$, compute $M_{ij} := \langle \boldsymbol{M}_{ij}, \boldsymbol{S}_2 \rangle$. Then $(M_{00}, M_{01}, M_{10}, M_{11})^\top$ is the base. The t-th column of the base, represented by $\boldsymbol{M}[t]$, is composed of 4 bits and can be interpreted as $(M_{00}[t], M_{01}[t], M_{10}[t], M_{11}[t])$.
6. For $t \in [\ell]$, the processing of $\boldsymbol{M}[t]$ falls into one of the following two cases:
   (a) *With a probability of* p, find m-bit ciphertext $\boldsymbol{G}[t]$ such that the following two conditions are satisfied:
      i. $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}_{00}(M_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{01}(M_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{10}(M_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}_{11}(M_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z_{00}[t] \\ Z_{01}[t] \\ Z_{10}[t] \\ Z_{11}[t] \end{pmatrix}$;
      ii. $Z_{ab}[t] = Z_{(a\oplus 1)b}[t] = Z_{a(b\oplus 1)}[t]$.

      Then use $Z_{ab}[t]$ as one bit of the output label C_0 and $Z_{(a\oplus 1)(b\oplus 1)}[t]$ as one bit of the output label C_1, where C_0 corresponds to FALSE.
   (b) *With a probability of* $1-p$, the desired mapping does not exist. Using a particular $\boldsymbol{G}[t]$ to indicate that skip this column.

   If the lengths of C_0 and C_1 reach κ, then the iteration ends. The sub-ciphertext $\boldsymbol{G}[t]$, with a length of m, represents the t-th part of the whole ciphertext $\boldsymbol{G}$.

- Encoding algorithm En:
  Given inputs $x_a, x_b \in \{0,1\}$, 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 Ev:
  Parameterized by integers q, n, ℓ, vectors $\{\boldsymbol{V}_{\alpha\beta} \mid \alpha, \beta \in \{0,1\}\}$, function Extend', random oracle RO and mapping function MAP. The length of each $\boldsymbol{V}_{\alpha\beta}$ is $n+q$, with entries in $\mathrm{GF}(2^\ell)$. The Extend' is defined as $\mathrm{GF}(2^\kappa)^2 \to \mathrm{GF}(2^\ell)^n$.
  1. The inputs are wire labels $A_\alpha||\alpha$, $B_\beta||\beta$ and the ciphertext $\boldsymbol{G}$.
  2. Make q distinct queries to the RO, which can be determined based on the input wire labels. Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, compute $(N_1, \ldots, N_n) := \mathsf{Extend}'(A_\alpha, B_\beta)$ and define $\boldsymbol{T} := (N_1, \ldots, N_n, Q'_1, \ldots, Q'_q)$.
  3. Compute $V_{\alpha\beta} := \langle \boldsymbol{V}_{\alpha\beta}, \boldsymbol{T} \rangle$, and use $V_{\alpha\beta}[t]$ to denote the t-th bit of $V_{\alpha\beta}$.
  4. For $t \in [\ell]$, if $\boldsymbol{G}[t]$ indicates that this column is valid, compute $Z_{\alpha\beta}[t] := \mathsf{MAP}_{\alpha\beta}(V_{\alpha\beta}[t], \boldsymbol{G}[t])$ and use $Z_{\alpha\beta}[t]$ as one bit of the output label. Continue to handle the next column until the length of the output label reaches κ.

Remark 7. All columns will be successfully processed in MODEL-2, while in MODEL-3, there's a success probability of p for each column. To achieve a κ-bit output label, the expected value for ℓ becomes κ/p. In MODEL-3, we also introduce two auxiliary functions: Extend and $\mathsf{Contract}$. These functions mainly tackle the issue stemming from the different lengths between wire labels and random oracle outputs. For instance, each row of the base — which results from the XOR of random oracle outputs and input labels — has a length of ℓ. As a result, the Extend function is essential to transform the input labels into elements in $\mathrm{GF}(2^\ell)$.

C.2 Proof of Theorem 2

Theorem 2. *Any ideally secure garbling scheme for an* AND *gate conforming to* MODEL-3 *must satisfy* $|\boldsymbol{G}| \geq 2\kappa$ *on average.*

Proof. In MODEL-3, each column's processing produces a bit of the output label with a probability p. To

prove that $|\boldsymbol{G}| = \sum_{t=1}^{\ell} |\boldsymbol{G}[t]| = \ell \cdot m \geq 2\kappa$, we need only to demonstrate the two following assertions:

1. If $p = 1$, then $m \geq 2$.
2. If $m = 1$, then $p \leq 1/2$ (indeed, we obtain that $p \leq 3/8$).

The first assertion arises directly from Theorem 1, so our attention is on the second. When $m = 1$, $\boldsymbol{G}[t]$ contains only a single bit, implying two potential values and making everything quite restrictive. From the first assertion, it's evident that if $m = 1$, then $p < 1$. Therefore, we should use one potential value of $\boldsymbol{G}[t]$ to indicate that the operation on the t-th column can yield one bit of the output label, and use another value to indicate that this column is invalid and should be simply skipped. Consequently, for each $\boldsymbol{M}[t]$, by varying $\boldsymbol{G}[t]$, the mapping $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ can generate at most one valid vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$.

Each column of the base $\boldsymbol{M}[t]$ in MODEL-3 has 16 potential values, with each value appearing with equal probability. To establish this, it suffices to demonstrate that there is no fixed relationship between different rows of the base. Otherwise, the privacy property of the garbling scheme would be compromised. Using proof by contradiction, we assume that there exists a certain relationship among the four rows of the base, and without loss of generality, the fourth row of the base is involved in this relationship. As the generation of the base is independent of the choice of permute bits, we may assume that the first three rows lead to producing C_0 and the fourth row to C_1. If the evaluator obtains input labels A_i and B_j, where $(i, j) \neq (1, 1)$, the valid bits of the first three rows of the base become reconstructable[13]. Under the assumption, the evaluator can then infer some information about the fourth row. This allows the evaluator to discern certain bits of the output label C_1, thereby breaching privacy. For clarity, we provide a concrete counterexample. Imagine the mapping function MAP is the same as in half gates, and the four rows of the base are as follows: $\mathsf{H}(A_0) \oplus \mathsf{H}(B_0)$, $\mathsf{H}(A_0) \oplus \mathsf{H}(B_1)$, $\mathsf{H}(A_1) \oplus \mathsf{H}(B_0)$, and $\mathsf{H}(A_1) \oplus \mathsf{H}(B_1)$. In this setup, the XOR operation on the first three rows produces the fourth row. Consequently, only 8 potential values arise for $\boldsymbol{M}[t]$. However, this construction compromises privacy. If we assume the first three rows correspond to C_0 and the last row to C_1, the evaluator, upon obtaining A_0 and B_0, can reconstruct the first three rows of the base. This would immediately reveal the fourth row and the output label C_1.

We use $\mathcal{S} = \mathcal{S}_{00} \cup \mathcal{S}_{01} \cup \mathcal{S}_{10} \cup \mathcal{S}_{11}$ to denote the set including all the possible values in Table 3. The mapping function $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ produces a valid vector $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ in $\mathcal{S}$. A subset of $\mathcal{S}$ is $\mathcal{S}_1 = \{(0, 0, 0, 0),\ (1, 1, 1, 1)\}$, and another subset is $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$.

Consider the 16 potential values of $\boldsymbol{M}[t]$. Let x denote the number of values yielding a valid vector in subset $\mathcal{S}_1$ and let y denote the number of values yielding a valid vector in subset $\mathcal{S}_2$. The residual $16 - x - y$ values don't give a valid vector and should be skipped. In other words, there are x potential values of $\boldsymbol{M}[t]$ mapping to every row (as illustrated in Table 3), while y values map to a specific row. To maintain the privacy property of the garbling scheme, the probability of $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ emerging from $\mathcal{S}_1$ should be equal to its probability from $\mathcal{S}_2$ [14]. This condition gives $y = 4x$.

Finally, we calculate the probability. We can compute the overall probability p as follows:

- For $\boldsymbol{M}[t]$ taking one of the x values, the probability of yielding a bit of the output label is 1.
- For $\boldsymbol{M}[t]$ taking one of the y values, the probability of yielding a bit of the output label is $1/4$. For example, the vector $(0, 0, 0, 1)$ is valid only in the case where permute bits $(a, b) = (0, 0)$.
- For $\boldsymbol{M}[t]$ taking one of the $16 - x - y$ values, the probability of yielding a bit of the output label is 0, since it's guaranteed to be invalid.

Therefore, considering each value of $\boldsymbol{M}[t]$ appears with equal probability, the overall probability p that a column will yield a bit of the output label is given by:

$$p = \frac{1}{16} \times 1 \times x + \frac{1}{16} \times \frac{1}{4} \times y + \frac{1}{16} \times 0 \times (16 - x - y) = \frac{x}{8}.$$

[13] Specifically, after the evaluator obtains the output label C_0, for each column, if the ciphertext $\boldsymbol{G}[t]$ indicates that this column is valid, then the evaluator can compute $M_{00}[t]$, $M_{01}[t]$, $M_{10}[t]$. Note that, under the fixed $\boldsymbol{G}[t]$ condition, MAP_{ij} is injective (where $i, j \in \{0, 1\}$), ensuring the deterministic values of $M_{00}[t]$, $M_{01}[t]$, $M_{10}[t]$. Taking MAP_{00} as an example, if it is not injective, meaning that regardless of whether $M_{00}[t]$ is 0 or 1, the mapping function MAP_{00} maps it to the same value, then 1 bit of security will be lost.

[14] Appearing in the first set indicates that the t-th bit of C_0 and C_1 are the same, while the second set indicates they differ. Both should have equal probabilities, as we've discussed in the proof of Theorem 1.

Given that $16 - x - y = 16 - 5x \geq 0$ and x is an integer, we have

$$p = \frac{x}{8} \leq \frac{3}{8} < \frac{1}{2}.$$

Therefore, the second assertion holds true.
If $p = 1$ then $m \geq 2$. We have $|\boldsymbol{G}| \geq 2\kappa$.
If $m = 1$ then $p < 1/2$. We have the expectation length of $\boldsymbol{G}$ with $|\boldsymbol{G}| = \kappa/p > 2\kappa$. □

D Supplemental Materials for Section 6

D.1 Processing Multi-columns Simultaneously

In the proof of Theorem 1 for MODEL-2, we note that once $\boldsymbol{M}[t]$ is set, the set of possible values for $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ by varying $\boldsymbol{G}[t]$, must intersect with each set in Table 3. If $\boldsymbol{G}[t]$ is only one bit, each $\boldsymbol{M}[t]$ can map to at most two distinct values. This leads to a higher occurrence of $(0,0,0,0)$ and $(1,1,1,1)$ in the output $\boldsymbol{Z}[t]$, compromising privacy. Therefore, we opt for $\boldsymbol{G}[t]$ to have two bits.

In fact, ensuring each $\boldsymbol{M}[t]$ maps to no more than three unique values suffices for privacy. For example, with a given $\boldsymbol{M}[t]$, the value set of $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ by changing $\boldsymbol{G}[t]$ could be $\{(0,0,0,0), (0,1,0,0), (1,0,0,0)\}$. If $(a,b) \in \{(0,0),(0,1)\}$, it maps to $(0,0,0,0)$. If $(a,b) = (1,0)$, it maps to $(0,1,0,0)$. If $(a,b) = (1,1)$, it maps to $(1,0,0,0)$. The probability that the bits in the same position of the output labels C_0 and C_1 match is $1/2$. In this regard, there is no issue with privacy. Therefore, when amortizing over each column, a more favorable lower bound for the length of the sub-ciphertext is $\log 3$, and it remains possible that a secure garbled circuit scheme that processes multiple columns simultaneously might achieve a ciphertext length close to $\log 3 \cdot \kappa$.

D.2 Formal Definition of MODEL-3′

The formal definition of MODEL-3′ is as follows:

- Garbling algorithm Gb:
  Parameterized by integers m, r, q, w, n, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{M}_{ij}^{ab,u} \mid i,j,a,b \in \{0,1\}, u \in [w]\}$, function $\mathsf{Contract}$, random oracle RO and mapping function $\{\mathsf{MAP}^u \mid u \in [w]\}$. The lengths of $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$ are all $r+q$, with entries in $\mathrm{GF}(2^\kappa)$. The length of each $\boldsymbol{M}_{ij}^{ab,u}$ is n, with entries in $\mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})$. The $\mathsf{Contract}$ is defined as $\mathrm{GF}(2^\kappa)^{r+q} \to \mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})^n$. The RO is defined as $\{0,1\}^* \to \mathrm{GF}(2^\kappa)$. Each function MAP^u is defined as $\mathrm{GF}(2^4) \times \mathrm{GF}(2^m) \to \mathrm{GF}(2^4)$.
  Also parameterized[15] by integers v, s, g, vectors $\{\boldsymbol{Y}^{ab} \mid a,b \in \{0,1\}\}$, matrix $\boldsymbol{\mathcal{X}}$, function $\mathsf{Truncate}$. The length of each vector is s, with entries in $\mathrm{GF}(2^v)$. The size of the matrix is $g \times (s+q)$, with entries in $\mathrm{GF}(2^v)$. The $\mathsf{Truncate}$ is defined as $\mathrm{GF}(2^\kappa)^q \to \mathrm{GF}(2^v)^q$.

  1. For $i \in [r]$, choose $R_i \leftarrow \mathrm{GF}(2^\kappa)$.
  2. Make q distinct queries to the RO, which can be determined based on the R_i values. Let $Q_1, \ldots, Q_q$ denote the responses to these queries, define $\boldsymbol{S}_1 := (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$, $\boldsymbol{S}_2 := \mathsf{Contract}(R_1, \ldots, R_r, Q_1, \ldots, Q_q)$, $\boldsymbol{S}_3 := \mathsf{Truncate}(Q_1, \ldots, Q_q)$.
  3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.
  4. Compute $A_0 := \langle \boldsymbol{A}_0, \boldsymbol{S}_1 \rangle$, $A_1 := \langle \boldsymbol{A}_1, \boldsymbol{S}_1 \rangle$, $B_0 := \langle \boldsymbol{B}_0, \boldsymbol{S}_1 \rangle$, $B_1 := \langle \boldsymbol{B}_1, \boldsymbol{S}_1 \rangle$. Then $A_0||0$ and $A_1||1$ are labels for one input wire, and $B_0||0$ and $B_1||1$ for the other. Subscripts indicate the color bits, with A_a and B_b representing FALSE.
  5. For $u \in [w]$ and $i,j \in \{0,1\}$, compute $M_{ij}^u := \langle \boldsymbol{M}_{ij}^{ab,u}, \boldsymbol{S}_2 \rangle$. Then $(M_{00}^u, M_{01}^u, M_{10}^u, M_{11}^u)^\top$ is the u-th base. The t-th column of the u-th base, represented by $\boldsymbol{M}^u[t]$, is composed of 4 bits and can be interpreted as $(M_{00}^u[t], M_{01}^u[t], M_{10}^u[t], M_{11}^u[t])$.

[15] These parameters are used for generating additional ciphertext, corresponding to the "dicing" technique in [RR21].

6. (*Generation of the main ciphertext*) For $t \in [\lceil \kappa/w \rceil]$, find m-bit ciphertext $\boldsymbol{G}[t]$ such that for any $u \in [w]$, the following two conditions are satisfied:

   (a) $\mathsf{MAP}^u(\boldsymbol{M}^u[t], \boldsymbol{G}[t]) = \begin{pmatrix} \mathsf{MAP}^u_{00}(M^u_{00}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{01}(M^u_{01}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{10}(M^u_{10}[t], \boldsymbol{G}[t]) \\ \mathsf{MAP}^u_{11}(M^u_{11}[t], \boldsymbol{G}[t]) \end{pmatrix} = \begin{pmatrix} Z^u_{00}[t] \\ Z^u_{01}[t] \\ Z^u_{10}[t] \\ Z^u_{11}[t] \end{pmatrix}$;

   (b) $Z^u_{ab}[t] = Z^u_{(a\oplus 1)b}[t] = Z^u_{a(b\oplus 1)}[t]$.

   Use $Z^1_{ab}[t], \ldots, Z^w_{ab}[t]$ as w bits of C_0, and $Z^1_{(a\oplus 1)(b\oplus 1)}[t], \ldots, Z^w_{(a\oplus 1)(b\oplus 1)}[t]$ as w bits of the output label C_1, where C_0 corresponds to FALSE. If the output label reaches κ bits, discard any excess bits.

7. (*Generation of the additional ciphertext*) Compute $\boldsymbol{E} := \boldsymbol{\mathcal{X}} \times (\boldsymbol{Y}^{ab} || \boldsymbol{S}_3)^\top$ as additional ciphertext, where $||$ denotes the concatenation of two vectors.

   Then $\boldsymbol{G}$ and $\boldsymbol{E}$ constitutes the whole ciphertext, with a total length of $\lceil \kappa/w \rceil \cdot m + g \cdot v$.

- Encoding algorithm En:
  Given inputs $x_a, x_b \in \{0,1\}$, 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 Ev:
  Parameterized by integers q, w, x, v, g, vectors $\{\boldsymbol{V}^u_{\alpha\beta} \mid \alpha, \beta \in \{0,1\}, u \in [w]\}$, $\{\boldsymbol{K}_{\alpha\beta} \mid \alpha, \beta \in \{0,1\}\}$, function $\mathsf{Contract}'$ and $\mathsf{Truncate}$, random oracle RO and mapping function MAP. The length of each $\boldsymbol{V}^u_{\alpha\beta}$ is x, with entries in $\mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})$. The length of each $\boldsymbol{K}_{\alpha\beta}$ is $g+q$, with entries in $\mathrm{GF}(2^v)$. The $\mathsf{Contract}'$ is defined as $\mathrm{GF}(2^\kappa)^{2+q} \to \mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})^x$.

  1. The inputs are wire labels $A_\alpha || \alpha$, $B_\beta || \beta$ and the garbled circuit ciphertext $\boldsymbol{G}$, $\boldsymbol{E}$.
  2. Make q distinct queries to the RO, which can be determined based on the input wire labels. Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries. Compute $\boldsymbol{T} := \mathsf{Contract}'(A_\alpha, B_\beta, Q'_1, \ldots, Q'_q)$, and $\boldsymbol{U} := \mathsf{Truncate}(Q'_1, \ldots, Q'_q)$.
  3. Compute $K_{\alpha\beta} := \langle \boldsymbol{K}_{\alpha\beta}, (\boldsymbol{E} || \boldsymbol{U}) \rangle$ and parse it into w vectors, each with a length of x and entries in $\mathrm{GF}(2^{\lceil \frac{\kappa}{w} \rceil})$, denote these vectors as $\{\hat{\boldsymbol{V}}^u_{\alpha\beta} \mid u \in [w]\}$.
  4. For $u \in [w]$, compute $V^u_{\alpha\beta} := \langle \boldsymbol{V}^u_{\alpha\beta} + \hat{\boldsymbol{V}}^u_{\alpha\beta}, \boldsymbol{T} \rangle$, and use $V^u_{\alpha\beta}[t]$ to denote the t-th bit of $V^u_{\alpha\beta}$.
  5. For $t \in [\lceil \frac{\kappa}{w} \rceil]$, use $Z^1_{\alpha\beta}[t] := \mathsf{MAP}^1_{\alpha\beta}(V^1_{\alpha\beta}[t], \boldsymbol{G}[t]), \ldots, Z^w_{\alpha\beta}[t] := \mathsf{MAP}^w_{\alpha\beta}(V^w_{\alpha\beta}[t], \boldsymbol{G}[t])$ as w bits of the output label. If the output label reaches κ bits, discard any excess bits.

Remark 8. Compared to MODEL-2, the definition of MODEL-3′ is notably more intricate. We delve deeper into its clarification. The integer w denotes the number of bases. In MODEL-2, $w = 1$ while in [RR21] $w = 2$. We have not set any specific constraints on the mapping function MAP^u, while in [RR21] it is a linear operation. As indicated by [RR21], the generation of each base is influenced by permute bits. Therefore, in the definition of MODEL-3′, the parameter $\boldsymbol{M}^{ab,u}_{ij}$ that used to generate the u-th base is associated with the permute bits (a, b). To enable the evaluator to reconstruct the $(2\alpha+\beta+1)$-th row of each base, an additional constant-size ciphertext $\boldsymbol{E}$ needs to be transmitted.

D.3 Proof of Lemma 1

Lemma 1. *In a garbling scheme with w bases adhering to* MODEL-3′*, suppose two bases $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$ (where $1 \le i < j \le w$) are **column-correlated**. Once $\boldsymbol{M}^i$ is determined, the number of potential values for each column of $\boldsymbol{M}^j$ is halved, reducing from 16 to 8.*

Proof. Given that $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$ satisfy **column-correlated**, vectors $\boldsymbol{P}^i$ and $\boldsymbol{P}^j$ of length four exist such that $\langle \boldsymbol{P}^i, \boldsymbol{M}^i \rangle = \langle \boldsymbol{P}^j, \boldsymbol{M}^j \rangle$. Since we've already pinned down what $\boldsymbol{M}^i$ is, we can compute $K = \langle \boldsymbol{P}^i, \boldsymbol{M}^i \rangle$. We represent $\boldsymbol{P}^j$ and $\boldsymbol{M}^j$ as consisting of four elements, as follows:

$$\boldsymbol{M}^j := (M^j_{00}, M^j_{01}, M^j_{10}, M^j_{11}) \quad \boldsymbol{P}^j := (P^j_{00}, P^j_{01}, P^j_{10}, P^j_{11}).$$

Using the above representation, we obtain: $P_{00}^j \cdot M_{00}^j + P_{01}^j \cdot M_{01}^j + P_{10}^j \cdot M_{10}^j + P_{11}^j \cdot M_{11}^j = K$. Without loss of generality, we can freely choose the values of M_{00}^j, M_{01}^j, and M_{10}^j. Once these values are fixed, M_{11}^j is determined by the above equation. Therefore, M_{11}^j cannot be freely chosen. As a result, every column of $\boldsymbol{M}^j$ has only $2^3 = 8$ potential values. □

D.4 Proof of Theorem 3

Theorem 3. *Consider an ideally secure garbling scheme in* MODEL-3′ *with* w bases. *Assume that each* base *has at most one* ***ancestor****. If* υ *pairs of* bases *are* ***column-correlated****, then for garbling an* AND *gate, the ciphertext length* $|\boldsymbol{G}|$ *must satisfy* $|\boldsymbol{G}| \geq (2 - \upsilon/w) \cdot \kappa$, *where* $\upsilon \leq w - 1$.

Proof. Given that w bases exist, of which υ pairs satisfy **column-correlated**, we can infer that $w - \upsilon$ bases lack an **ancestor**, while the remaining υ bases possess an **ancestor**. With this in mind, we present the following two assertions:

1. For a base without an **ancestor**, processing each column necessitates a ciphertext of at least 2 bits.
2. Conversely, for a base that does have an **ancestor**, processing each column demands a ciphertext of at least 1 bit.

If we prove the above two assertions, then for each processing, at least $2(w - \upsilon)$-bit ciphertext is needed for $w - \upsilon$ bases, and at least υ-bit ciphertext is needed for υ bases. Given that one processing can yield w bits of the output label, at least κ/w processing steps are required. Therefore, the total ciphertext length $|G| \geq (2(w - \upsilon) + \upsilon) \cdot (\kappa/w) = (2 - \upsilon/w) \cdot \kappa$.

We denote the base u as $\boldsymbol{M}^u = (M_{00}^u, M_{01}^u, M_{10}^u, M_{11}^u)$, where $u \in [w]$, and we refer to the t-th column of $\boldsymbol{M}^u$ as $\boldsymbol{M}^u[t]$.

We begin by proving the first assertion. Assuming the base u has no **ancestor**, it can be considered independent, and therefore has 16 possible values (recall that at the beginning of Section 6.3.1, we assumed that every column of each base would have 16 potential values). Regardless of the permute bits randomly selected by the garbler, $\boldsymbol{M}^u[t]$ can assume any of the 16 potential values. Therefore, for a fixed $\boldsymbol{M}^u[t]$, we need to vary the value of the ciphertext $\boldsymbol{G}^u[t]$ [16] to map $\boldsymbol{M}^u[t]$ to each of the 4 cases outlined in Table 3. Given that the mapping function MAP^u is independent of the permute bits, the length of $\boldsymbol{G}^u[t]$ must be at least 2 bits, as already demonstrated in the proof of Theorem 1. In addition, it is worth noting that in the proof of Theorem 1, we do not need to prove that $\boldsymbol{M}^u[t]$ can take 16 values (even though it is true). In MODEL-2, the value of $\boldsymbol{M}^u[t]$ is independent of the permute bits, so each potential value needs to be mapped to 4 cases. However, in MODEL-3′, it is explicitly required that $\boldsymbol{M}^u[t]$ can assume 16 potential values in every cases because its value may depend on the permute bits.

Next, we proceed to prove the second assertion. This time, we assume that the base u has an ancestor $\boldsymbol{M}^v$, where $v < u$. Once the value of $\boldsymbol{M}^v$ is determined, Lemma 1 shows that every column of $\boldsymbol{M}^u$ can take on only 8 potential values. Without loss of generality, we assume that the permute bits $(a, b) = (0, 0)$. We need to map these 8 potential values to elements in the corresponding set, as follows:

$$\left\{ \begin{array}{l} (a_0, b_0, c_0, d_0),\ (a_1, b_1, c_1, d_1) \\ (a_2, b_2, c_2, d_2),\ (a_3, b_3, c_3, d_3) \\ (a_4, b_4, c_4, d_4),\ (a_5, b_5, c_5, d_5) \\ (a_6, b_6, c_6, d_6),\ (a_7, b_7, c_7, d_7) \end{array} \right\} \xrightarrow{\mathsf{MAP}^u \text{ with } \boldsymbol{G}^u[t]} \left\{ \begin{array}{l} (0,0,0,0) \\ (0,0,0,1) \\ (1,1,1,1) \\ (1,1,1,0) \end{array} \right\}.$$

In the left side, (a_i, b_i, c_i, d_i) represent the 8 distinct values of $\boldsymbol{M}^u[t]$, where $0 \leq i \leq 7$. We employ a proof by contradiction, assuming that the length of $\boldsymbol{G}^u[t]$ is 0. As defined in MODEL-3′, we decompose the function MAP^u into four sub-functions $(\mathsf{MAP}_{00}^u, \mathsf{MAP}_{01}^u, \mathsf{MAP}_{10}^u, \mathsf{MAP}_{11}^u)$. For $\mathsf{MAP}_{00}^u(a_i, \boldsymbol{G}[t])$, since the length of $\boldsymbol{G}[t]$ is 0, $\mathsf{MAP}_{00}^u(0, \boldsymbol{G}[t])$ can yield at most one value, and similarly, $\mathsf{MAP}_{00}^u(1, \boldsymbol{G}[t])$ can also yield at most one value. The same conclusion applies to the other sub-functions as well.

First, let us consider all a_i values. For $a_i \in \{0, 1\}, 0 \leq i \leq 7$, there exists a value $x \in \{0, 1\}$ such that the number of a_i where $a_i = x$ is at least 4. We reorder the 8 distinct values of $\boldsymbol{M}^u[t]$ such that for $0 \leq i \leq r$, $a_i = x$, and for $r + 1 \leq i \leq 7$, $a_i = x \oplus 1$, with $r \geq 3$. If we map all a_i to the same value regardless of

[16] Here, we decompose the ciphertext $\boldsymbol{G}[t]$ into w parts, each designated for handling a column extracted from w bases.

whether $a_i = 0$ or $a_i = 1$, then we would incur a loss of 1-bit in security. Therefore, for $0 \le i \le r$, we map a_i to bit y, and for $r+1 \le i \le 7$, we map a_i to $y \oplus 1$.

Next, we consider all b_i values. Given our assumption that permute bits $(a, b) = (0, 0)$, for $0 \le i \le 7$, a_i and b_i should be mapped to the same value. If for $0 \le i < j \le r$, there exists $b_i \ne b_j$, then we have $\mathsf{MAP}^u_{01}(0, \boldsymbol{G}[t]) = \mathsf{MAP}^u_{01}(1, \boldsymbol{G}[t]) = y$. However, for $r+1 \le i \le 7$, b_i needs to be mapped to $y \oplus 1$, that is, $\mathsf{MAP}^u_{01}(b_i, \boldsymbol{G}[t]) = y \oplus 1$. Whether $b_i = 0$ or $b_i = 1$, this contradicts the previous equation. Therefore, we can conclude that for $0 \le i \le r$, all b_i are the same bit , and for $r+1 \le i \le 7$, all b_i are also the same bit.

For all values of c_i, the same conclusion holds: for $0 \le i \le r$, all c_i are the same bit, and for $r+1 \le i \le 7$, all c_i are also the same bit.

Thus far, we have derived a contradiction. For $0 \le i \le r$, the first three bits of the vector (a_i, b_i, c_i, d_i) are identical. By varying d_i, we can obtain at most two distinct vectors, contradicting the condition that $r \ge 3$. □

D.5 Proof of Theorem 4

Before formally presenting the proof, we first review the design paradigm of [RR21], which involves transforming the design of a garbling scheme into solving a system of linear equations and requiring compatibility with free-XOR. For example, if we have three bases, the garbling scheme can be expressed in the following form (derived from Equation (4) in [RR21]):

$$\overbrace{\begin{matrix}\text{the first base}\\ \\ \\ \\ \text{the second base}\\ \\ \\ \\ \text{the third base}\\ \\ \\ \end{matrix}\!\!\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\\1&0&0\\ \hline 0&1&0\\0&1&0\\0&1&0\\0&1&0\\ \hline 0&0&1\\0&0&1\\0&0&1\\0&0&1\end{bmatrix}}^{\boldsymbol{\mathcal{W}}} \overbrace{\begin{bmatrix}C^{(1)}\\C^{(2)}\\C^{(3)}\end{bmatrix}}^{\mathbf{C}} \oplus \boldsymbol{\mathcal{V}} \overbrace{\begin{bmatrix}G_0\\G_1\\G_2\\G_3\end{bmatrix}}^{\mathbf{G}} = \boldsymbol{\mathcal{M}}\mathbf{H} \oplus \boldsymbol{\mathcal{R}} \overbrace{\begin{bmatrix}A_0^{(1)}\\A_0^{(2)}\\A_0^{(3)}\\B_0^{(1)}\\B_0^{(2)}\\B_0^{(3)}\\\Delta^{(1)}\\\Delta^{(2)}\\\Delta^{(3)}\end{bmatrix}}^{\mathbf{Q}} \oplus \overbrace{\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\\1&0&0\\ \hline 0&0&0\\0&0&0\\0&0&0\\0&1&0\\ \hline 0&0&0\\0&0&0\\0&0&0\\0&0&1\end{bmatrix}}^{\boldsymbol{\mathcal{T}}} \overbrace{\begin{bmatrix}\Delta^{(1)}\\\Delta^{(2)}\\\Delta^{(3)}\end{bmatrix}}^{\mathbf{D}}, \tag{2}$$

abbreviated as $\boldsymbol{\mathcal{W}}\mathbf{C} \oplus \boldsymbol{\mathcal{V}}\mathbf{G} = \boldsymbol{\mathcal{M}}\mathbf{H} \oplus \boldsymbol{\mathcal{R}}\mathbf{Q} \oplus \boldsymbol{\mathcal{T}}\mathbf{D}$.

Note that we rearrange the original equation so that the first four rows of the equation correspond to the processing of the first base, the middle four rows to the second base, and the last four rows to the third base. The matrix $\boldsymbol{\mathcal{V}} \in \mathrm{GF}(2)^{12\times 4}$ represents the mapping function MAP, which is a linear function in this context. The vector $\mathbf{H} \in \mathrm{GF}(2^{\kappa/3})^{\ell}$ represents potential hash queries, for example, it might be $(\mathsf{H}(A_0), \mathsf{H}(A_1), \mathsf{H}(B_0), \mathsf{H}(B_1), \cdots)^\top$. The matrix $\boldsymbol{\mathcal{M}} \in \mathrm{GF}(2)^{12\times \ell}$ represents the hash composition of each row in each base. The matrix $\boldsymbol{\mathcal{R}} \in \mathrm{GF}(2)^{12\times 9}$ represents the input label composition of each row in each base. The red positions in the matrix $\boldsymbol{\mathcal{T}} \in \mathrm{GF}(2)^{12\times 3}$ are related to the choice of permute bits (a, b). Specifically, in the red positions, the $(4-2a-b)$, $(8-2a-b)$, and $(12-2a-b)$-th rows are set to 1, while the rest are set to 0.

To better understand the relationship between our base terminology and the linear equation system representation, we consider a concrete example. Assume the three bases are as follows:

the first base	the second base	the third base
$\mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \oplus S^1_{00}$	$\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^2_{00}$	$\mathsf{H}(B_0) \oplus \mathsf{H}'(A_0 \oplus B_0) \oplus S^3_{00}$
$\mathsf{H}(A_0) \oplus \mathsf{H}(B_1) \oplus S^1_{01}$	$\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^2_{01}$	$\mathsf{H}(B_1) \oplus \mathsf{H}'(A_0 \oplus B_1) \oplus S^3_{01}$
$\mathsf{H}(A_1) \oplus \mathsf{H}(B_0) \oplus S^1_{10}$	$\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^2_{10}$	$\mathsf{H}(B_0) \oplus \mathsf{H}'(A_1 \oplus B_0) \oplus S^3_{10}$
$\mathsf{H}(A_1) \oplus \mathsf{H}(B_1) \oplus S^1_{11}$	$\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_1) \oplus S^2_{11}$	$\mathsf{H}(B_1) \oplus \mathsf{H}'(A_1 \oplus B_1) \oplus S^3_{11}$

where S^u_{ij} $(i, j \in \{0, 1\}, u \in \{1, 2, 3\})$ represents a certain combination of input labels, and H, H' are two different hash functions. Recall that we are now in the free-XOR setting, therefore we have $A_0 \oplus A_1 =$

$B_0 \oplus B_1 = \Delta$. If the mapping function MAP is linear, then the processing of the above three bases can be represented using a linear equation system, as shown below:

$$\boldsymbol{\mathcal{W}} \begin{bmatrix} C^{(1)} \\ C^{(2)} \\ C^{(3)} \end{bmatrix} \oplus \boldsymbol{\mathcal{V}} \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = \left[\begin{array}{cccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} \mathsf{H}(A_0) \\ \mathsf{H}(A_1) \\ \mathsf{H}(B_0) \\ \mathsf{H}(B_1) \\ \mathsf{H}(A_0 \oplus B_0) \\ \mathsf{H}(A_0 \oplus B_1) \\ \mathsf{H}'(A_0 \oplus B_0) \\ \mathsf{H}'(A_0 \oplus B_1) \end{bmatrix} \oplus \boldsymbol{\mathcal{R}}\mathbf{Q} \oplus \boldsymbol{\mathcal{T}}\mathbf{D},$$

where $\boldsymbol{\mathcal{R}}\mathbf{Q} = (S^1_{00}, S^1_{01}, S^1_{10}, S^1_{11}, S^2_{00}, S^2_{01}, S^2_{10}, S^2_{11}, S^3_{00}, S^3_{01}, S^3_{10}, S^3_{11})^\top$.

In the above example, the vector composed of S^u_{ij} is another equivalent representation of $\boldsymbol{\mathcal{R}}\mathbf{Q}$, and this representation is general, not limited to this example. For the convenience of subsequent proofs, we expand the representation of each S^u_{ij} as follows:

$$S^u_{00} = x^u_1 A^{(1)}_0 \oplus x^u_2 A^{(2)}_0 \oplus x^u_3 A^{(3)}_0 \oplus x^u_4 B^{(1)}_0 \oplus x^u_5 B^{(2)}_0 \oplus x^u_6 B^{(3)}_0 \quad (3)$$

$$S^u_{01} = y^u_1 A^{(1)}_0 \oplus y^u_2 A^{(2)}_0 \oplus y^u_3 A^{(3)}_0 \oplus y^u_4 (B^{(1)}_0 \oplus \Delta^{(1)}) \oplus y^u_5 (B^{(2)}_0 \oplus \Delta^{(2)}) \oplus y^u_6 (B^{(3)}_0 \oplus \Delta^{(3)}) \quad (4)$$

$$S^u_{10} = z^u_1 (A^{(1)}_0 \oplus \Delta^{(1)}) \oplus z^u_2 (A^{(2)}_0 \oplus \Delta^{(2)}) \oplus z^u_3 (A^{(3)}_0 \oplus \Delta^{(3)}) \oplus z^u_4 B^{(1)}_0 \oplus z^u_5 B^{(2)}_0 \oplus z^u_6 B^{(3)}_0 \quad (5)$$

$$S^u_{11} = w^u_1 (A^{(1)}_0 \oplus \Delta^{(1)}) \oplus w^u_2 (A^{(2)}_0 \oplus \Delta^{(2)}) \oplus w^u_3 (A^{(3)}_0 \oplus \Delta^{(3)}) \oplus w^u_4 (B^{(1)}_0 \oplus \Delta^{(1)}) \oplus w^u_5 (B^{(2)}_0 \oplus \Delta^{(2)}) \oplus w^u_6 (B^{(3)}_0 \oplus \Delta^{(3)}), \quad (6)$$

and coefficients $x^u_i, y^u_i, z^u_i, w^u_i \in \{0,1\}$, for $i \in [6], u \in [3]$.

Next, we represent the main theorem that we intend to prove.

Theorem 4. *If a secure garbling scheme can be represented by a linear equation system, and the correlation between bases can be expressed using Definition 2, then this scheme requires at least $3/2 \cdot \kappa + O(1)$ bits of ciphertext length for garbling an* AND *gate.*

Proof. Recalling the case with three bases, we attempt to achieve a garbling scheme that requires only $4\kappa/3$ bits of ciphertext by introducing more correlations. Firstly, we will demonstrate that such a secure garbling scheme does not exist. Finally, we will extend this to the general case, allowing any number of bases. We will break down the proof into three steps.

Step1 - Determine the vector H and matrices $\boldsymbol{\mathcal{M}}$, $\boldsymbol{\mathcal{V}}$.

We allow three types of hash queries: the first type contains only input label A_α, the second type contains only input label B_β, and the third type contains only $A_\alpha \oplus B_\beta$. For example, the hash composition of the first row of the first base might be $\mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus \cdots$. However, for the same input label, different hash functions can be used.

It should be noted that allowing "unlinear" hash queries like $\mathsf{H}(A_\alpha, B_\beta)$ does not contribute to smaller garbled circuit ciphertext. The reason is that this would result in the four rows of a base being independently random, preventing us from utilizing the correlation between rows to reduce the length of the ciphertext while achieving compatibility with free-XOR (this point can be observed from our transition from Construction #1 to Construction #2).

Based on the above observations, we can obtain the form of vector $\mathbf{H}$ is like

$$(\mathsf{H}(A_0), \mathsf{H}(A_1), \mathsf{H}'(A_0), \mathsf{H}'(A_1), \cdots, \mathsf{H}(B_0), \mathsf{H}(B_1), \cdots, \mathsf{H}(A_0 \oplus B_0), \mathsf{H}(A_0 \oplus B_0 \oplus \Delta), \cdots)^\top.$$

Next, we need to determine matrix $\boldsymbol{\mathcal{M}}$, composed of four submatrices. Submatrix $\boldsymbol{\mathcal{M}}_1$ is for hash queries with input A_0 or A_1, submatrix $\boldsymbol{\mathcal{M}}_2$ for B_0 or B_1, submatrix $\boldsymbol{\mathcal{M}}_3$ for $A_0 \oplus B_0$ or $A_0 \oplus B_0 \oplus \Delta$, and

the fourth submatrix is a zero matrix, as follows:

$$\boldsymbol{\mathcal{M}}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{\mathcal{M}}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{\mathcal{M}}_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \boldsymbol{\mathcal{M}}_4 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

That is to say, matrix $\boldsymbol{\mathcal{M}}$ can be divided into blocks with size 4×2. For instance, for the $\boldsymbol{\mathcal{M}}$ in the previous example, we can write it as

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \boldsymbol{\mathcal{M}}_1 & \boldsymbol{\mathcal{M}}_2 & \boldsymbol{\mathcal{M}}_4 & \boldsymbol{\mathcal{M}}_4 \\ \boldsymbol{\mathcal{M}}_1 & \boldsymbol{\mathcal{M}}_4 & \boldsymbol{\mathcal{M}}_3 & \boldsymbol{\mathcal{M}}_4 \\ \boldsymbol{\mathcal{M}}_4 & \boldsymbol{\mathcal{M}}_2 & \boldsymbol{\mathcal{M}}_4 & \boldsymbol{\mathcal{M}}_3 \end{bmatrix}.$$

To ensure that the Equation 2 is solvable, the column space of matrix $\boldsymbol{\mathcal{M}}$ should be the same as the column space of matrix $[\boldsymbol{\mathcal{W}}||\boldsymbol{\mathcal{V}}]$. Therefore, we can simply select four linearly independent columns from matrix $\boldsymbol{\mathcal{M}}$ (and these four columns should also be linearly independent with matrix $\boldsymbol{\mathcal{W}}$) to form the matrix $\boldsymbol{\mathcal{V}}$. Without loss of generality, we can assume that the first row of matrix $\boldsymbol{\mathcal{V}}$ is all zeros (if a certain position in the first row of matrix $\boldsymbol{\mathcal{V}}$ is not zero, then we can XOR the column corresponding to that position with the first column of matrix $\boldsymbol{\mathcal{W}}$ to obtain a new column to replace the original one.). Similarly, we can assume that the 5-th and 9-th rows of matrix $\boldsymbol{\mathcal{V}}$ are also zeros. As a result, we can directly form the output label C_0 or C_1 by combining the first rows of the three bases, without the need to XOR with any ciphertext.

Step2 - Obtain the correlation within and between bases.

First, we obtain the internal correlation within each base, namely the correlation between the four rows of each base. In Step1, it can be seen that the XOR of the four rows of submatrix $\boldsymbol{\mathcal{M}}_i$ (where $i \in \{1,2,3,4\}$) equals $\mathbf{0}$. Therefore, the XOR of the first four rows of matrix $\boldsymbol{\mathcal{M}}$, the XOR of the middle four rows, and the XOR of the last four rows all equal $\mathbf{0}$. Obviously, matrices $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{V}}$ (where $\boldsymbol{\mathcal{V}}$ is derived from $\boldsymbol{\mathcal{M}}$) also satisfy this relationship. Consequently, by XORing the first four rows, the middle four rows, and the last four rows of the Equation 2, we can establish the following three equations:

$$S^1_{00} \oplus S^1_{01} \oplus S^1_{10} \oplus S^1_{11} = \Delta^{(1)} \tag{7}$$
$$S^2_{00} \oplus S^2_{01} \oplus S^2_{10} \oplus S^2_{11} = \Delta^{(2)} \tag{8}$$
$$S^3_{00} \oplus S^3_{01} \oplus S^3_{10} \oplus S^3_{11} = \Delta^{(3)}. \tag{9}$$

Next, we attempt to find correlations between different bases.

Lemma 3. *For the submatrix $\boldsymbol{\mathcal{V}}_0$, composed of the first four rows of matrix $\boldsymbol{\mathcal{V}}$, its rank must be 2.*

Proof. Since matrix $\boldsymbol{\mathcal{V}}$ is obtained from $\boldsymbol{\mathcal{M}}$, and assuming its first row consists entirely of zero elements, we can deduce that the elements of matrix $\boldsymbol{\mathcal{V}}_0$ are restricted such that each column vector of $\boldsymbol{\mathcal{V}}_0$ is an element from the set $\{(0,0,0,0)^\top, (0,0,1,1)^\top, (0,1,0,1)^\top, (0,1,1,0)^\top\}$. If we can prove that at least two of the column vectors of $\boldsymbol{\mathcal{V}}_0$ are non-zero and distinct, then the rank of $\boldsymbol{\mathcal{V}}_0$ is 2, as there are two linearly independent column vectors.

Assuming that $\boldsymbol{\mathcal{V}}_0$ has only one column of non-zero element, or multiple columns of identical element, then there must exist a row i which is entirely composed of zero elements, where $i \in \{2,3,4\}$. This means that if the color bits (α, β) obtained by the evaluator satisfy $2\alpha + \beta \in \{0, i-1\}$, then there is no need to XOR any ciphertext, and the corresponding row of the base is just the output label.

Without loss of generality, assume $i = 2$. If the permute bits (a, b) belongs to $\{(0,0),(0,1)\}$, then the first and second rows of the first base are equal. Due to the randomness of the random oracle outputs, it is necessary to ensure that the hash queries in the first and second rows are equal. Therefore, we can deduce that $S^1_{00} = S^1_{01}$. Similarly, if (a, b) is in $\{(1,0),(1,1)\}$, then the first two rows of the base correspond to two different output labels, which have an offset of $\Delta^{(1)}$, so we have $S^1_{00} = S^1_{01} \oplus \Delta^{(1)}$. Based on Equations 3 and 4, we can derive the following relationship:

$$x^1_j = y^1_j, \text{for } j \in [6].$$

In S^1_{00} and S^1_{01}, the only coefficient related to $\Delta^{(1)}$ is y^1_4. If $(a,b) \in \{(0,0),(0,1)\}$, then it must be that $x^1_4 = y^1_4 = 0$. Conversely, if $(a,b) \in \{(1,0),(1,1)\}$, then it must be that $x^1_4 = y^1_4 = 1$. If the evaluator receives color bits $(\alpha,\beta) = (0,0)$, then she can determine the value of S^1_{00}, namely she can learn x^1_4. Based on whether x^1_4 is 0 or 1, the evaluator can infer whether the permute bits belong to $\{(0,0),(0,1)\}$ or $\{(1,0),(1,1)\}$, thereby violating privacy. □

From Lemma 3, we can adjust matrix $\boldsymbol{\mathcal{V}}$ so that its first four rows are equal to:

$$\boldsymbol{\mathcal{V}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

From now on, for simplicity, we will temporarily ignore the part related to permute bits in the Equation 2, that is, temporarily disregard $\boldsymbol{\mathcal{T}}\mathbf{D}$. In such case, for each row of a base, after XORing with the corresponding ciphertext, the same output label will be obtained. We represent the four rows of the first base using $\boldsymbol{M}^1 = (M^1_{00}, M^1_{01}, M^1_{10}, M^1_{11})^\top$, and therefore we have

$$M^1_{00} = M^1_{01} \oplus G_0 = M^1_{10} \oplus G_1 = M^1_{11} \oplus G_0 \oplus G_1, \text{ namely, } \begin{cases} G_0 = M^1_{00} \oplus M^1_{01} \\ G_1 = M^1_{00} \oplus M^1_{10} \end{cases}.$$

Based on the determined first base, for the second base, $\boldsymbol{M}^2 = (M^2_{00}, M^2_{01}, M^2_{10}, M^2_{11})^\top$, we conduct a similar analysis. Using X_0 and X_1 to represent some linear combinations of G_0, G_1, G_2, G_3, we can obtain

$$M^2_{00} = M^2_{01} \oplus X_0 = M^2_{10} \oplus X_1 = M^2_{11} \oplus X_0 \oplus X_1, \text{ namely, } \begin{cases} X_0 = M^2_{00} \oplus M^2_{01} \\ X_1 = M^2_{00} \oplus M^2_{10} \end{cases}.$$

Recall that the three bases cannot be uncorrelated, as if there were no correlation among them, each column of a base would require at least $2-$bit sub-ciphertext, making it impossible to complete the mapping of three columns with just 4 bits. Therefore, without loss of generality, we can assume that there is a correlation between the first base and the second base because otherwise, we could simply exchange the order of three bases[17]. Therefore, the processing of the first base and the second base must share some ciphertext, namely $\{X_0, X_1, X_0 \oplus X_1\} \cap \{G_0, G_1, G_0 \oplus G_1\} \neq \emptyset$. Corresponding to Definition 2, we can deduce the existence of non-zero vectors $\boldsymbol{P}^1$ and $\boldsymbol{P}^2$, such that $\langle \boldsymbol{P}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle$, where $\boldsymbol{P}^1, \boldsymbol{P}^2 \in \mathcal{U} = \{(1,1,0,0)^\top, (1,0,1,0)^\top, (1,0,0,1)^\top\}$.

Finally, we need to establish the correlation between the third base, $\boldsymbol{M}^3 = (M^3_{00}, M^3_{01}, M^3_{10}, M^3_{11})^\top$, and the previous two bases. There are three possible cases:

1. There exist non-zero vectors $\boldsymbol{Q}^1$ and $\boldsymbol{Q}^3$, such that $\langle \boldsymbol{Q}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$, where $\boldsymbol{Q}^1, \boldsymbol{Q}^3 \in \mathcal{U}$.
2. There exist non-zero vectors $\boldsymbol{Q}^2$ and $\boldsymbol{Q}^3$, such that $\langle \boldsymbol{Q}^2, \boldsymbol{M}^2 \rangle = \langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$, where $\boldsymbol{Q}^2, \boldsymbol{Q}^3 \in \mathcal{U}$.
3. There exist non-zero vectors $\boldsymbol{Q}^1$, $\boldsymbol{Q}^2$, and $\boldsymbol{Q}^3$, such that $\langle \boldsymbol{Q}^1, \boldsymbol{M}^1 \rangle + \langle \boldsymbol{Q}^2, \boldsymbol{M}^2 \rangle = \langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$, where $\boldsymbol{Q}^1, \boldsymbol{Q}^2, \boldsymbol{Q}^3 \in \mathcal{U}$.

The first case and the second case are equivalent because for the second case, we only need to swap the order of the first base and second base to transform it into the first case. However, for the third case, although it goes beyond our Definition 2, we can use the same proof technique as for the first case to conclude that it is impossible to occur. Therefore, we only need to focus on the first case.

Up to this point, we have obtained the correlation between the first and second bases (indicated by vectors $\boldsymbol{P}^1, \boldsymbol{P}^2$) and the correlation between the first and third bases (indicated by vectors $\boldsymbol{Q}^1, \boldsymbol{Q}^3$).

Ultimately, we will demonstrate that if the three bases possess the above correlations, it would violate privacy.

Step3 - Show such a scheme would leak privacy.

[17] We should note that, when $X_0 = G_0 \oplus G_2$ and $X_1 = G_0 \oplus G_3$, we cannot consider that the first base and the second base share ciphertext. This is because, with the first appearances of G_2 and G_3, X_0 and X_1 should be considered as two entirely new ciphertext. In other words, by adjusting matrix $\boldsymbol{\mathcal{V}}$, we can use $G'_2 = G_0 \oplus G_2$ and $G'_3 = G_0 \oplus G_3$ to replace G_2 and G_3.

Recall that we allow three types of hash queries: the first type is only related to the input label A_α (TYPE-1), the second type is related to B_β (TYPE-2), and the third type is only related to $A_\alpha \oplus B_\beta$ (TYPE-3). From Lemma 3, it is known that the hash queries composing the first base must include at least two types, otherwise the rank of the submatrix $\boldsymbol{\mathcal{V}}_0$ cannot be 2. Similarly, this holds true for the second and third bases as well.

Assuming that the hash-query-type of a base are TYPE-1 + TYPE-2, then by XORing the first row with the second row, the hash queries related to the input label A_α will be eliminated (because both the first and second rows correspond to A_0), leaving only TYPE-2-hash-query, i.e., the hash queries related to the input label B_β. In Table 5, we have summarized all the possible scenarios that might occur.

Table 5: Hash-query-type for a base and its row-wise XOR results. The table columns represent the hash combinations that compose a base, with a total of four possibilities. The table rows represent the XOR results of two rows of a base. For example, $(1,1,0,0)^\top$ corresponds to the XOR of the first and second rows of the base, which is analogous to the vector inner product in Definition 2.

	TYPE-1 + TYPE-2	TYPE-1 + TYPE-3	TYPE-2 + TYPE-3	TYPE-1 + TYPE-2 + TYPE-3
$(1,1,0,0)^\top$	TYPE-2	TYPE-3	TYPE-2 + TYPE-3	TYPE-2 + TYPE-3
$(1,0,1,0)^\top$	TYPE-1	TYPE-1 + TYPE-3	TYPE-3	TYPE-1 + TYPE-3
$(1,0,0,1)^\top$	TYPE-1 + TYPE-2	TYPE-1	TYPE-2	TYPE-1 + TYPE-2

Before we continue, we first prove a lemma.

Lemma 4. *For two distinct bases $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$, where $i,j \in \{1,2,3\}$ and $i \neq j$, if there exist vectors $\boldsymbol{P}^i$ and $\boldsymbol{P}^j$ such that $\langle \boldsymbol{P}^i, \boldsymbol{M}^i \rangle = \langle \boldsymbol{P}^j, \boldsymbol{M}^j \rangle$, where $\boldsymbol{P}^i, \boldsymbol{P}^j \in \mathcal{U}$, then if $\boldsymbol{P}^i = \boldsymbol{P}^j$, it would violate the privacy of the garbling scheme.*

Proof. We only consider $\boldsymbol{P}^i = \boldsymbol{P}^j = (1,0,0,1)^\top$ because the proof method is the same for other cases.

From the condition $\langle \boldsymbol{P}^i, \boldsymbol{M}^i \rangle = \langle \boldsymbol{P}^j, \boldsymbol{M}^j \rangle$, we have $M^i_{00} \oplus M^i_{11} = M^j_{00} \oplus M^j_{11}$. Taking M^i_{00} as an example, it includes a combination of hash queries and input labels, the latter of which we denote as S^i_{00}. Due to the randomness of the output from the random oracle, the parts related to hash queries in $M^i_{00} \oplus M^i_{11} = M^j_{00} \oplus M^j_{11}$ must cancel each other out. Therefore, we have $S^i_{00} \oplus S^i_{11} = S^j_{00} \oplus S^j_{11}$. In S^i_{00}, S^i_{11}, S^j_{00}, and S^j_{11}, the coefficients related to the input labels $A^{(\mathtt{i})}_0$ and $B^{(\mathtt{i})}_0$ must also satisfy this relation. From Equations 3 and 6, we have

$$x^i_i \oplus w^i_i = x^j_i \oplus w^j_i, \text{ and } x^i_{i+3} \oplus w^i_{i+3} = x^j_{i+3} \oplus w^j_{i+3}. \tag{10}$$

Recall that we previously ignored the part of the Equation 2 related to permute bits, namely $\boldsymbol{\mathcal{T}}\mathbf{D}$ (we assumed that the four rows of a base, after being processed by the ciphertext, would map to the same output label). However, in fact, based on different choices of permute bits, only three rows are mapped to the same output label, while the other row is mapped to a different label with a Δ offset. The premise for obtaining the relation $S^i_{00} \oplus S^i_{11} = S^j_{00} \oplus S^j_{11}$ is that the first and fourth rows of each base are mapped to the same output label, which corresponds to the permute bits $(a,b) \in \{(0,1),(1,0)\}$. Nevertheless, if the permute bits $(a,b) \in \{(0,0),(1,1)\}$, then the first and fourth rows of each base will be mapped to two different output labels. In this case, the relation we obtain is $S^i_{00} \oplus S^i_{11} \oplus \Delta^{(\mathtt{i})} = S^j_{00} \oplus S^j_{11} \oplus \Delta^{(\mathtt{j})}$. In S^i_{00}, S^i_{11}, S^j_{00}, and S^j_{11}, the coefficients related to $\Delta^{(\mathtt{i})}$ are w^i_i, w^i_{i+3}, w^j_i, and w^j_{i+3}. Therefore, if the permute bits $(a,b) \in \{(0,1),(1,0)\}$, then we have $w^i_i \oplus w^i_{i+3} \oplus w^j_i \oplus w^j_{i+3} = 0$. On the other hand, if $(a,b) \in \{(0,0),(1,1)\}$, then we have $w^i_i \oplus w^i_{i+3} \oplus w^j_i \oplus w^j_{i+3} = 1$.

From Equation 10, we can derive $x^i_i \oplus x^i_{i+3} \oplus x^j_i \oplus x^j_{i+3} = w^i_i \oplus w^i_{i+3} \oplus w^j_i \oplus w^j_{i+3}$. If the evaluator receives color bits $(\alpha,\beta) = (0,0)$, then she will obtain the value of $x^i_i \oplus x^i_{i+3} \oplus x^j_i \oplus x^j_{i+3}$. Based on whether this value is 0 or 1, the evaluator can distinguish whether the permute bits belong to $\{(0,1),(1,0)\}$ or $\{(0,0),(1,1)\}$. □

It is known that the first base and the second base satisfy $\langle \boldsymbol{P}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle$. According to Lemma 4, we have $\boldsymbol{P}^1 \neq \boldsymbol{P}^2$. Since $\boldsymbol{P}^1, \boldsymbol{P}^2 \in \mathcal{U} = \{(1,1,0,0)^\top, (1,0,1,0)^\top, (1,0,0,1)^\top\}$, there are a total of 6 potential values for $(\boldsymbol{P}^1, \boldsymbol{P}^2)$. Due to the randomness of permute bits, where any row of the base corresponds to

outputting TRUE with a probability of $1/4$, the 6 potential cases have symmetry. Therefore, we only need to analyze one of them.

Lemma 5. *If $\boldsymbol{P}^1 = (1,1,0,0)^\top$ and $\boldsymbol{P}^2 = (1,0,1,0)^\top$, then the garbling scheme does not possess the property of privacy.*

Proof. As indicated in Table 5, the intersection of the first and second rows of the table is only TYPE-3. This means that if $\langle \boldsymbol{P}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle$ holds true, then the hash-query-type for the first base must be TYPE-1 + TYPE-3, and for the second base, it must be TYPE-2 + TYPE-3. The further explanation is that if the hash-query-type for the first base is TYPE-1 + TYPE-3, then $\langle \boldsymbol{P}^1, \boldsymbol{M}^1 \rangle$ is left only with TYPE-3-hash-query. Therefore, $\langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle$ must also be a TYPE-3-hash-query, and this is only possible when the hash-query-type for the second base is TYPE-2 + TYPE-3. If we disregard $\mathcal{T}\mathbf{D}$, then we have the relationship

$$S^1_{00} \oplus S^1_{01} = S^2_{00} \oplus S^2_{10}. \tag{11}$$

Next, we will shift our focus to the correlation between the first base and third base, namely $\langle \boldsymbol{Q}^1, \boldsymbol{M}^1 \rangle = \langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$. Similarly, according to Lemma 4, we have $\boldsymbol{Q}^1 \neq \boldsymbol{Q}^3$. Note that we have determined the hash-query-type of the first base to be TYPE-1 + TYPE-3. We discuss the following three scenarios based on the different values of $\boldsymbol{Q}^1$:

1. $\boldsymbol{Q}^1 = (1,1,0,0)^\top$: In this case, $\boldsymbol{P}^1 = \boldsymbol{Q}^1$, therefore we can obtain that $\langle \boldsymbol{P}^2, \boldsymbol{M}^2 \rangle = \langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$. Since $\langle \boldsymbol{Q}^1, \boldsymbol{M}^1 \rangle$ is a TYPE-3-hash-query, $\langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$ is also a TYPE-3-hash-query. According to Table 5, the hash-query-type of the third base is TYPE-2 + TYPE-3, and $\boldsymbol{Q}^3 = (1,0,1,0)^\top$. Therefore, we arrive at $\boldsymbol{P}^2 = \boldsymbol{Q}^3$, but according to Lemma 4, this is not feasible.
2. $\boldsymbol{Q}^1 = (1,0,1,0)^\top$: In this case, $\langle \boldsymbol{Q}^1, \boldsymbol{M}^1 \rangle$ is a (TYPE-1 + TYPE-3)-hash-query, but we cannot find a $\boldsymbol{Q}^3 \neq \boldsymbol{Q}^1$ such that $\langle \boldsymbol{Q}^3, \boldsymbol{M}^3 \rangle$ is also of the same type.
3. $\boldsymbol{Q}^1 = (1,0,0,1)^\top$: Similarly, according to Table 5, we can obtain that the hash-query-type of the third base is TYPE-1 + TYPE-2, and $\boldsymbol{Q}^3 = (1,0,1,0)^\top$. If we do not consider $\mathcal{T}\mathbf{D}$, then we have the relationship

$$S^1_{00} \oplus S^1_{11} = S^3_{00} \oplus S^3_{10}. \tag{12}$$

   This is the most complex scenario, and next, we will focus on analyzing this case in detail.

Combining Equations 7, 11, and 12, we can derive the following equation

$$S^1_{00} \oplus S^2_{00} \oplus S^3_{00} = S^1_{10} \oplus S^2_{10} \oplus S^3_{10} \oplus \Delta^{(1)}.$$

According to Equations 3 and 5, by comparing the coefficients of the input label $A^{(1)}_0$, we can obtain

$$x^1_1 \oplus x^2_1 \oplus x^3_1 = z^1_1 \oplus z^2_1 \oplus z^3_1. \tag{13}$$

Previously, we disregarded $\mathcal{T}\mathbf{D}$, meaning we ignored the impact of permute bits. Now it's time to consider it. If the permute bits $(a,b) \in \{(0,0),(1,0)\}$, then the following equation holds

$$S^1_{00} \oplus S^2_{00} \oplus S^3_{00} = S^1_{10} \oplus S^2_{10} \oplus S^3_{10}. \tag{14}$$

However, if the permute bits $(a,b) \in \{(0,1),(1,1)\}$, we need to modify it to

$$S^1_{00} \oplus S^2_{00} \oplus S^3_{00} = S^1_{10} \oplus S^2_{10} \oplus S^3_{10} \oplus \Delta^{(1)} \oplus \Delta^{(2)} \oplus \Delta^{(3)}. \tag{15}$$

Finally, we consider the coefficients related to $\Delta^{(1)}$ in Equations 14 and 15. If the permute bits $(a,b) \in \{(0,0),(1,0)\}$, then $z^1_1 \oplus z^2_1 \oplus z^3_1 = 0$. If the permute bits $(a,b) \in \{(0,1),(1,1)\}$, then $z^1_1 \oplus z^2_1 \oplus z^3_1 = 1$. If the evaluator receives color bits $(\alpha,\beta) = (0,0)$, then in conjunction with Equation 13, it becomes apparent that such a garbling scheme does not satisfy privacy. □

Interestingly, the proof approach we use for Lemmas 3, 4, and 5 is similar. Specifically, we first derive the relationship $\mathbb{E}_1$ satisfied by S^u_{ij}, where $i, j \in \{0, 1\}$ and $u \in [3]$, which is related to permute bits. Based on this $\mathbb{E}_1$, we can deduce the relationship $\mathbb{E}_2$ that the coefficients of the input labels composing S^u_{ij} need to satisfy, and this relationship is independent of permute bits. Additionally, from $\mathbb{E}_1$, we can deduce a relationship $\mathbb{E}_3$ related to Δ, which is dependent on permute bits. Finally, the evaluator have the coefficients of the corresponding input labels based on her row's $S^u_{\alpha\beta}$, and combine $\mathbb{E}_2$ and $\mathbb{E}_3$ to obtain partial information about permute bits.

Therefore, we can arrive at the following claim:

Claim 1. *If a secure garbling scheme is composed of* 3 bases *and can be represented by a linear equation system as shown in Equation 2, then it requires at least* $5/3 \cdot \kappa + O(1)$ *bits of ciphertext length for garbling an* AND *gate.*

Proof. From the previous proofs, it is known that if the length of vector $\mathbf{G}$ is 4, consisting of G_0, G_1, G_2, G_3, then such a garbling scheme will not meet privacy requirement.

It is feasible to achieve a ciphertext length of $5/3 \cdot \kappa + O(1)$ bits. This means that the first and second bases have a correlation, while the third base is independent, not correlated with the first two bases. Therefore, for one column taken from the first base and one from the second base, a 3-bit sub-ciphertext is needed. According to Theorem 1, for one column in the third base, only a 2-bit sub-ciphertext is required. Hence, each processing step needs a 5-bit sub-ciphertext. With a total of $\kappa/3$ operations, the required ciphertext length is $5/3 \cdot \kappa + O(1)$ bits. □

Furthermore, we can observe that if one base correlates with multiple subsequent bases, it results in an insecure garbling scheme. In other words, a base should at most correlate with only one subsequent base in accordance with Definition 2. Finally, combining this with Theorem 3, we can conclude the proof. □

D.6 An Attack on Sliced Garbling

In a recent work by Ashur et al. [AHS24], they showed an extension of [RR21] for garbling a special 3-input gate $x \wedge (y \oplus z)$, utilizing three bases (in our terminology). Their construction achieved a ciphertext length of only $4\kappa/3$. By specifically setting z to 0, they obtained a scheme to garble the 2-input AND gate, only requiring ciphertext length of $4\kappa/3$.

At first glance, their result might not appear contradictory to the impossibility stated in our Theorem 4, as they are dealing with a specific structure of a 3-input gate. However, setting $z = 0$ just implies introducing additional constant in the proof of our theorem, which does not affect the conclusion we have drawn. Therefore, their work is indeed in contradiction with our result. Actually, there exists a gap between their construction and their security proof. In their construction, they claimed that the linear equation system is solvable, but this does not necessarily imply that the evaluator's marginal view is independent of the (secret) permute bits. Employing the same proof method in our Theorem 4, it can be directly concluded that their garbling scheme for the 3-input gate fails to meet privacy requirement. This implies that the evaluator can infer some information about the true values of wires during the evaluation process.

Next, we provide a detailed description of the attack on their scheme. As before, the scheme can be expressed as a system of linear equations, as depicted in Equation 16. We assume that the labels for the three input wires are (A_0, A_1), (B_0, B_1), and (C_0, C_1), while the labels for the output wire are (D_0, D_1). We also employ the parameters they provided in Appendix A. (Although there are a total of 24 rows in the system, for simplicity, we include only the first 9 rows.)

$$\left[\begin{array}{ccccccc}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\1&0&0&0&0&0&1\\0&1&0&0&0&1&1\\0&0&1&0&0&1&1\\1&0&0&0&1&0&1\\0&1&0&0&1&0&1\\0&0&1&1&0&0&1\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\end{array}\right]\left[\begin{array}{c}D^{(1)}\\D^{(2)}\\D^{(3)}\\G_0\\G_1\\G_2\\G_3\end{array}\right]=\left[\begin{array}{cccccccc}1&0&1&0&0&0&1&0\\0&0&1&0&1&0&1&0\\1&0&0&0&1&0&1&0\\1&0&1&0&0&0&0&1\\0&0&1&0&0&1&0&1\\1&0&0&0&0&1&0&1\\1&0&0&1&0&0&0&1\\0&0&0&1&1&0&0&1\\1&0&0&0&1&0&0&1\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\end{array}\right]\left[\begin{array}{c}\mathsf{H}(A_0)\\\mathsf{H}(A_1)\\\mathsf{H}(B_0)\\\mathsf{H}(B_1)\\\mathsf{H}(C_0)\\\mathsf{H}(C_1)\\\mathsf{H}(A_0\oplus B_0\oplus C_0)\\\mathsf{H}(A_0\oplus B_0\oplus C_1)\end{array}\right]\oplus\boldsymbol{\mathcal{R}}\left[\begin{array}{c}A_0^{(1)}\\A_0^{(2)}\\A_0^{(3)}\\B_0^{(1)}\\B_0^{(2)}\\B_0^{(3)}\\C_0^{(1)}\\C_0^{(2)}\\C_0^{(3)}\\\Delta^{(1)}\\\Delta^{(2)}\\\Delta^{(3)}\end{array}\right]\oplus\boldsymbol{\mathcal{T}}\left[\begin{array}{c}\Delta^{(1)}\\\Delta^{(2)}\\\Delta^{(3)}\end{array}\right] \tag{16}$$

The matrix $\boldsymbol{\mathcal{T}} \in \mathrm{GF}(2)^{24\times 3}$ depends on the permute bits (a, b, c). We use $\mathbf{0}_{3\times 3}$ to represent the 3×3 zero matrix, and $\mathcal{I}_3$ to represent the 3×3 identity matrix. To show the attack, we only need to focus on the first 9 rows of $\boldsymbol{\mathcal{T}}$ (hightlighted in red fonts).

1. If permute bits $(a, b, c) = (0, 0, 0)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}]^\top$;
2. If permute bits $(a, b, c) = (0, 0, 1)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3]^\top$;
3. If permute bits $(a, b, c) = (0, 1, 0)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3]^\top$;
4. If permute bits $(a, b, c) = (0, 1, 1)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}]^\top$;
5. If permute bits $(a, b, c) = (1, 0, 0)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}]^\top$;
6. If permute bits $(a, b, c) = (1, 0, 1)$, then $\boldsymbol{\mathcal{T}} = [\mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}]^\top$;
7. If permute bits $(a, b, c) = (1, 1, 0)$, then $\boldsymbol{\mathcal{T}} = [\mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}]^\top$;
8. If permute bits $(a, b, c) = (1, 1, 1)$, then $\boldsymbol{\mathcal{T}} = [\mathbf{0}_{3\times 3}\ \mathcal{I}_3\ \mathcal{I}_3\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}\ \mathbf{0}_{3\times 3}]^\top$.

It is clear that when $a = 0$, the first 9 rows of matrix $\boldsymbol{\mathcal{T}}$ are entirely composed of zeros, while for $a = 1$, they are not. Recall that the matrix $\boldsymbol{\mathcal{R}}$ is in $\mathrm{GF}(2)^{24\times 12}$. We choose an alternative representation to express certain rows of the matrix $\boldsymbol{\mathcal{R}}$:

$$\begin{aligned}
&\text{1-st row of } \boldsymbol{\mathcal{R}}: \ x_1A_0^{(1)} \oplus x_2A_0^{(2)} \oplus x_3A_0^{(3)} \oplus x_4B_0^{(1)} \oplus x_5B_0^{(2)} \oplus x_6B_0^{(3)} \oplus x_7C_0^{(1)} \oplus x_8C_0^{(2)} \oplus x_9C_0^{(3)}\\
&\text{2-nd row of } \boldsymbol{\mathcal{R}}: \ y_1A_0^{(1)} \oplus y_2A_0^{(2)} \oplus y_3A_0^{(3)} \oplus y_4B_0^{(1)} \oplus y_5B_0^{(2)} \oplus y_6B_0^{(3)} \oplus y_7C_0^{(1)} \oplus y_8C_0^{(2)} \oplus y_9C_0^{(3)}\\
&\text{7-th row of } \boldsymbol{\mathcal{R}}: \ z_1A_0^{(1)} \oplus z_2A_0^{(2)} \oplus z_3A_0^{(3)} \oplus z_4(B_0^{(1)} \oplus \Delta^{(1)}) \oplus z_5(B_0^{(2)} \oplus \Delta^{(2)}) \oplus z_6(B_0^{(3)} \oplus \Delta^{(3)}) \oplus z_7C_0^{(1)} \oplus z_8C_0^{(2)} \oplus z_9C_0^{(3)}\\
&\text{8-th row of } \boldsymbol{\mathcal{R}}: \ w_1A_0^{(1)} \oplus w_2A_0^{(2)} \oplus w_3A_0^{(3)} \oplus w_4(B_0^{(1)} \oplus \Delta^{(1)}) \oplus w_5(B_0^{(2)} \oplus \Delta^{(2)}) \oplus w_6(B_0^{(3)} \oplus \Delta^{(3)}) \oplus w_7C_0^{(1)} \oplus w_8C_0^{(2)} \oplus w_9C_0^{(3)},
\end{aligned}$$

and coefficients $x_i, y_i, z_i, w_i \in \{0, 1\}$, for $i \in [9]$.

Interestingly, by XORing rows 1, 2, 7, and 8 of the Equation 16, we observe that all terms related to the hash queries and ciphertext G_i cancel out. When permute bit $a = 0$, the result of the XORing is

$$(x_1 \oplus y_1 \oplus z_1 \oplus w_1)A_0^{(1)} \oplus \cdots \oplus (x_4 \oplus y_4 \oplus z_4 \oplus w_4)B_0^{(1)} \oplus (z_4 \oplus w_4)\Delta^{(1)} \oplus \cdots \oplus (x_9 \oplus y_9 \oplus z_9 \oplus w_9)C_0^{(3)} = \mathbf{0}.$$

Therefore, we can conclude that

$$x_4 \oplus y_4 \oplus z_4 \oplus w_4 = 0 \text{ and } z_4 \oplus w_4 = 0.$$

However, if permute bit $a = 1$, then the XORing result is

$$(x_1 \oplus y_1 \oplus z_1 \oplus w_1)A_0^{(1)} \oplus \cdots \oplus (x_4 \oplus y_4 \oplus z_4 \oplus w_4)B_0^{(1)} \oplus (z_4 \oplus w_4)\Delta^{(1)} \oplus \cdots \oplus (x_9 \oplus y_9 \oplus z_9 \oplus w_9)C_0^{(3)} = \Delta^{(1)} \oplus \Delta^{(2)}.$$

So we have

$$x_4 \oplus y_4 \oplus z_4 \oplus w_4 = 0 \text{ and } z_4 \oplus w_4 = 1.$$

If the evaluator's input labels are (A_0, B_0, C_0), she will obtain all coefficients x_i and y_i (this represents her marginal view). Depending on whether $x_4 \oplus y_4$ equals 0, the evaluator can deduce if $z_4 \oplus w_4$ (which equals $x_4 \oplus y_4$) is also 0, thereby revealing the permute bit a and breaching the privacy requirement.

Remark 9. Consequently, the challenge of achieving garbling for the 2-input AND gate with ciphertext less than 1.5κ bits still remains an open question. However, we believe that the impossibility result in [AHS24] is still meaningful. The authors have demonstrated that, for garbling an AND gate with fan-in greater than 2, simply introducing additional bases following the approach outlined in [RR21] is futile. We remark that our Theorem 4 and their impossibility result do not overlap and can complement each other.

D.7 Detailed Description of Construction #3

Recall that we introduce two bases as follows:

the left base	the right base
$\mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \oplus S^1_{00}$	$\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_0) \oplus S^2_{00}$
$\mathsf{H}(A_0) \oplus \mathsf{H}(B_1) \oplus S^1_{01}$	$\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^2_{01}$
$\mathsf{H}(A_1) \oplus \mathsf{H}(B_0) \oplus S^1_{10}$	$\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^2_{10}$
$\mathsf{H}(A_1) \oplus \mathsf{H}(B_1) \oplus S^1_{11}$	$\mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_1) \oplus S^2_{11}$

The output length of $\mathsf{H}(\cdot)$ is $\kappa/2$ bits[18]. Similarly, each S^u_{ij} (where $i, j \in \{0,1\}$, $u \in \{1,2\}$) also has a length of $\kappa/2$ bits. Each S^u_{ij} represents a linear combinations of values derived from the input labels, potentially through operations like extracting $\kappa/2$ bits from the input labels.

In the definition of MODEL-3′, we highlight that the generation of each base may depend on permute bits. This holds true for our construction as well. Therefore, we will categorize our discussion into four cases based on the different permute bits.

Case1 A_0, B_0 correspond to FALSE

In this case, following the order of the color bits, the fourth row of each base corresponds to the output of TRUE. If we set $S^1_{00} \oplus S^1_{10} = S^2_{01} \oplus S^2_{10}$, then the XOR of the 1-st and 3-rd rows of the left base will be equal to the XOR of the 2-nd and 3-rd rows of the right base, namely

$$\begin{aligned} &\mathsf{H}(A_0) \oplus \mathsf{H}(B_0) \oplus S^1_{00} \quad \oplus \quad \mathsf{H}(A_1) \oplus \mathsf{H}(B_0) \oplus S^1_{10} \\ =\ &\mathsf{H}(A_0) \oplus \mathsf{H}(A_0 \oplus B_1) \oplus S^2_{01} \quad \oplus \quad \mathsf{H}(A_1) \oplus \mathsf{H}(A_1 \oplus B_0) \oplus S^2_{10} \end{aligned}$$

For each operation, we need to select one column from the left base and one from the right base, and process them together. We use $(M^0_{00}[t], M^0_{01}[t], M^0_{10}[t], M^0_{11}[t])$ to denote the first column, and $(M^1_{00}[t], M^1_{01}[t], M^1_{10}[t], M^1_{11}[t])$ to denote the second column.

If $M^0_{00}[t] = M^0_{10}[t]$, then based on the above equation, we have $M^1_{01}[t] = M^1_{10}[t]$. In this case, the bit requiring flipping in the first column would either be $M^0_{01}[t]$ or $M^0_{11}[t]$, whereas in the second column, the bit requiring flipping would either be $M^1_{00}[t]$ or $M^1_{11}[t]$. Conversely, if $M^0_{00}[t] \neq M^0_{10}[t]$ then we must have $M^1_{01}[t] \neq M^1_{10}[t]$. In this situation, the bit requiring flipping in the first column would either be $M^0_{00}[t]$ or $M^0_{10}[t]$, whereas in the second column, the position requiring flipping would either be $M^1_{01}[t]$ or $M^1_{10}[t]$. In summary, Table 6 shows the corresponding bit-flip positions between the first and second columns.

Table 6: The corresponding bit-flip positions between two columns in **Case1** and **Case2**.

the first column	the second column
$(0,0)$	$(0,1)$ or $(1,0)$
$(0,1)$	$(0,0)$ or $(1,1)$
$(1,0)$	$(0,1)$ or $(1,0)$
$(1,1)$	$(0,0)$ or $(1,1)$

Based on the above table, two columns can be processed with a 3-bit ciphertext. For example, using ciphertext $(0,0,0)$ indicates that in the first column, the bit to be flipped is $M^0_{00}[t]$, and in the second column,

[18] For consistency with MODEL-3′, we might assume $\mathsf{H}(\cdot)$ to have an output length of κ bits. By applying a Contract function, we can reduce this to $\kappa/2$ bits.

the bit to be flipped is $M^1_{01}[t]$. Utilizing $(0,0,1)$ indicates that in the first column, the bit to be flipped is $M^0_{00}[t]$, while in the second column, the bit to be flipped is $M^1_{10}[t]$.

Case2 - Case4

We provide a brief description for **Case2**, **Case3**, and **Case4**:

- **Case2**: Both A_0 and B_1 correspond to FALSE. Setting $S^1_{01} \oplus S^1_{11} = S^2_{00} \oplus S^2_{11}$, then the positions requiring flipping on the two columns are specified in Table 6.
- **Case3**: Both A_1 and B_0 correspond to FALSE. By setting $S^1_{00} \oplus S^1_{10} = S^2_{00} \oplus S^2_{11}$, the positions that need flipping on the two columns are specified in Table 7.
- **Case4**: Both A_1 and B_1 correspond to FALSE. If we set $S^1_{01} \oplus S^1_{11} = S^2_{01} \oplus S^2_{10}$, the required positions for flipping on the two columns are outlined in Table 7.

Table 7: The corresponding bit-flip positions between two columns in **Case3** and **Case4**.

the first column	the second column
$(0,0)$	$(0,0)$ or $(1,1)$
$(0,1)$	$(0,1)$ or $(1,0)$
$(1,0)$	$(0,0)$ or $(1,1)$
$(1,1)$	$(0,1)$ or $(1,0)$

Up to this point, we have remained two unresolved problems:

1. There isn't a flip method that can cover every case. We illustrate with an example, when the ciphertext is $(0,0,0)$, in **Case1** and **Case2**, the bit in the second column that needs to be flipped is at position $(0,1)$. However, in **Case3** and **Case4**, the bit needing flipping is at position $(0,0)$. Due to privacy concerns, the evaluator cannot discern which case she is in, therefore she is uncertain about flipping the bit at position $(0,0)$ or $(0,1)$.
2. We have detailed the relationships that the S^u_{ij} (where $i,j \in \{0,1\}$, $u \in \{1,2\}$) must satisfy in different cases. However, we have not provided the specific values for each S^u_{ij} that would satisfy both the privacy and correctness requirements.

We will address these issues one by one.

D.7.1 A Unified Flip Method

For **Case1** and **Case2**, we will flip the two columns according to Table 6. Fortunately, for **Case3** and **Case4**, if we XOR the third and fourth rows of the right base with $1^{\kappa/2}$, then we can also flip according to Table 6 instead of Table 7.

We delve deeper into the reason behind this. For example, in **Case3** described in the previous section, we have $S^1_{00} \oplus S^1_{10} = S^2_{00} \oplus S^2_{11}$. This means the XOR of the first and third rows of the left base is equal to the XOR of the first and fourth rows of the right base. If $M^0_{00}[t] = M^0_{10}[t]$, then $M^1_{00}[t] = M^1_{11}[t]$. As a result, the flip position for the first column is either $(0,1)$ or $(1,1)$, and the flip position for the second column is either $(0,1)$ or $(1,0)$. However, if the third and fourth rows of the right base are XORed with $1^{\kappa/2}$, then the original $M^1_{00}[t] = M^1_{11}[t]$ will change to $M^1_{00}[t] \neq M^1_{11}[t]$, meaning the flip position for the second column becomes either $(0,0)$ or $(1,1)$. The same analysis can be applied to other situations. Therefore, we can flip according to Table 6 in both **Case3** and **Case4**.

However, we have not resolved the issue because if the garbler informs the evaluator to XOR the third and fourth rows of the right base with $1^{\kappa/2}$, it will breach privacy. We can address this issue by introducing some obfuscation, as detailed in Table 8.

For both **Case1** and **Case2**, actually there are two choices available: either keep all rows of the right base unchanged or XOR all rows of the right base with $1^{\kappa/2}$. Similarly, for **Case3** and **Case4**, there are also two choices. We previously discussed the first choice, where the third and fourth rows of the right base are XORed with $1^{\kappa/2}$, but it's equally feasible to XOR the first and second rows of the right base with $1^{\kappa/2}$. It is

Table 8: XOR operation choices for the right base in different cases.

color bits	**Case1 & Case2**		**Case3 & Case4**	
	Choice1	Choice2	Choice1	Choice2
$(0,0)$	0	$1^{\kappa/2}$	0	$1^{\kappa/2}$
$(0,1)$	0	$1^{\kappa/2}$	0	$1^{\kappa/2}$
$(1,0)$	0	$1^{\kappa/2}$	$1^{\kappa/2}$	0
$(1,1)$	0	$1^{\kappa/2}$	$1^{\kappa/2}$	0

straightforward to verify that regardless of which choice is taken from Table 8, two columns can be flipped according to the rules outlined in Table 6. The garbler will randomly choose one of the two choices based on the permute bits, that is, depending on which case he is in.

Finally, we can employ the technique from [KKS16] to ensure privacy. Specifically, the garbler uses four control bits to instruct the evaluator whether a given row should be XORed with $1^{\kappa/2}$. For example, a control bit of 0 indicates that the row does not need to be XORed with $1^{\kappa/2}$, while a value of 1 indicates the opposite. If the garbler opts for the first choice in **Case3**, then the four control bits are $(0,0,1,1)$. The garbler can encrypt each control bit using the least significant bit of each random oracle output. For clarity, the garbler might use the last bit of $\mathsf{H}(A_0,B_0)$, $\mathsf{H}(A_0,B_1)$, $\mathsf{H}(A_1,B_0)$, and $\mathsf{H}(A_1,B_1)$ to encrypt the four control bits respectively. The evaluator can decrypt and retrieve only one of these control bits. As illustrated in Table 8, regardless of what color bits the evaluator sees and regardless of which case the evaluator falls into, the probability that a row needs to be XORed with $1^{\kappa/2}$ remains at $1/2$. Therefore, the evaluator cannot distinguish which case she is in.

Up to this point, the first problem has been resolved. Only one problem remains, which is specify the specific values for S_{ij}^u, where $i,j \in \{0,1\}$, $u \in \{1,2\}$.

D.7.2 Equation Solving

To ensure compatibility with free-XOR, the two output labels should satisfy $C_0 \oplus C_1 = \Delta$, where Δ is a global offset. Let $\Delta^{(1)}$ denotes all the odd-indexed bits of Δ with a length of $\kappa/2$, and $\Delta^{(2)}$ represents all the even-indexed bits of Δ, also of length $\kappa/2$.

As discussed in Section 4.3, the compatibility with free-XOR requires the XOR among the four rows of the base to equal $\Delta \oplus 1^\kappa$. In this section's construction, the t-th column from the left base and the t-th column from the right base are taken during the t-th operation. This results in the $(2t-1)$-th and $(2t)$-th bits of the output label. Consequently, the XOR among the four rows of the left base should be $\Delta^{(1)} \oplus 1^{\kappa/2}$, and for the right base, it should be $\Delta^{(2)} \oplus 1^{\kappa/2}$. All random oracle queries will cancel out, simplifying the equations to:

$$S_{00}^1 \oplus S_{01}^1 \oplus S_{10}^1 \oplus S_{11}^1 = \Delta^{(1)} \oplus 1^{\kappa/2}$$
$$S_{00}^2 \oplus S_{01}^2 \oplus S_{10}^2 \oplus S_{11}^2 = \Delta^{(2)} \oplus 1^{\kappa/2}$$

In Table 9, we detail the conditions S_{ij}^u must satisfy in various cases.

The garbler cannot directly disclose to the evaluator the specific value of each S_{ij}^u, as doing so would enable the evaluator to determine which case she is in based on the relationships between different S_{ij}^u, thereby compromising privacy. As before, we once again utilize the idea from [KKS16] to address this challenge. Specifically, the garbler encrypts the composition of each S_{ij}^u using random oracle outputs. After obtaining A_α and B_β, the evaluator can only decrypt specific control bits to retrieve the values of $S_{\alpha\beta}^1$ and $S_{\alpha\beta}^2$. We must ensure that from this marginal view, the evaluator cannot deduce the permute bits. In other words, $S_{\alpha\beta}^1$ and $S_{\alpha\beta}^2$ must appear independent of the permute bits.

Determining the specific value of each S_{ij}^u is a mathematical challenge for which we seek solutions. We use $A_0^{(1)}$ to denote all the odd-indexed bits of the label A_0, with a length of $\kappa/2$, and $A_0^{(2)}$ to represent all the even-indexed bits of the label A_0. Similarly, we define $A_1^{(1)}$, $A_1^{(2)}$, $B_0^{(1)}$, $B_0^{(2)}$, $B_1^{(1)}$, and $B_1^{(2)}$. Based on the free-XOR requirements, it can be inferred that $A_0^{(1)} \oplus A_1^{(1)} = B_0^{(1)} \oplus B_1^{(1)} = \Delta^{(1)}$ and $A_0^{(2)} \oplus A_1^{(2)} =$

Table 9: The conditions that S_{ij}^u (where $i,j \in \{0,1\}$, $u \in \{1,2\}$) need to satisfy.

	conditions for S_{ij}^u
Case1	$S_{00}^1 \oplus S_{10}^1 = S_{01}^2 \oplus S_{10}^2$
Case2	$S_{01}^1 \oplus S_{11}^1 = S_{00}^2 \oplus S_{11}^2$
Case3	$S_{00}^1 \oplus S_{10}^1 = S_{00}^2 \oplus S_{11}^2$
Case4	$S_{01}^1 \oplus S_{11}^1 = S_{01}^2 \oplus S_{10}^2$
All Cases	$S_{00}^1 \oplus S_{01}^1 \oplus S_{10}^1 \oplus S_{11}^1 = \Delta^{(1)} \oplus 1^{\kappa/2}$ $S_{00}^2 \oplus S_{01}^2 \oplus S_{10}^2 \oplus S_{11}^2 = \Delta^{(2)} \oplus 1^{\kappa/2}$

$B_0^{(2)} \oplus B_1^{(2)} = \Delta^{(2)}$. For simplicity, we assume the specific form of each S_{ij}^u to be as follows:

$$
\begin{aligned}
S_{00}^1 &= x_{00}A_0^{(1)} \oplus x_{01}A_0^{(2)} \oplus x_{02}B_0^{(1)} \oplus x_{03}B_0^{(2)} \oplus 1^{\kappa/2} && x_{00},x_{01},x_{02},x_{03} \in \{0,1\}\\
S_{00}^2 &= x_{10}A_0^{(1)} \oplus x_{11}A_0^{(2)} \oplus x_{12}B_0^{(1)} \oplus x_{13}B_0^{(2)} \oplus 1^{\kappa/2} && x_{10},x_{11},x_{12},x_{13} \in \{0,1\}\\
S_{01}^1 &= x_{20}A_0^{(1)} \oplus x_{21}A_0^{(2)} \oplus x_{22}B_1^{(1)} \oplus x_{23}B_1^{(2)} && x_{20},x_{21},x_{22},x_{23} \in \{0,1\}\\
S_{01}^2 &= x_{30}A_0^{(1)} \oplus x_{31}A_0^{(2)} \oplus x_{32}B_1^{(1)} \oplus x_{33}B_1^{(2)} && x_{30},x_{31},x_{32},x_{33} \in \{0,1\}\\
S_{10}^1 &= x_{40}A_1^{(1)} \oplus x_{41}A_1^{(2)} \oplus x_{42}B_0^{(1)} \oplus x_{43}B_0^{(2)} && x_{40},x_{41},x_{42},x_{43} \in \{0,1\}\\
S_{10}^2 &= x_{50}A_1^{(1)} \oplus x_{51}A_1^{(2)} \oplus x_{52}B_0^{(1)} \oplus x_{53}B_0^{(2)} && x_{50},x_{51},x_{52},x_{53} \in \{0,1\}\\
S_{11}^1 &= x_{60}A_1^{(1)} \oplus x_{61}A_1^{(2)} \oplus x_{62}B_1^{(1)} \oplus x_{63}B_1^{(2)} && x_{60},x_{61},x_{62},x_{63} \in \{0,1\}\\
S_{11}^2 &= x_{70}A_1^{(1)} \oplus x_{71}A_1^{(2)} \oplus x_{72}B_1^{(1)} \oplus x_{73}B_1^{(2)} && x_{70},x_{71},x_{72},x_{73} \in \{0,1\}
\end{aligned}
$$

Next, we only need to determine the unknown coefficients above, namely x_{ij}, where $0 \le i \le 7$ and $0 \le j \le 3$. Like [RR21], through computer search, we obtain four solutions for each case, as shown in Table 10, 11, 12, and 13.

Table 10: Solutions for **Case1**.

	Choice1	Choice2	Choice3	Choice4
$(x_{00},x_{01},x_{02},x_{03})$	$(1,1,0,1)$	$(0,1,0,0)$	$(1,0,1,0)$	$(0,0,1,1)$
$(x_{10},x_{11},x_{12},x_{13})$	$(0,0,0,1)$	$(0,1,1,0)$	$(1,1,1,1)$	$(1,0,0,0)$
$(x_{20},x_{21},x_{22},x_{23})$	$(1,0,0,0)$	$(0,0,0,1)$	$(1,1,1,1)$	$(0,1,1,0)$
$(x_{30},x_{31},x_{32},x_{33})$	$(0,1,1,0)$	$(0,0,0,1)$	$(1,0,0,0)$	$(1,1,1,1)$
$(x_{40},x_{41},x_{42},x_{43})$	$(0,0,1,0)$	$(1,0,1,1)$	$(0,1,0,1)$	$(1,1,0,0)$
$(x_{50},x_{51},x_{52},x_{53})$	$(1,0,0,1)$	$(1,1,1,0)$	$(0,1,1,1)$	$(0,0,0,0)$
$(x_{60},x_{61},x_{62},x_{63})$	$(0,1,1,1)$	$(1,1,1,0)$	$(0,0,0,0)$	$(1,0,0,1)$
$(x_{70},x_{71},x_{72},x_{73})$	$(1,1,1,0)$	$(1,0,0,1)$	$(0,0,0,0)$	$(0,1,1,1)$

After the garbler has randomly selected the permute bits and determined the specific case he is in, he processes to randomly choose one of the four available choices within that case. The garbler then encrypts each x_{ij}, where $0 \le i \le 7$ and $0 \le j \le 3$, and sends them to the evaluator. Once the evaluator obtains A_α and B_β, she can decrypt only 8 of these bits, allowing her to determine the values of $S_{\alpha\beta}^1$ and $S_{\alpha\beta}^2$.

A noteworthy observation is that every combination of $S_{\alpha\beta}^1$ and $S_{\alpha\beta}^2$ appears in any given case[19]. For example, the combination of S_{01}^1 and S_{01}^2 from *Choice1* in **Case1** is also found in *Choice4* of **Case2**, *Choice3* of

[19] Although the total number of x_{ij} is 32, implying the need for a 32-bit additional ciphertext to be transmitted, this can be reduced to just 8 bits based on our observation. This is due to the fact that there are only 4 possible values for $(S_{\alpha\beta}^1, S_{\alpha\beta}^2)$, with each value being representable using just 2 bits. Considering that there are 4 potential values for (α, β), a total of only 8 bits are necessary.

Table 11: Solutions for **Case2**.

	Choice1	Choice2	Choice3	Choice4
$(x_{00}, x_{01}, x_{02}, x_{03})$	$(1,1,0,1)$	$(0,1,0,0)$	$(1,0,1,0)$	$(0,0,1,1)$
$(x_{10}, x_{11}, x_{12}, x_{13})$	$(0,0,0,1)$	$(0,1,1,0)$	$(1,1,1,1)$	$(1,0,0,0)$
$(x_{20}, x_{21}, x_{22}, x_{23})$	$(0,1,1,0)$	$(1,1,1,1)$	$(0,0,0,1)$	$(1,0,0,0)$
$(x_{30}, x_{31}, x_{32}, x_{33})$	$(1,1,1,1)$	$(1,0,0,0)$	$(0,0,0,1)$	$(0,1,1,0)$
$(x_{40}, x_{41}, x_{42}, x_{43})$	$(0,1,0,1)$	$(1,1,0,0)$	$(0,0,1,0)$	$(1,0,1,1)$
$(x_{50}, x_{51}, x_{52}, x_{53})$	$(0,1,1,1)$	$(0,0,0,0)$	$(1,0,0,1)$	$(1,1,1,0)$
$(x_{60}, x_{61}, x_{62}, x_{63})$	$(1,1,1,0)$	$(0,1,1,1)$	$(1,0,0,1)$	$(0,0,0,0)$
$(x_{70}, x_{71}, x_{72}, x_{73})$	$(1,0,0,1)$	$(1,1,1,0)$	$(0,1,1,1)$	$(0,0,0,0)$

Table 12: Solutions for **Case3**.

	Choice1	Choice2	Choice3	Choice4
$(x_{00}, x_{01}, x_{02}, x_{03})$	$(1,1,0,1)$	$(0,1,0,0)$	$(1,0,1,0)$	$(0,0,1,1)$
$(x_{10}, x_{11}, x_{12}, x_{13})$	$(0,0,0,1)$	$(0,1,1,0)$	$(1,1,1,1)$	$(1,0,0,0)$
$(x_{20}, x_{21}, x_{22}, x_{23})$	$(1,1,1,1)$	$(0,1,1,0)$	$(1,0,0,0)$	$(0,0,0,1)$
$(x_{30}, x_{31}, x_{32}, x_{33})$	$(1,0,0,0)$	$(1,1,1,1)$	$(0,1,1,0)$	$(0,0,0,1)$
$(x_{40}, x_{41}, x_{42}, x_{43})$	$(1,0,1,1)$	$(0,0,1,0)$	$(1,1,0,0)$	$(0,1,0,1)$
$(x_{50}, x_{51}, x_{52}, x_{53})$	$(1,1,1,0)$	$(1,0,0,1)$	$(0,0,0,0)$	$(0,1,1,1)$
$(x_{60}, x_{61}, x_{62}, x_{63})$	$(1,0,0,1)$	$(0,0,0,0)$	$(1,1,1,0)$	$(0,1,1,1)$
$(x_{70}, x_{71}, x_{72}, x_{73})$	$(0,1,1,1)$	$(0,0,0,0)$	$(1,0,0,1)$	$(1,1,1,0)$

Table 13: Solutions for **Case4**.

	Choice1	Choice2	Choice3	Choice4
$(x_{00}, x_{01}, x_{02}, x_{03})$	$(1,1,0,1)$	$(0,1,0,0)$	$(1,0,1,0)$	$(0,0,1,1)$
$(x_{10}, x_{11}, x_{12}, x_{13})$	$(0,0,0,1)$	$(0,1,1,0)$	$(1,1,1,1)$	$(1,0,0,0)$
$(x_{20}, x_{21}, x_{22}, x_{23})$	$(0,0,0,1)$	$(1,0,0,0)$	$(0,1,1,0)$	$(1,1,1,1)$
$(x_{30}, x_{31}, x_{32}, x_{33})$	$(0,0,0,1)$	$(0,1,1,0)$	$(1,1,1,1)$	$(1,0,0,0)$
$(x_{40}, x_{41}, x_{42}, x_{43})$	$(1,1,0,0)$	$(0,1,0,1)$	$(1,0,1,1)$	$(0,0,1,0)$
$(x_{50}, x_{51}, x_{52}, x_{53})$	$(0,0,0,0)$	$(0,1,1,1)$	$(1,1,1,0)$	$(1,0,0,1)$
$(x_{60}, x_{61}, x_{62}, x_{63})$	$(0,0,0,0)$	$(1,0,0,1)$	$(0,1,1,1)$	$(1,1,1,0)$
$(x_{70}, x_{71}, x_{72}, x_{73})$	$(0,0,0,0)$	$(0,1,1,1)$	$(1,1,1,0)$	$(1,0,0,1)$

Case3, and *Choice2* of **Case4**. As a result, the evaluator, based on the values of S_{01}^1 and S_{01}^2, cannot deduce which case she is in.

In Appendix D.8, we provide a formal description and security proof for this construction.

D.8 Security Proof of Construction #3

For the construction presented in Section 6.4, we provide a formal description and security proof.

Formal Description. Our garbling scheme is formally described in Figures 11 to 15. The main body of the algorithm Gb, along with En and De, is essentially consistent with the description in Appendix B.4. The only point to note is that in the Gb algorithm, the definition of H'' used for producing decoding information is $\{0,1\}^\kappa \times \mathbb{Z} \to \{0,1\}^\kappa$.

In the subroutine algorithm MajVote, the Sample function is first called, returning $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$. Specifically, $\boldsymbol{\mathcal{S}}$ is a certain combination of input labels, which we have discussed in detail in Appendix D.7. Another value $\boldsymbol{P}$ addresses the inconsistency of bit-flipping rules between (**Case1**, **Case2**) and (**Case3**, **Case4**). Its potential values are shown in Table 8. The Sample function randomly selects from these possible values. Following that, two bases are generated based on $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$. In this context, the hash function H we employ is $\{0,1\}^\kappa \times \mathbb{Z} \to \{0,1\}^{\kappa/2}$. Next, for the first base, we employ the majority voting as usual to produce one bit of the output label and a ciphertext of two bits indicating which position needs to be flipped. As observed from Table 6, if the position to be flipped in the first column is determined, there are only two possible scenarios for the position to be flipped in the second column. Thus, for the column retrieved from the second base, we only need to use the first bit of the flipped position to indicate which position requires flipping. Lastly, it is imperative to encrypt $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$ before transmitting them to the evaluator. The rationale behind this is that if the evaluator gains full knowledge of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$, it would violate privacy (a point we have already discussed in detail). We employ another hash function, $\mathsf{H}' : \{0,1\}^{2\kappa} \times \mathbb{Z} \to \{0,1\}^3$, to perform one-time pad encryption. Given that the evaluator possesses input labels A_α and B_β, she can only decrypt the $(2\alpha+\beta+1)$-th row.

For the evaluation algorithm Ev, the initial step involves decrypting to obtain partial information of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$. Utilizing this partial information, the evaluator can reconstruct a row of the left base and a row of the right base. Next, the evaluator can perform flipping according to the rules laid out in Table 6. For a given iteration, suppose the ciphertext is (g_0, g_1, g_2). The position to be flipped in the first column is given by $g_0||g_1$. Based on Table 6, we can deduce that the position to be flipped in the second column is $g_2||g_1 \oplus g_2 \oplus 1$.

It should be noted that in our scheme description, we only considered reducing the ciphertext length required to garble an AND gate to $1.5\kappa+O(1)$. We did not address the minimization of hash queries. Indeed, simple optimizations can be implemented. For example, we can define H as $\{0,1\}^\kappa \times \mathbb{Z} \to \{0,1\}^{\kappa/2+3}$, eliminating the need for additional H' queries. Instead, one can simply truncate the last 3 bits of H as the output of H'.

Security Proof. We need to define the security property that the hash function H must satisfy. Here, we no longer employ the TCCR for naturally derived keys as in Definition 3. Instead, we utilize the randomized TCCR (RTCCR) assumption from [RR21]. We adopt this approach for reasons similar to [RR21]; for more details, please refer to Section 2.3 of their work.

Definition 4. *Let $\mathcal{H}$ be a family of hash functions where each function maps $\{0,1\}^n \times \mathbb{Z} \to \{0,1\}^m$, $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^n$ be a random oracle, $\mathcal{L}$ be a linear function set where each $L \in \mathcal{L}$ maps $\{0,1\}^n \to \{0,1\}^m$, and let $\mathcal{R}$ be a distribution over $\{0,1\}^n$. Sample $\mathsf{H} \leftarrow \mathcal{H}$, $\Delta \leftarrow \mathcal{R}$, and define an oracle $\mathcal{O}_{\mathsf{H},\Delta}^{rtccr}(x,t,L) = \mathsf{H}(x \oplus \Delta, t) \oplus L(\Delta)$. We say the $\mathcal{H}$ is randomized TCCR for $\mathcal{L}$ if, for any PPT distinguisheres D_1, D_2 that never repeat an oracle query to $\mathcal{O}_{\mathsf{H},\Delta}^{rtccr}(x,t,L)$ on the same (x,t),*

$$\left| \Pr_{\mathsf{H},\Delta}[v \leftarrow \mathcal{A}_1^{\mathsf{H},\mathcal{O}_{\mathsf{H},\Delta}^{rtccr}}; \mathcal{A}_2(v,\mathsf{H}) = 1] - \Pr_{\mathsf{H},\mathsf{RO}}[v \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{RO}}; \mathcal{A}_2(v,\mathsf{H}) = 1] \right| \textit{ is negligible.}$$

In the above definition, the hash function H is sampled from $\mathcal{H}$. Consequently, we need to make some modifications to the scheme described in Figures 11 to 15. In the Gb algorithm, the garbler should first sample $\mathsf{H} \leftarrow \mathcal{H}$. Then, this H is incorporated as a part of the garbled circuit F. For the evaluator, it is necessary to extract H from F and subsequently make oracle queries.

Garbling algorithm $\mathsf{Gb}(1^\kappa, f)$:

$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$\Delta \leftarrow \{0,1\}^{\kappa-1}$
$\Delta = \Delta || 1$
for $i = 1$ to inputs :
  $W_i \leftarrow \{0,1\}^{\kappa-1}$
  $W_i = W_i || 0$
  $\pi_i \leftarrow \{0,1\}$
for $i = \mathsf{inputs} + 1$ to $|f|$:
  $(A_0, B_0) := (W_{\mathsf{in}_1(i)}, W_{\mathsf{in}_2(i)})$
  $(\pi_A, \pi_B) := (\pi_{\mathsf{in}_1(i)}, \pi_{\mathsf{in}_2(i)})$
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    $W_i := A_0 \oplus B_0$
    $\pi_i := \pi_A \oplus \pi_B$
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $(C, F_i) := \mathsf{MajVote}(A_0, B_0, \Delta, \pi_A, \pi_B, i)$
    $\pi_i := \mathsf{lsb}(C)$
    $W_i := C \oplus \pi_i \Delta$
for $i \in \mathsf{outputs}, j \in \{0,1\}$:
  $d_i^j := \mathsf{H}''(W_i \oplus (j \oplus \pi_i)\Delta, i)$
return $(F,\ e = (\Delta, W_{[1,\mathsf{inputs}]}, \pi_{[1,\mathsf{inputs}]}),\ d)$

Private algorithm $\mathsf{Sample}(\pi_A, \pi_B)$:

if $(\pi_A, \pi_B) = (0,0)$:
  $\boldsymbol{\mathcal{S}} \leftarrow$ {four choices in Table 10} // $\boldsymbol{\mathcal{S}} \in \mathrm{GF}(2)^{8\times 4}$
  $\boldsymbol{P} \leftarrow \{(0,0,0,0),(1,1,1,1)\}$ // $\boldsymbol{P} \in \mathrm{GF}(2)^{1\times 4}$
if $(\pi_A, \pi_B) = (0,1)$:
  $\boldsymbol{\mathcal{S}} \leftarrow$ {four choices in Table 11}
  $\boldsymbol{P} \leftarrow \{(0,0,0,0),(1,1,1,1)\}$
if $(\pi_A, \pi_B) = (1,0)$:
  $\boldsymbol{\mathcal{S}} \leftarrow$ {four choices in Table 12}
  $\boldsymbol{P} \leftarrow \{(0,0,1,1),(1,1,0,0)\}$
if $(\pi_A, \pi_B) = (1,1)$:
  $\boldsymbol{\mathcal{S}} \leftarrow$ {four choices in Table 13}
  $\boldsymbol{P} \leftarrow \{(0,0,1,1),(1,1,0,0)\}$
return $(\boldsymbol{\mathcal{S}},\ \boldsymbol{P})$

Private algorithm $\mathsf{MajVote}(A_0, B_0, \Delta, \pi_A, \pi_B, i)$:

$(\boldsymbol{\mathcal{S}},\ \boldsymbol{P}) := \mathsf{Sample}(\pi_A, \pi_B)$

Parse matrix $\boldsymbol{\mathcal{S}}$ as $\begin{bmatrix} \boldsymbol{\mathcal{S}}^1_{00} & \boldsymbol{\mathcal{S}}^2_{00} \\ \boldsymbol{\mathcal{S}}^1_{01} & \boldsymbol{\mathcal{S}}^2_{01} \\ \boldsymbol{\mathcal{S}}^1_{10} & \boldsymbol{\mathcal{S}}^2_{10} \\ \boldsymbol{\mathcal{S}}^1_{11} & \boldsymbol{\mathcal{S}}^2_{11} \end{bmatrix} := \begin{bmatrix} \boxed{x_{00} & x_{01}} & x_{02} & x_{03} \\ \boxed{x_{10} & x_{11}} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \\ x_{40} & x_{41} & x_{42} & x_{43} \\ x_{50} & x_{51} & x_{52} & x_{53} \\ x_{60} & x_{61} & x_{62} & x_{63} \\ x_{70} & x_{71} & x_{72} & x_{73} \end{bmatrix}$

Parse $\boldsymbol{P}$ as $(p_{00}, p_{01}, p_{10}, p_{11})$

Figure 11: The Gb algorithm of the Construction #3.

$$\begin{bmatrix} M^0_{00} \\ M^1_{00} \\ M^0_{01} \\ M^1_{01} \\ M^0_{10} \\ M^1_{10} \\ M^0_{11} \\ M^1_{11} \end{bmatrix} := \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \mathsf{H}(A_0, 3i-3) \\ \mathsf{H}(A_0 \oplus \Delta, 3i-3) \\ \mathsf{H}(B_0, 3i-2) \\ \mathsf{H}(B_0 \oplus \Delta, 3i-2) \\ \mathsf{H}(A_0 \oplus B_0, 3i-1) \\ \mathsf{H}(A_0 \oplus B_0 \oplus \Delta, 3i-1) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mathcal{S}}^1_{00} & \boldsymbol{\mathcal{S}}^2_{00} & \mathbf{0} \\ \boldsymbol{\mathcal{S}}^1_{01} & \boldsymbol{\mathcal{S}}^2_{01} & \boldsymbol{\mathcal{S}}^2_{01} \\ \boldsymbol{\mathcal{S}}^1_{10} & \boldsymbol{\mathcal{S}}^2_{10} & \boldsymbol{\mathcal{S}}^1_{10} \\ \boldsymbol{\mathcal{S}}^1_{11} & \boldsymbol{\mathcal{S}}^2_{11} & \boldsymbol{\mathcal{S}}^1_{11} + \boldsymbol{\mathcal{S}}^2_{11} \end{bmatrix} \times$$

$$\begin{bmatrix} A^{(1)}_0 \\ A^{(2)}_0 \\ B^{(1)}_0 \\ B^{(2)}_0 \\ \Delta^{(1)} \\ \Delta^{(2)} \end{bmatrix} + \begin{bmatrix} 1 \\ p_{00} \oplus 1 \\ 0 \\ p_{01} \\ 0 \\ p_{10} \\ 0 \\ p_{11} \end{bmatrix} \times 1^{\frac{\kappa}{2}}$$

for $t = 1$ to $\kappa/2$:

  if $M^0_{\pi_A||\pi_B}[t] = M^0_{\pi_A\oplus 1||\pi_B}[t] = M^0_{\pi_A||\pi_B\oplus 1}[t]$:

    $(c_0, g_0||g_1) := (M^0_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1||\pi_B \oplus 1)$

  else if $M^0_{\pi_A||\pi_B}[t] = M^0_{\pi_A\oplus 1||\pi_B}[t]$:

    $(c_0, g_0||g_1) := (M^0_{\pi_A||\pi_B}[t],\ \pi_A||\pi_B \oplus 1)$

  else if $M^0_{\pi_A||\pi_B}[t] = M^0_{\pi_A||\pi_B\oplus 1}[t]$:

    $(c_0, g_0||g_1) := (M^0_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1||\pi_B)$

  else if $M^0_{\pi_A||\pi_B\oplus 1}[t] = M^0_{\pi_A\oplus 1||\pi_B}[t]$:

    $(c_0, g_0||g_1) := (M^0_{\pi_A||\pi_B}[t] \oplus 1,\ \pi_A||\pi_B)$

  if $M^1_{\pi_A||\pi_B}[t] = M^1_{\pi_A\oplus 1||\pi_B}[t] = M^1_{\pi_A||\pi_B\oplus 1}[t]$:

    $(c_1, g_2) := (M^1_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1)$

  else if $M^1_{\pi_A||\pi_B}[t] = M^1_{\pi_A\oplus 1||\pi_B}[t]$:

    $(c_1, g_2) := (M^1_{\pi_A||\pi_B}[t],\ \pi_A)$

  else if $M^1_{\pi_A||\pi_B}[t] = M^1_{\pi_A||\pi_B\oplus 1}[t]$:

    $(c_1, g_2) := (M^1_{\pi_A||\pi_B}[t],\ \pi_A \oplus 1)$

  else if $M^1_{\pi_A||\pi_B\oplus 1}[t] = M^1_{\pi_A\oplus 1||\pi_B}[t]$:

    $(c_1, g_2) := (M^1_{\pi_A||\pi_B}[t] \oplus 1,\ \pi_A)$

  $(C[t], G[t]) := (c_0||c_1,\ g_0||g_1||g_2)$

Compress $\boldsymbol{\mathcal{S}}$ into $\begin{bmatrix} S^1_{00} & S^2_{00} \\ S^1_{01} & S^2_{01} \\ S^1_{10} & S^2_{10} \\ S^1_{11} & S^2_{11} \end{bmatrix}$

$$\begin{bmatrix} Z_{00} \\ Z_{01} \\ Z_{10} \\ Z_{11} \end{bmatrix} = \begin{bmatrix} S^1_{00}||S^2_{00}||p_{00} \\ S^1_{01}||S^2_{01}||p_{01} \\ S^1_{10}||S^2_{10}||p_{10} \\ S^1_{11}||S^2_{11}||p_{11} \end{bmatrix} + \begin{bmatrix} \mathsf{H}'(A_0, B_0, i) \\ \mathsf{H}'(A_0, B_0 \oplus \Delta, i) \\ \mathsf{H}'(A_0 \oplus \Delta, B_0, i) \\ \mathsf{H}'(A_0 \oplus \Delta, B_0 \oplus \Delta, i) \end{bmatrix}$$

return $(C,\ (G, (Z_{00}, Z_{01}, Z_{10}, Z_{11})))$

Figure 12: Continuation of the Gb algorithm. As indicated by Footnote 19, we can compress $\boldsymbol{\mathcal{S}}$ into 8 bits. For example, $x_{00}||x_{01}||x_{02}||x_{03}||x_{10}||x_{11}||x_{12}||x_{13}$ can be represented by 2 bits, namely S^1_{00} and S^2_{00}.

Evaluation algorithm $\mathsf{Ev}(F, X)$:

for $i = \mathsf{inputs} + 1$ to $|f|$:
  $(A, B) := (X_{\mathsf{in}_1(i)}, X_{\mathsf{in}_2(i)})$
  $(\alpha, \beta) := (\mathsf{lsb}(A), \mathsf{lsb}(B))$
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    $X_i := A \oplus B$
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $(G, (Z_{00}, Z_{01}, Z_{10}, Z_{11})) := F_i$
    $s_0||s_1||p := \mathsf{H}'(A, B, i) \oplus Z_{\alpha||\beta}$
    $x_0||x_1||x_2||x_3||x_4||x_5||x_6||x_7 :=$ expand $s_0||s_1$ into 8 bits
    $M^0 := \mathsf{H}(A, 3i-3) \oplus \mathsf{H}(B, 3i-2) \oplus x_0 A^{(1)} \oplus x_1 A^{(2)} \oplus x_2 B^{(1)} \oplus x_3 B^{(2)}$
    $M^1 := \mathsf{H}(A, 3i-3) \oplus \mathsf{H}(A \oplus B, 3i-1) \oplus x_4 A^{(1)} \oplus x_5 A^{(2)} \oplus x_6 B^{(1)} \oplus$
      $x_7 B^{(2)} \oplus p 1^{\kappa/2}$
    if $\alpha||\beta = 00$:
      $M^0 = M^0 \oplus 1^{\kappa/2}$
      $M^1 = M^1 \oplus 1^{\kappa/2}$
    for $t = 1$ to $\kappa/2$:
      $g_0||g_1||g_2 := G[t]$
      $X_i[t] := M^0[t]||M^1[t]$
      if $g_0||g_1 = \alpha||\beta$:
        $X_i[t] = X_i[t] \oplus 10$
      if $g_2||g_1 \oplus g_2 \oplus 1 = \alpha||\beta$:
        $X_i[t] = X_i[t] \oplus 01$
for $i \in \mathsf{outputs}$:
  $Y_i := X_i$
return Y

Figure 13: The Ev algorithm of the Construction #3.

Encoding algorithm $\mathsf{En}(e = (\Delta, W, \pi), x)$:

for $i = 1$ to inputs :
  $X_i := W_i \oplus (x_i \oplus \pi_i)\Delta$
return X

Figure 14: The En algorithm of the Construction #3.

Decoding algorithm $\mathsf{De}(d, Y)$:

for $i \in \mathsf{outputs}$:
  if $\exists\ j$ such that $d_i^j = \mathsf{H}''(Y_i, i)$:
    $y_i := j$
  else
    abort
return y

Figure 15: The De algorithm of the Construction #3.

Now, we formally prove the security of our scheme.

Theorem 9. *Let $\mathcal{H}$ be a family of hash functions where $n = \kappa$, $m = \kappa/2$, and let $\mathcal{L} = \{L_{ab}(\Delta) = a\Delta^{(1)} \oplus b\Delta^{(2)}\}$. If $\mathcal{H}$ is randomized TCCR for $\mathcal{L}$, and H', H'' are random oracles, then our construction described in Figures 11 to 15 is a secure garbling scheme.*

Proof. Our proof approach is similar to that of Theorem 8. The correctness of our construction can be directly inferred from the description in Section 6.4 and Appendix D.7. The proof of obliviousness can be directly derived from the proof of privacy, and the proof of authenticity aligns with that of Theorem 8. Therefore, we focus solely on proving that this construction meets the privacy requirement.

Privacy. We employ hybrid arguments to prove this property. The algorithms of the simulator Sim is depicted in Figure 16. For each AND gate, in addition to generating 1.5κ random bits as the main ciphertext, a constant number of random bits must also be produced as the additional ciphertext.

$\underline{\mathsf{Sim}_{priv}(1^\kappa, f, y = f(x))}$:
$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$(F, X) \leftarrow \mathsf{Sim}_{obliv}(1^\kappa, f)$
$Y := \mathsf{Ev}(F, X)$
for $i \in \mathsf{outputs}$:
  $d_i^{y_i} := \mathsf{H}''(Y_i, i)$
  $d_i^{y_i \oplus 1} \leftarrow \{0,1\}^\kappa$
return $(F,\ X,\ d)$

$\underline{\mathsf{Sim}_{obliv}(1^\kappa, f)}$:
$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{type}) := f$
$\mathsf{H} \leftarrow \mathcal{H}$
for $i = 1$ to inputs :
  $E_i \leftarrow \{0,1\}^\kappa$
for $i = \mathsf{inputs} + 1$ to $|f|$:
  if $\mathsf{type}(i) = \mathsf{XOR}$:
    continue
  else if $\mathsf{type}(i) = \mathsf{AND}$:
    $G \leftarrow \{0,1\}^{1.5\kappa}$
    for $s, t \in \{0,1\}$:
      $Z_{s||t} \leftarrow \{0,1\}^3$
    $P_i := (G, (Z_{00}, Z_{01}, Z_{10}, Z_{11}))$
return $(F = (P, \mathsf{H}),\ X = E)$

Figure 16: The algorithms of Sim for proof of privacy.

Hybrid1: In this hybrid, we make the following modifications to the real garbling:

- Track all active wire labels, instead of tracking labels with color bits of 0.
- Use an equivalent representation for majority voting. Assume that for the i-th AND gate's two input wires, the active input labels are A and B, and the permute bits are π_A and π_B. Next, we discuss based on different cases:
  - $(\pi_A, \pi_B) = (0,0)$: We compute the XOR of the first and second rows of the left base as T_0^0, and the XOR between the first row and the third row as T_1^0:

$$\begin{aligned} T_0^0 :=& \mathsf{H}(B, 3i-2) \oplus \mathsf{H}(B \oplus \Delta, 3i-2) \oplus s_0^0 A^{(1)} \oplus s_0^1 A^{(2)} \oplus s_0^2 B^{(1)} \oplus s_0^3 B^{(2)} \oplus s_0^4 \Delta^{(1)} \oplus \\ & s_0^5 \Delta^{(2)} \oplus s_0^6 1^{\kappa/2} \\ T_1^0 :=& \mathsf{H}(A, 3i-3) \oplus \mathsf{H}(A \oplus \Delta, 3i-3) \oplus s_1^0 A^{(1)} \oplus s_1^1 A^{(2)} \oplus s_1^2 B^{(1)} \oplus s_1^3 B^{(2)} \oplus s_1^4 \Delta^{(1)} \oplus \\ & s_1^5 \Delta^{(2)} \oplus s_1^6 1^{\kappa/2} \end{aligned}$$

    where each s_u^v is determined by $\boldsymbol{S}$ and $\boldsymbol{P}$. Similarly, we compute the XOR between the first and second rows of the right base as T_0^1, and the XOR between the second and third rows of the right

base as T_1^1:

$$T_0^1 := \mathsf{H}(A \oplus B, 3i-1) \oplus \mathsf{H}(A \oplus B \oplus \Delta, 3i-1) \oplus t_0^0 A^{(1)} \oplus t_0^1 A^{(2)} \oplus t_0^2 B^{(1)} \oplus t_0^3 B^{(2)} \oplus t_0^4 \Delta^{(1)} \oplus t_0^5 \Delta^{(2)} \oplus t_0^6 1^{\kappa/2}$$

$$T_1^1 := \mathsf{H}(A, 3i-3) \oplus \mathsf{H}(A \oplus \Delta, 3i-3) \oplus t_1^0 A^{(1)} \oplus t_1^1 A^{(2)} \oplus t_1^2 B^{(1)} \oplus t_1^3 B^{(2)} \oplus t_1^4 \Delta^{(1)} \oplus t_1^5 \Delta^{(2)} \oplus t_1^6 1^{\kappa/2}$$

where each t_u^v is also determined by $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$. As indicated by Table 9, under the condition of $(\pi_A, \pi_B) = (0,0)$, we have $S_{00}^1 \oplus S_{10}^1 = S_{01}^2 \oplus S_{10}^2$, namely $T_1^0 = T_1^1 \oplus (s_1^6 \oplus t_1^6)1^{\kappa/2}$. Therefore, using T_0^0, T_1^0, and T_0^1, we can represent the majority voting process.

– Other cases: For the cases where $(\pi_A, \pi_B) \neq (0,0)$, we can conduct a similar analysis. We find that in any case, the majority voting process can be represented using the aforementioned three T forms.

- We remove the process of generating the output label of an AND gate within the subroutine MajVote. In the main Gb algorithm, the output label is obtained through evaluation using the ciphertext and input labels.

This hybrid is derived from the real garbling through simple variable substitutions and equivalent operation replacements; therefore, their distributions are identical.

Hybrid2: In this hybrid, we replace the hash queries using Δ with random oracle queries. Recall that in Definition 4, we defined the oracle $\mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(x,t,L) = \mathsf{H}(x \oplus \Delta, t) \oplus L(\Delta)$. Since $L \in \{L_{ab}(\Delta) = a\Delta^{(1)} \oplus b\Delta^{(2)}\}$, we can express $\mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(x,t,L)$ as $\mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(x,t,a,b) = \mathsf{H}(x \oplus \Delta, t) \oplus a\Delta^{(1)} \oplus b\Delta^{(2)}$.

Assuming $(\pi_A, \pi_B) = (0,0)$, we can express the previously mentioned T_0^0, T_1^0, and T_0^1 in the following form:

- $T_0^0 = \mathsf{H}(B, 3i-2) \oplus \mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(B, 3i-2, s_0^4, s_0^5) \oplus s_0^0 A^{(1)} \oplus s_0^1 A^{(2)} \oplus s_0^2 B^{(1)} \oplus s_0^3 B^{(2)} \oplus s_0^6 1^{\kappa/2}$
- $T_1^0 = \mathsf{H}(A, 3i-3) \oplus \mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(A, 3i-3, s_1^4, s_1^5) \oplus s_1^0 A^{(1)} \oplus s_1^1 A^{(2)} \oplus s_1^2 B^{(1)} \oplus s_1^3 B^{(2)} \oplus s_1^6 1^{\kappa/2}$
- $T_0^1 = \mathsf{H}(A \oplus B, 3i-1) \oplus \mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(A \oplus B, 3i-1, t_0^4, t_0^5) \oplus t_0^0 A^{(1)} \oplus t_0^1 A^{(2)} \oplus t_0^2 B^{(1)} \oplus t_0^3 B^{(2)} \oplus t_0^6 1^{\kappa/2}$

Due to the property of RTCCR, we can replace $\mathcal{O}^{rtccr}_{\mathsf{H},\Delta}(x,t,a,b)$ with a random value. Therefore, we obtain three independent random values: T_0^0, T_1^0, and T_0^1. For the cases where $(\pi_A, \pi_B) \neq (0,0)$, we can derive the same result.

Hybrid3: In this hybrid, the ciphertext for each AND gate is randomly generated. Specifically, for each ciphertext $(G, (Z_{00}, Z_{01}, Z_{10}, Z_{11}))$, we have $G \leftarrow \{0,1\}^{\kappa/2}$ and $Z_{uv} \leftarrow \{0,1\}^3$ for $u, v \in \{0,1\}$. In every case, the three T values are independently random. Therefore, the main ciphertext G generated based on these three values is also random. Given our assumption that H′ is a random oracle, the values $(Z_{00}, Z_{01}, Z_{10}, Z_{11})$ generated in Hybrid2 are indistinguishable from those randomly generated here. Additionally, during the process of evaluating an AND gate to obtain the output label, the marginal view of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{P}$ is independent of the truth table. Therefore, the distributions of Hybrid2 and Hybrid3 are identical.

In Hybrid3, the ciphertext is randomly generated. Additionally, the decoding information produced by Sim_{priv} is also identically distributed with Hybrid3. Consequently, the distribution of Sim_{priv} aligns with that of Hybrid3. □

E Supplemental Materials for Section 7

E.1 Proof of Theorem 5

Theorem 5. *Any ideally secure garbling scheme for an* AND *gate that satisfies the* MODEL-0 *must have* $|\boldsymbol{G}| \geq 4\kappa$.

Proof. As shown in Table 3, for varying choices of permute bits $a, b \in \{0,1\}$, there exist 10 potential values for $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$ (the size of $\mathcal{S} = \mathcal{S}_{00} \cup \mathcal{S}_{01} \cup \mathcal{S}_{10} \cup \mathcal{S}_{11}$ is 10). Given that these values are randomly chosen, any of the 10 possibilities could be assumed. Therefore, for a fixed $\boldsymbol{M}[t]$, the set

derived by applying different ciphertext $\boldsymbol{G}[t]$ should encompass the entire range of these potential values for $(Z_{00}[t], Z_{01}[t], Z_{10}[t], Z_{11}[t])$. Considering there are $2^{|\boldsymbol{G}[t]|}$ possible values for $\boldsymbol{G}[t]$, it is evident that $2^{|\boldsymbol{G}[t]|} \geq 10$. Since $|\boldsymbol{G}[t]|$ must be an integer, it naturally follows that $|\boldsymbol{G}[t]| \geq 4$. □

E.2 Proof of Lemma 2

Lemma 2. *For ℓ columns of the output labels, the total number of potential values is given by $2^{2\ell+2} - 3 \cdot 2^{\ell}$, where $1 \leq \ell \leq \kappa$.*

Proof. Let $\mathcal{S}$ represent the set of all possible values in Table 3. A subset of $\mathcal{S}$ is $\mathcal{S}_1 = \{(0,0,0,0),\ (1,1,1,1)\}$, and another subset is $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$.

We use induction to prove the lemma. When $\ell = 1$, referring to Theorem 5, the possible values for a single column equal the size of $\mathcal{S}$. This gives $2^{2\cdot 1+2} - 3 \cdot 2^1 = 10 = |\mathcal{S}|$. So, our base case holds.

Assume that for $\ell = n$, the number of possible values for n columns is $2^{2n+2} - 3 \cdot 2^n$. Now, we consider $\ell = n + 1$. We split our discussion into two cases based on the first column:

1. If the first column belongs to $\mathcal{S}_1$, then the number of potential values for the first column is 2. The number of potential values for the next n columns is $2^{2n+2} - 3 \cdot 2^n$.
2. If the first column belongs to $\mathcal{S}_2$, then the number of potential values for the first column is 8. The number of potential values for the next n columns is 4^n.

Taking these two cases together, the total number of possible values for $\ell = n + 1$ columns is:

$$2 \cdot (2^{2n+2} - 3 \cdot 2^n) + 8 \cdot 4^n = 2^{2(n+1)+2} - 3 \cdot 2^{(n+1)}$$

This matches our induction hypothesis and confirms the lemma for all ℓ in the given range. □

E.3 Proof of Theorem 6

Theorem 6. *Any ideally secure garbling scheme for an* AND *gate with n input wires that satisfies the* MODEL-3″ *must have $|\boldsymbol{G}| \geq n\kappa$.*

Proof. In MODEL-3″, we continue to use the column-wise processing approach. Each time, a column is extracted from the base. Given n inputs, a total of n permute bits are required, leading to 2^n possible cases. For the t-th column, represented as $\boldsymbol{M}[t]$ (which consists of 2^n bits), our goal is for $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ to cover all possible cases by varying the ciphertext $\boldsymbol{G}[t]$.

If the length of $\boldsymbol{G}[t]$ is $n-1$, then for any given $\boldsymbol{M}[t]$, the mapping function $\mathsf{MAP}(\boldsymbol{M}[t], \boldsymbol{G}[t])$ can produce at most 2^{n-1} distinct outputs. Considering that there are 2^n possible cases, either $\{0, \ldots, 0\}$ (all zeros) or $\{1, \ldots, 1\}$ (all ones) must be included among these outputs.

This implies that, of the 2^n potential mappings, a minimum of $2^{n-1} + 1$ will result in either $\{0, \ldots, 0\}$ or $\{1, \ldots, 1\}$, restricting other mappings to at most $2^{n-1} - 1$ that can produce other values. Mappings of the former kind will cause the same bit position in the output labels C_0 and C_1 to be identical, whereas mappings of the latter kind will result in differing bits.

Consequently, this leads to a higher probability of the same bit position in C_0 and C_1 being equal than being unequal, thereby violating privacy. Therefore, we must have $\boldsymbol{G}[t] \geq n$. □