Bitwise Garbling Schemes
A Model with $\frac{3}{2}\kappa$-bit Lower Bound of Ciphertexts

Fei Xu[1], Honggang Hu[1,2](✉), and Changhong Xu[1]

[1] School of Cyber Security and Technology, University of Science and Technology of China, Hefei 230027, China
{xf555233,xuchangh}@mail.ustc.edu.cn
[2] Hefei National Laboratory, Hefei 230088, China
hghu2005@ustc.edu.cn

Abstract. At Eurocrypt 2015, Zahur, Rosulek, and Evans proposed the model of Linear Garbling Schemes. This model proved a 2κ-bit lower bound of ciphertexts for a broad class of garbling schemes. Since then, several methods have been developed that bypass this lower bound, albeit with a notable limitation: Their reliance on specifically correlated input wire labels restricts their applicability across most gates. At Crypto 2021, Rosulek and Roy presented the innovative "three-halves" garbling scheme in which AND gates cost $1.5\kappa + 5$ bits and XOR gates are free. A noteworthy aspect of their scheme is the slicing-and-dicing technique, which is applicable universally to all AND gates when garbling a boolean circuit. Following this revelation, Rosulek and Roy presented several open problems. Our research primarily addresses one of them: "*Is 1.5κ bits optimal for garbled AND gates in a more inclusive model than Linear Garbling Schemes?*"

In this paper, we propose the **Bitwise Garbling Schemes**, a model that seamlessly incorporates the slicing-and-dicing technique. Our key revelation is that 1.5κ bits is indeed optimal for arbitrary garbled AND gates in our model. Since Rosulek and Roy also suggested another problem which questions the necessity of free-XOR, we explore constructions without free-XOR and prove a 2κ-bit lower bound. Therefore, sacrificing compatibility with free-XOR does not lead to a more efficient scheme.

Keywords: Garbled circuit · 2PC · Linear garbling scheme

1 Introduction

Since Yao introduced Garbled Circuits (GC) in [29], they have gained significant attention. It is believed that GC are the simplest construction when realizing secure two-party computation (2PC). Up to now, garbled circuit is still the primary technique in the 2PC setting due to their efficiency.

The main reason of their high efficiency is that both parties only use fast symmetric-key operations. Since necessary computation can be finished apace, the actual bottleneck of GC lies in the communication overhead. There are a continual line of works [17,19,20,22,25,26,30] aiming to reduce the length of data required to encrypt individual gates, as this data will subsequently be transmitted from the garbler to the evaluator. We distinguish between additional bits which are used to control the evaluator's behavior and ciphertexts which are directly used to compute the output wire label. In addition, we refer to these two parts collectively as material [18]. For example, the material of an AND gate in the "three-halves" garbling scheme [26] contains three 0.5κ-bit ciphertexts and 5 additional bits.

Zahur, Rosulek, and Evans proposed the half-gates scheme and the model of Linear Garbling Schemes in [30]. As we mentioned above, they proved there exists a 2κ-bit lower bound of ciphertexts in this model. Meanwhile, the half-gates scheme ensures that the communication cost per AND gate is 2κ bits. Hence, this scheme is optimal in this model. However, Rosulek and Roy [26] then proposed the state-of-the-art "three-halves" garbling scheme, which totally breaks this lower bound. The novel slicing technique, in which different halves of the output wire label can be computed via different linear combinations, lies outside this model, introducing more possibilities. Intuitively, since the "three-halves" garbling scheme improves the size of material by slicing the output wire label into halves, a further slicing could potentially yield even better outcomes. For example, we may require only $\frac{4}{3}\kappa$ bits, $\frac{5}{4}\kappa$ bits, or potentially even fewer. Therefore, Rosulek and Roy [26] proposed an open question:

Is 1.5κ bits optimal for garbled AND gates in a more inclusive model than than Linear Garbling Schemes?

We discuss this model and technique later in Sect. 3.

1.1 Our Contributions

We propose a model called **Bitwise Linear Garbling Schemes**, which builds upon the foundation of the traditional Linear Garbling Schemes model. This means that all practical garbling schemes captured by the old model are naturally included in our new model. The primary improvement of our model is our focus on the slicing technique. We consider the most extreme case where the κ-bit wire label can be sliced into κ bits. As a result, in our new model, each bit of the output wire label can be computed via a different linear combination. As we mentioned, this model is *bitwise*. One may argue that this idea is only implied by the slicing technique, short of inclusiveness. However, it is necessary for a garbling scheme to guarantee the security of each bit of the output wire label. Hence, we believe that our bitwise processing is hard to circumvent, which reflects the inclusive nature of our model.

In response to this open question, we consider the garbling of *an arbitrary AND gate*, rather than just a single isolated AND gate. In this case, based on our classification of oracle responses, we prove a 1.5κ-bit lower bound in our model achieved with free-XOR. We believe that this classification is essential in the two-party setting.

Inspired by [9], we also deal with non-linear actions by proposing the model of **Bitwise Garbling Schemes**. This model not only includes the dicing technique completely, but also makes the 1.5κ-bit lower bound more convincing.

Meanwhile, we also discover the importance of free-XOR. It is quite interesting that free-XOR plays a crucial role in both XOR gates and AND gates in practice. When constructions similar to free-XOR (see Sect. 5.1) are forbidden, we can prove a 2κ-bit lower bound. We show that sacrificing compatibility with free-XOR does not provide any advantage in our model, even under the gate-hiding assumption [26].

There are also some methods [3,17,28] that manage to achieve k-bit ciphertext on a single AND gate in isolation. We can analyze them in a similar way. These methods that utilize constructions similar to free-XOR, but do not retain the same correlation on the output wire, remain outside of our model. Their deviation is primarily due to their inapplicability throughout the circuit. If they are prescribed as general methods to match our model, they still follow the above lower bound.

Techniques for proving lower bounds. Our starting point is Bitwise Linear Garbling Schemes, inspired by Linear Garbling Schemes and the slicing technique naturally. In short, each bit of the output wire label, which can be computed via a different linear combination, must be private. It is easy to find that privacy comes from the non-linearity of the queries to the random oracle in Linear Garbling Schemes.

Therefore, we propose a pivotal observation: The queries to the random oracle can be classified. To the best of our knowledge, all known practical garbling schemes align with this observation. For simplicity, let's consider a garbled AND gate which takes wire labels A_i, B_j where $i, j \in \{0,1\}$ as inputs with a random oracle H. We use $E_{i,j}$ to represent the evaluator with wire labels (A_i, B_j). Assuming each input wire label is independently sampled from $\{0,1\}^\kappa$, it is intuitive to exemplify with the following forms: $H(A_i)$, $H(B_j)$ and $H(A_i, B_j)$. What is different is we view $H(A_0)$ as oracle responses which can be computed by $\{E_{0,0}, E_{0,1}\}$, while $\{E_{1,0}, E_{1,1}\}$ can only guess them. Obviously, we can not list all oracle responses, but we can consider all subsets of $\{E_{0,0}, E_{0,1}, E_{1,0}, E_{1,1}\}$. We require that every oracle response be computed by at least one evaluator, and we associate this response with a subset containing corresponding evaluators. In light of the limited number of subsets, we finitely classify oracle responses. For the sake of presentation, we *choose* a common form to represent all oracle responses associated with a subset. For example, in the free-XOR setting, $A_0 \oplus B_1 = A_1 \oplus B_0$, so we choose $H(A_0 \oplus B_1)$ to represent oracle responses associated with $\{E_{0,1}, E_{1,0}\}$ in our discussion and proofs of lower bounds. We **insist** that the random oracle is not necessarily queried in this form.

Furthermore, oracle responses are in charge of ensuring security. Each bit of the output wire label needs a linear combination of all possible oracle responses $(Q_1, Q_2, \ldots, Q_q)$ to keep private, which allows us to build a matrix. Roughly speaking, in order to compute the k-th bit of the output wire label of the evaluator with (A_i, B_j), we allocate a vector to compute the inner product of this vector and $(Q_1, Q_2, \ldots, Q_q)$. We study the rank of this matrix by considering the security, and prove the lower bound. This technique was presented in Linear Garbling Schemes [30].

In [9], Fan, Lu and Zhou viewed the mapping from bases (which are similar to oracle responses) and ciphertexts to the output wire label as a function. The model of Linear Garbling Schemes is included, if we consider a linear function which performs linear combinations of oracle responses and ciphertexts. By considering non-linear functions, they dealt with non-linear mapping. Inspired by their idea, we then propose the Bitwise Garbling Schemes, which also allows non-linear mapping. Note that this model includes the dicing technique.

1.2 Related Works

GC are widely regarded as the most common approach to 2PC in many cases. Moreover, the foundational concept behind GC is pivotal even when the number of involved parties exceeds two [11,23,24]. GC are also employed with respect to different network conditions and application scenarios [2,7,12,13,14], through diverse ideas and techniques. Concurrently, a multitude of studies [8,16,27] have emerged to adapt GC for malicious secure 2PC. A seminal advancement in the GC domain is the introduction of a key framework named **garbling schemes** by Bellare, Hoang, and Rogaway [5]. This framework not only standardized a series of related works but also solidified the description and security properties of GC, making it more convenient to elaborate formally. Additionally, there are two prominent techniques in this area. Many garbling schemes utilize these techniques, optimizing either computational or communicational efficiency.

The *point-and-permute* technique [4] requires the garbler to sample a random *permute bit* per wire. Although each permute bit needs to be secret, the garbler can utilize the XOR operation between this bit and the actual logic value to produce two contrasting *color bits*. Each color bit corresponds to a specific wire label, while only one of them is revealed to the evaluator. In the majority of garbling schemes employing this technique, the evaluator can take advantage of color bits of wire labels to choose the corresponding ciphertexts for all gates, as the garbler has arranged ciphertexts based on color bits by convention. This technique avoids the evaluator's need for multiple attempts to decide on the right ciphertexts, leading to a reduction in computational cost. However, some methods [3,17,26,28] also show that it is possible to circumvent the 2κ bits lower bound if the evaluator's behavior is not totally decided by color bits. In line with this, our model does not impose such a constraint. Note that this technique itself costs 1 bit of each wire label, which technically reduces the security parameter by 1. Nevertheless, this decrement is typically overlooked in general discourse.

The *free-XOR* technique introduced by Kolesnikov and Schneider [20] has been playing an important role in GC acceleration. The garbler chooses a global and secret XOR-difference Δ, and two wire labels of the same wire always keep this difference. In the context of XOR gates, this technique simplifies the operations for both parties involved. They merely have to perform the XOR operation on two input wire labels to determine the output wire label, and the communication cost of XOR gates is reduced to zero. It's also worth noting that this technique mandates a distinct security requirement for hash functions [6,26]. Because output wire labels are also restricted to maintain the XOR-difference $\varDelta$, Rosulek and Roy [26] also proposed another question: *Does it help to sacrifice compatibility with free-XOR?* In this paper, we also answer this question.

2 Preliminaries

2.1 Notations

$x \xleftarrow{\$} X$ means that x is uniformly sampled from the uniform distribution X. The notation $[n]$ denotes the set $\{1, \ldots, n\}$. We use bold symbols to denote vectors, e.g., $\boldsymbol{e}, \boldsymbol{X}, \boldsymbol{Y}$. Calligraphic fonts are used to denote sets, e.g., $\mathcal{E}, \mathcal{Z}$. In the context of garbling schemes, we may also refer to the garbler or evaluator using pronouns he or she. κ denotes the computational security parameter.

We mainly focus on how to garble an arbitrary AND gate g. Two input wires a and b of g are encoded as wire label pairs (A_0, A_1) and (B_0, B_1). Each wire label is uniformly sampled from $\{0,1\}^\kappa$. During GC evaluation, the actual logic value on wire a is denoted as x_a. We use A_0^i to denote the i-th bit of A_0. One wire label in a pair (A_0, A_1) represents the logic value 1 on this wire, while the other represents 0. The evaluator obtains one of them, based on x_a. The output wire c is encoded as wire label pair (C_0, C_1). We denote the concatenation of two wire labels A_i, B_j by $A_i \parallel B_j$. To make the evaluator with two wire labels (A_i, B_j) from two label pairs obtain her corresponding output wire label correctly, the garbler also arranges ciphertexts $G_1, \ldots, G_m$ where m is the number of ciphertexts.

Note that the evaluator only has one element of the set $\{(A_i, B_j) | i, j \in \{0,1\}\}$, while the garbler has to consider all of them. For simplicity, we regard $E_{i,j}$ as the evaluator with (A_i, B_j). This suggests that four distinct types of evaluators coexist simultaneously. When considering the security property, we hope to protect $E_{i,j}$ from each of $\{E_{\bar{i},j}, E_{i,\bar{j}}, E_{\bar{i},\bar{j}}\}$ because an adversary may possess one of them and threat privacy. (We sometimes use $\bar{i}$ instead of $1-i$.)

2.2 Garbling Schemes

We use the definition of garbling schemes from [26].

Definition 1. *A garbling scheme consists of four algorithms as below.*

$(M, \boldsymbol{e}, \boldsymbol{D}) \leftarrow \mathsf{Garble}\ (1^\kappa, f)$*: Output the material M of GC, encoding information $\boldsymbol{e}$ and decoding strings $\boldsymbol{D}$ on parameter 1^κ and the description of the boolean circuit f.*

$\boldsymbol{X} := \mathsf{Encode}(\boldsymbol{e}, \boldsymbol{x})$*: Transform the cleartext input $\boldsymbol{x}$ to the garbled input $\boldsymbol{X}$ with encoding information $\boldsymbol{e}$.*

$\boldsymbol{Y} := \mathsf{Eval}(M, \boldsymbol{X})$*: On the input $(M, \boldsymbol{X})$, evaluate the garbled output $\boldsymbol{Y}$.*

$\boldsymbol{y} := \mathsf{Decode}(\boldsymbol{D}, \boldsymbol{Y})$*: Transform the garbled output $\boldsymbol{Y}$ to the cleartext output $\boldsymbol{y}$ with decoding strings $\boldsymbol{D}$.*

A garbling scheme satisfies the following security properties.

Correctness: *After getting* $(M, \boldsymbol{e}, \boldsymbol{D}) \leftarrow \mathsf{Garble}\ (1^\kappa, f)$ *for the boolean circuit* f *and cleartext input* $\boldsymbol{x}$, $\mathsf{Decode}(\boldsymbol{D}, \mathsf{Eval}(M, \mathsf{Encode}(\boldsymbol{e}, \boldsymbol{x}))) = f(x)$ *fails with negligible probability.*

Privacy: *The output of a simulator with input* $(1^\kappa, f, \boldsymbol{y})$ *is indistinguishable from* $(M, \boldsymbol{X}, \boldsymbol{D})$ *generated in the usual way. This means that* $(M, \boldsymbol{X}, \boldsymbol{D})$ *should not reveal any information about* $\boldsymbol{x}$ *except* $\boldsymbol{y} = f(\boldsymbol{x})$.

Obliviousness: *The output of a simulator with input* $(1^\kappa, f)$ *is indistinguishable from* $(M, \boldsymbol{X})$ *generated in the usual way. This means that* $(M, \boldsymbol{X})$ *should not reveal any information about* $\boldsymbol{x}$ *since decoding information is unknown.*

Authenticity: *Given the collection* $(M, \boldsymbol{X}, \boldsymbol{D})$, *the probability of producing* $\boldsymbol{Y}' \neq \mathsf{Eval}(M, \boldsymbol{X})$ *such that* $\mathsf{Decode}(d, \boldsymbol{Y}') \neq \perp$ *is negligible. In other words, no PPT adversary* $\mathcal{A}$ *can somehow produce a garbled output which can be decoded as a cleartext output different from* $\boldsymbol{y}$ *with non-negligible probability.*

3 Technical Overview: Garbling Schemes

In this section, we review the Linear Garbling Schemes model and the slicing-and-dicing technique in the "three-halves" garbling scheme. We offer a more detailed review of the old model, as our novel model builds upon it. While the model of Linear Garbling Schemes includes all known practical garbling schemes at that time, several works [3,17,26,28] pointed out its shortcomings. Such insights paved the way for the development of a new model.

3.1 Linear Garbling Schemes

In the Linear Garbling Schemes model, parties are viewed as computationally unbounded entities which can make polynomially many queries to a random oracle. This standard setting about Minicrypt is also a fitting description of practical garbling schemes. We follow the concept of **ideal security** in this model, which requires that no adversary has advantage better than $\text{poly}(\kappa)/2^\kappa$. [3] Readers are referred to this model in [30]. When garbling an AND gate, this model is as follows:

Garble: This algorithm is parameterized by integers m, r, q and vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{C}_{a,b,0} | a, b \in \{0,1\}\}$, $\{\boldsymbol{C}_{a,b,1} | a, b \in \{0,1\}\}$, and $\{\boldsymbol{G}^{(i)}_{a,b} | a, b \in \{0,1\}\}$. Each vector has length of $r + q$, and consists of entries in $GF(2^\kappa)$.

1. For $i \in [r]$, choose $R_i \xleftarrow{\$} GF(2^\kappa)$.
2. Make q distinct queries to the random oracle (which can be chosen as a deterministic function of the R_i values) and get responses $Q_1, \ldots, Q_q$. We place these values on which the algorithm can act linearly in $\boldsymbol{S} = (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$.

[3] Clearly, a garbling scheme on security parameter $\kappa - 1$ also provides security $\text{poly}(\kappa)/2^\kappa$. However, we consider the concrete parameter κ. In other words, we do not allow to degrade the security parameter.

3. Choose two permute bits $a, b \xleftarrow{\$} \{0,1\}$ for two input wires.
4. For $i \in \{0,1\}$, compute $A_i = \langle \boldsymbol{A}_i, \boldsymbol{S} \rangle$, $B_i = \langle \boldsymbol{B}_i, \boldsymbol{S} \rangle$ and $C_i = \langle \boldsymbol{C}_{a,b,i}, \boldsymbol{S} \rangle$. Then two input wire labels are $(A_0 \parallel 0, A_1 \parallel 1)$. As we state above, these subscripts denote the public color bits. A_a and B_b correspond to FALSE. Let C_0 correspond to FALSE.
5. For $i \in [m]$, compute $G_i = \langle \boldsymbol{G}^{(i)}_{a,b}, \boldsymbol{S} \rangle$. These values comprise the garbled circuit.

Encode: On input $x_a, x_b \in \{0,1\}$, set color bits $\alpha := x_a \oplus a$ and $\beta := x_b \oplus b$. The evaluator gets $A_\alpha \parallel \alpha$ and $B_\beta \parallel \beta$.
Eval: Parameterized by q' and vectors $\{\boldsymbol{V}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ of length $q' + m + 2$.

1. The evaluator has wire labels $A_\alpha \parallel \alpha$, $B_\beta \parallel \beta$, and ciphertexts $G_1, \ldots, G_m$.
2. Make q distinct queries to the random oracle and get responses $Q'_1, \ldots, Q'_{q'}$. We also place these values on which the algorithm can act linearly in $\boldsymbol{T} = (A_\alpha, B_\beta, Q'_1, \ldots, Q'_{q'}, G_1, \ldots, G_m)$.
3. Output the inner product $\langle \boldsymbol{V}_{\alpha,\beta}, \boldsymbol{T} \rangle$.

To ensure the correctness, the equation $C_{(a\oplus\alpha)\wedge(b\oplus\beta)} = \langle \boldsymbol{V}_{\alpha,\beta}, \boldsymbol{T} \rangle$ must hold. $\boldsymbol{T}$ is divided into a *public* part and a *private* part. $\boldsymbol{T}^{pub}$ consists of wire labels and responses. Note that $\{Q'_1, \ldots, Q'_{q'}\}$ must be a subset of $\{Q_1, \ldots, Q_q\}$, since the garbler has to be able to anticipate it. Hence, $\boldsymbol{T}^{pub}$ is a linear function of $\boldsymbol{S}$ which only depends on α, β. We denote it by $\boldsymbol{T}^{pub} = \mathbb{M}_{\alpha,\beta} \times \boldsymbol{S}^\top$. Similarly, $\boldsymbol{T}^{prv}$ which consists of ciphertexts is also a linear function of $\boldsymbol{S}$ which only depends on a, b. Assume a matrix $\mathbb{G}_{a,b}$ whose rows are $\boldsymbol{G}^{(1)}_{a,b}, \ldots, \boldsymbol{G}^{(m)}_{a,b}$, we denote it by $\boldsymbol{T}^{prv} = \mathbb{G}_{a,b} \times \boldsymbol{S}^\top$.

Then we divide $\boldsymbol{V}_{\alpha,\beta}$ similarly, and get the following condition:

$$\begin{aligned}\left\langle \boldsymbol{C}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}, \boldsymbol{S} \right\rangle &= \left\langle \boldsymbol{V}^{pub}_{\alpha,\beta}, \boldsymbol{T}^{pub} \right\rangle + \left\langle \boldsymbol{V}^{prv}_{\alpha,\beta}, \boldsymbol{T}^{prv} \right\rangle \\ &= \left\langle \boldsymbol{V}^{pub}_{\alpha,\beta}, \mathbb{M}_{\alpha,\beta} \times \boldsymbol{S}^\top \right\rangle + \left\langle \boldsymbol{V}^{prv}_{\alpha,\beta}, \mathbb{G}_{a,b} \times \boldsymbol{S}^\top \right\rangle \\ &= \langle \boldsymbol{Z}_{\alpha,\beta}, \boldsymbol{S} \rangle + \left\langle \boldsymbol{V}^{prv}_{\alpha,\beta} \times \mathbb{G}_{a,b}, \boldsymbol{S} \right\rangle,\end{aligned}$$

where $\boldsymbol{Z}_{\alpha,\beta} = \boldsymbol{V}^{pub}_{\alpha,\beta} \times \mathbb{M}_{\alpha,\beta}$ is a vector depending on α, β.

The vector $\boldsymbol{S}$ is uniformly distributed. Hence, the following equation must hold:

$$\boldsymbol{C}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)} = \boldsymbol{Z}_{\alpha,\beta} + \boldsymbol{V}^{prv}_{\alpha,\beta} \times \mathbb{G}_{a,b}. \tag{1}$$

Zahur, Rosulek, and Evans proved three pivotal claims:

– *Claim* 1: Matrices $\{\mathbb{G}_{a,b} | a, b \in \{0,1\}\}$ are all distinct.
– *Claim* 2: Vectors $\{\boldsymbol{Z}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ are pairwise linearly independent.
– *Claim* 3: Vectors $\{\boldsymbol{V}^{prv}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ are pairwise linearly independent.

In our opinion, *Claim* 2 is crucial. So we give their proof of *Claim* 2. Suppose that it is violated by $\boldsymbol{Z}_{0,1} = \sigma \boldsymbol{Z}_{0,0}$, where σ is a scalar. Then $E_{0,0}$ can also compute $\langle \boldsymbol{V}_{0,1}, \boldsymbol{T} \rangle = \sigma \langle \boldsymbol{V}_{0,0}^{pub}, \boldsymbol{T}^{pub} \rangle + \langle \boldsymbol{V}_{0,1}^{prv}, \boldsymbol{T}^{prv} \rangle$. Therefore, $E_{0,0}$ has output wire labels for two different cases. This is not allowed when garbling an AND gate.

To prove the lower bound, we get four equations by considering $(\alpha, \beta) \in \{(0,0),(0,1)\}$ and $(a,b) \in \{(0,0),(0,1)\}$ for Equation (1). By combining these equations appropriately, we get:

$$(\boldsymbol{V}_{0,1}^{prv} - \boldsymbol{V}_{0,0}^{prv}) \times (\mathbb{G}_{0,1} - \mathbb{G}_{0,0}) = \mathbf{0}.$$

Based on *Claim* 1 and *Claim* 3, we can find that $\boldsymbol{V}_{0,1}^{prv} - \boldsymbol{V}_{0,0}^{prv}$ is a nonzero vector and $\mathbb{G}_{0,1} - \mathbb{G}_{0,0}$ is a nonzero matrix. So $\mathbb{G}_{0,1} - \mathbb{G}_{0,0}$ must have at least 2 rows. This implies that $\mathbb{G}_{a,b}$ has at least 2 rows, resulting in ciphertexts that are at least 2κ bits in length.

3.2 Slicing-and-Dicing

The "three-halves" garbling scheme [26] uses the slicing-and-dicing technique to beat the old lower bound. In the Linear Garbling Schemes model, $\{\boldsymbol{V}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ are fixed. However, the dicing technique enables the garbler to send additional bits apart from 1.5κ-bit ciphertexts. These bits are generated by encrypting *control bits*. In this scheme, control bits are used to determine how to combine different parts of input wire labels to compute the output wire label, i.e., $\boldsymbol{V}_{\alpha,\beta}$. Moreover, these control bits are sampled by a randomized algorithm, ensuring that the evaluator learns nothing from them. This idea, which first appeared in [17], is outside of the old model. While our first model does not provide complete coverage of this idea, we choose a trivial approach where the evaluator is assumed to know how to compute her output wire label, allowing us to overlook these bits. Our second model, in which actions of the evaluator can be decided by values in $\boldsymbol{S}$ and ciphertexts, includes the dicing technique.

Our major concern is the slicing technique which enables the evaluator to exploit more linear combinations. As noted by Rosulek and Roy [26], it increases the linear-algebraic dimension in which the scheme operates. An intuitive difference between this scheme and previous schemes is that this scheme can operate on a 4×2 sub-construction (see Table 1). To explain how this technique works, we examine the half-gates scheme in a linear-algebraic perspective:

$$\begin{bmatrix} 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 1&1&1 \end{bmatrix} \begin{bmatrix} C \\ G_0 \\ G_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1&0&1&0 \\ 1&0&0&1 \\ 0&1&1&0 \\ 0&1&0&1 \end{bmatrix}}_{\mathbb{M}_H} \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix} \oplus \begin{bmatrix} 0&0 \\ 1&0 \\ 0&0 \\ 1&1 \end{bmatrix} \begin{bmatrix} A_0 \\ \Delta \end{bmatrix} \oplus \underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{t}} \Delta.$$

The main reason for the 2κ-bit ciphertexts in the half-gates scheme is that the rank of the matrix $\mathbb{M}_H$ is 3. (These hash outputs collectively are regarded as the oracle responses.) By setting the output wire label C as $H(A_0) \oplus H(B_0)$, we only need two κ-bit ciphertexts to solve the mismatches between different rows.

Note that the fourth row can be obtained by XORing the top three rows, so it is free in terms of ciphertexts.

If we slice the output wire label C into κ bits, we can approximate this garbling as iterating a 4×1 sub-construction κ times: A 4×1 sub-construction is used to compute one bit of the output wire label, e.g., $C^1 = H(A_0)^1 \oplus H(B_0)^1$. To compute each bit of the output wire label, both parties need to combine wire labels, 1-bit oracle responses and ciphertexts linearly. This is how we include half-gates when the output wire label is sliced.

Table 1. The oracle responses used in different halves of the output wire label. These oracle responses are of length $\kappa/2$ and free-XOR technique is used.

Input wire labels	Oracle responses	
	Left half	Right half
(A_0, B_0)	$H(A_0) \oplus H(A_0 \oplus B_0)$	$H(B_0) \oplus H(A_0 \oplus B_0)$
(A_0, B_1)	$H(A_0) \oplus H(A_0 \oplus B_1)$	$H(B_1) \oplus H(A_0 \oplus B_1)$
(A_1, B_0)	$H(A_1) \oplus H(A_0 \oplus B_1)$	$H(B_0) \oplus H(A_0 \oplus B_1)$
(A_1, B_1)	$H(A_1) \oplus H(A_0 \oplus B_0)$	$H(B_1) \oplus H(A_0 \oplus B_0)$

We now consider the "three-halves" garbling scheme with a focus on the oracle responses as presented in Table 1. One can easily check that each half follows the above half-gates construction. Therefore, both halves need two 0.5κ-bit ciphertexts. It still does not provide any improvement since 2κ bits are needed.

However, from a linear-algebraic perspective, we can formulate a matrix of rank 5 to multiply the vector of these oracle responses. Specifically, let us consider $C^L_{i,j}$ and $C^R_{i,j}$ as the oracle responses for the left and right half of $C_{i,j}$. The formulation is as follows:

$$\begin{bmatrix} C^L_{0,0} \\ C^R_{0,0} \\ C^L_{0,1} \\ C^R_{0,1} \\ C^L_{1,0} \\ C^R_{1,0} \\ C^L_{1,1} \\ C^R_{1,1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,1\,0 \end{bmatrix}}_{\mathbb{M}'_H} \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix}.$$

The half-gates construction points out that each half needs two 0.5κ-bit ciphertexts after combining the halves of input wire labels. However, we find that it is possible to use three 0.5κ-bit ciphertexts by choosing these halves skillfully, since the rank of $\mathbb{M}'_H$ is 5. That is to say, one of the ciphertexts on the left is the same as the one on the right.

Simply using two different linear combinations for the evaluator does not directly lead to a better outcome. However, the increasement in linear-algebraic dimension (namely, $H(A_i \oplus B_j)$) creates an opportunity to reduce repeated ciphertexts. This aspect is pivotal when constructing a new model.

3.3 Intuition

We argue that it is the slicing instead of dicing technique that plays a crucial role in beating the old lower bound. Actually, this argument has been reflected in the "three-halves" garbling scheme [26]. Initially, the slicing technique is essential for creating the possibility to save a ciphertext. However, if the evaluator uses her color bits to directly compute the output wire label (involving input wire label halves, oracle responses and ciphertexts), the truth table supported by this fixed linear combination is not sufficient. Hence, considering the security aspect, it is necessary that the evaluator's actions remain unfixed even when color bits are provided. This is where the dicing technique comes into effect.

This leads us to an intuitive idea of our first model: we take into account of all possibilities introduced by the slicing technique, and sideline the dicing technique. We can achieve this by assuming that the evaluator already knows the linear operations she is expected to learn. The consideration of the dicing technique is left to the second model. Obviously, the most extreme case caused by the slicing technique is that every bit of the output wire label can be computed by a different linear combination. Moreover, maximizing the linear-algebraic dimension in which a scheme can operate, this idea is convenient for us to consider the security of all bits. Note that we consider 1-bit oracle responses and ciphertexts in our models.

3.4 Key Observation

To analyze the lower bound of our intuition, we need a key observation. Till now, the 4κ linear combinations caused by our intuition are too complicated. Note that the Linear Garbling Schemes model simply lists q responses $Q_1, \ldots, Q_q$ and the evaluator is assumed to obtain a subset of it. This gives the evaluator capability beyond those available in a garbling scheme. For instance, the evaluator with (A_i, B_j) should not have access to $H(A_{1-i})$.

To make our model and its lower bound more accurate, we present an observation: the oracle responses in our model can be classified (see Definition 2). As far as we know, all previous garbling schemes follow this observation. When the free-XOR technique is not allowed, we usually make sure each wire label is sampled independently. (Those methods [3,17,28] mentioned earlier, which break the 2κ-bit lower bound, do not satisfy this situation.) We consider forms of oracle responses, such as $H(A_i)$, $H(B_j)$ and $H(A_i \parallel B_j)$. On oracle responses of the form $H(A_i)$, we require that the evaluator knowing A_i compute them. Meanwhile, the probability of correctly guessing $H(A_i)$ follows ideal security with respect to κ. Using only a portion of an input wire label degrades the security parameter. When free-XOR is enabled, we immediately notice a new oracle response form: $H(A_i \oplus B_j)$. The evaluator with (A_{1-i}, B_{1-j}) can also learn this response, because $A_0 \oplus A_1 = B_0 \oplus B_1 = \Delta$. This type of oracle response plays a crucial role in the "three-halves" garbling scheme and our model, even if only AND gates are involved.

One may argue that we can not guarantee all oracle responses are considered, since we can not list all possible forms. In our opinion, oracle responses must be computed by at least one evaluator, while other evaluators who guess about them can not break the ideal security. Hence, our classification of oracle responses is actually based on all possible **sets of evaluators**.

Observation: For an arbitrary garbled AND gate with input wire labels (A_0, A_1) and (B_0, B_1), define the function $l : \{A_0, A_1\} \times \{B_0, B_1\} \to \{0,1\}^*$. For (A_i, B_j) where $i, j \in \{0,1\}$, $l(A_i, B_j)$ generates a bit string of length not less than κ, ensuring at least κ bits of entropy. Then, we propose Definition 2 for representative form of a type of oracle response.

Definition 2. *For (A_i, B_j) where $i, j \in \{0,1\}$, and oracle responses of the form $H(l(A_i, B_j))$, if we can construct a set $\mathcal{E}_{l(A_i,B_j)}$ such that:*

1. *For any evaluator in the set $\mathcal{E}_{l(A_i,B_j)}$, she obtains $H(l(A_i, B_j))$ with probability 1;*
2. *Any adversary $\mathcal{A}$ that makes polynomially many queries to the random oracle and even possesses $E_{i',j'}$ outside of $\mathcal{E}_{l(A_i,B_j)}$ cannot learn $H(l(A_i, B_j))$ with an advantage better than poly$(\kappa)/2^\kappa$;*

then we regard $H(l(A_i, B_j))$ as a representative form of oracle response for the set $\mathcal{E}_{l(A_i,B_j)}$. In short, $H(l(A_i, B_j))$ is associated with $\mathcal{E}_{l(A_i,B_j)}$.

Since we choose $H(l(A_i, B_j))$ as a representative form, we represent all 1-bit oracle responses associated with $\mathcal{E}_{l(A_i,B_j)}$ as $H(l(A_i, B_j))_1, H(l(A_i, B_j))_2, \ldots$

Note that there are only four evaluators in $\{E_{i,j} | i, j \in \{0,1\}\}$, so $\mathcal{E}_{l(A_i,B_j)}$ containing these evaluators are also finite. Concretely, there are only $2^4 = 16$ possible constructions of this set. For example, let $\mathcal{E}_{l(A_i,B_j)} = \{E_{0,0}\}$. Clearly, $E_{0,1}$ has A_0 and $E_{1,0}$ has B_0. To ensure that $E_{0,1}$ and $E_{1,0}$ fail to get oracle responses, $E_{0,0}$ uses A_0 and B_0 to query the random oracle. There are numerous available forms, such as $H(A_0, B_0)_1$, $H'(A_0 + B_0)_1$ and $H(A_0, B_0, \nu)_2$ where ν is a gate-specific nonce. Nevertheless, we only concern whether they can be computed by evaluators in (or outside of) this set. Hence, we choose $H(A_0, B_0)$ to represent all oracle responses associated with $\{E_{0,0}\}$. Note that we do not require that the random oracle must be queried in this form. In short, we say $H(A_0, B_0)$ is associated with $\{E_{0,0}\}$.

Similarly, we choose a form $H(A_i, B_j)$ associated with $\{E_{i,j}\}$. A form $H(A_i)$ (resp. $H(B_j)$) is associated with $\{E_{i,j}, E_{i,\bar{j}}\}$ (resp. $\{E_{i,j}, E_{\bar{i},j}\}$). The free-XOR technique finds a set $\mathcal{E}_{A_i \oplus B_j} = \{E_{i,j}, E_{\bar{i},\bar{j}}\}$ for the form $H(A_i \oplus B_j)$. If we only rule out the empty set $\emptyset$ and trivial $\mathcal{E} = \{E_{0,0}, E_{0,1}, E_{1,0}, E_{1,1}\}$, those sets containing three elements are still out of consideration. Without loss of generality, suppose there is a set $\mathcal{E}_{l(A_i,B_j)} = \{E_{0,0}, E_{0,1}, E_{1,0}\}$. We need to ensure that both $E_{0,0}$ and $E_{0,1}$ obtain $H(l(A_i, B_j))$, while B_0 and B_1 remain independent from their perspective. This implies that $l(A_i, B_j) = l'(A_0)$, which $E_{1,0}$ can only make a guess about. Thus, constructing such a set is impossible. Since all possible sets have been taken into account, we realize that oracle responses are finitely classified.

When free-XOR is enabled, we choose these representative forms: $H(A_i)$, $H(B_j)$, $H(A_i \oplus B_j)$ and $H(A_i, B_j)$ respectively associated with $\{E_{i,j}, E_{i,\bar{j}}\}$, $\{E_{i,j}, E_{\bar{i},j}\}$, $\{E_{i,j}, E_{\bar{i},\bar{j}}\}$ and $\{E_{i,j}\}$.

Using one input wire label to query the random oracle provides sets of two elements, while we hope that $E_{i,j}$ keeps her output wire label private. In terms of security, only $H(A_i)$ can not be accepted, since $E_{i,\bar{j}}$ can also compute it. Thus, set intersection is a viable and favored approach, which inspires a new claim in Sect. 4.2. For example, we enforce $E_{0,0}$ to use $H(A_0) \oplus H(B_0)$ because it can only be computed by

$$\{E_{0,0}, E_{0,1}\} \cap \{E_{0,0}, E_{1,0}\} = \{E_{0,0}\}.$$

4 Two New Models and Lower Bounds

In this section, we introduce new models of Bitwise Linear Garbling Schemes and Bitwise Garbling Schemes. As our main focus, we consider garbled AND gates with free-XOR in this part. From now on, we keep three positive integers q, t, u. Each type of oracle response is a vector of q different responses, e.g., the form $H(A_0)$ is a vector containing q oracle responses: $(H(A_0)_1, H(A_0)_2, \ldots, H(A_0)_q)$. The garbler has t types of oracle responses, while the evaluator has u types based on her input wire labels. $\mathbf{0}$ denotes the zero vector of length q.

4.1 The First Model: Bitwise Linear Garbling Schemes

As described in Sect. 3.4, this model is based on the old model, the slicing technique and our observation. We define this model still by presenting three procedures.

Garble: This algorithm is parameterized by integers m, r, q, t and vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$. Each vector has length r, with entries in $GF(2^\kappa)$. Meanwhile, vectors $\{\boldsymbol{C}^j_{a,b,0} | a, b \in \{0,1\}, j \in [\kappa]\}$, $\{\boldsymbol{C}^j_{a,b,1} | a, b \in \{0,1\}, j \in [\kappa]\}$, and $\{\boldsymbol{G}^{(i)}_{a,b} | a, b \in \{0,1\}\}$ are all of length $r + tq$, with entries in $GF(2^\kappa)$.

1. For $i \in [r]$, choose $R_i \xleftarrow{\$} GF(2^\kappa)$ to get $\boldsymbol{R} = \{R_1, \ldots, R_r\}$.
2. For $i \in \{0,1\}$, compute $A_i = \langle \boldsymbol{A}_i, \boldsymbol{R} \rangle$, $B_i = \langle \boldsymbol{B}_i, \boldsymbol{R} \rangle$.
3. Choose two permute bits $a, b \xleftarrow{\$} \{0,1\}$ for two input wires.
4. For t types of oracle responses, make tq distinct queries to the random oracle and get tq bits $Q^i_1, \ldots, Q^i_q$, $i \in [t]$. We place these values on which the algorithm can act linearly in $\boldsymbol{S} = (R_1, \ldots, R_r, Q^1_1, \ldots, Q^t_q)$.
5. We compute $C^j_i = \langle \boldsymbol{C}^j_{a,b,i}, \boldsymbol{S} \rangle_\kappa$ where $i \in \{0,1\}, j \in [\kappa]$. Let C_0 (comprising $C^1_0, \ldots, C^\kappa_0$) correspond to FALSE. [4]

[4] Note that we use 1-bit responses of the form Q^j_i where $i \in [q], j \in [t]$ and $R_i \in GF(2^\kappa)$ to compute one bit of the output wire label. Hence, we use $\langle \cdot, \cdot \rangle_\kappa$ instead of $\langle \cdot, \cdot \rangle$. The realization of $\langle \cdot, \cdot \rangle_\kappa$ depending on actual schemes is omitted.

6. For $i \in [m]$, compute $G_i = \langle \boldsymbol{G}^{(i)}_{a,b}, \boldsymbol{S} \rangle_\kappa$. These values are ciphertexts of the garbled circuit.

Encode: On input $x_a, x_b \in \{0,1\}$, set color bits $\alpha := x_a \oplus a$ and $\beta := x_b \oplus b$. The evaluator gets $A_\alpha \parallel \alpha$ and $B_\beta \parallel \beta$.
Eval: Parameterized by m, q, u and vectors $\{\boldsymbol{V}^i_{\alpha,\beta} | \alpha, \beta \in \{0,1\}, i \in [\kappa]\}$ of length $uq + m + 2$.

1. The evaluator has input wire labels $A_\alpha \parallel \alpha$, $B_\beta \parallel \beta$, and ciphertexts $G_1, \ldots, G_m$.
2. Define a function $f : [u] \to [t]$. For u types of oracle responses, make uq distinct queries to the random oracle and get responses $Q^{f(j)}_1, \ldots, Q^{f(j)}_q$, where $j \in [u], f(j) \in [t]$. As we mention above, these oracle responses construct a subset of $\{Q^i_1, \ldots, Q^i_q | i \in [t]\}$. Therefore, $\{f(1), \ldots, f(u)\} \subset [t]$. In fact, $f(j)$ depends on input wire labels, but we neglect them for simplicity. Therefore, we get these values $\boldsymbol{T} = (A_\alpha, B_\beta, Q^{f(1)}_1, \ldots, Q^{f(u)}_q, G_1, \ldots, G_m)$ on which the algorithm can act linearly.
3. Output the inner product $\langle \boldsymbol{V}^i_{\alpha,\beta}, \boldsymbol{T} \rangle_\kappa$, $i \in [\kappa]$.

Because all oracle responses computed by (A_0, A_1, B_0, B_1) construct subsets of $\{Q^1_1, \ldots, Q^t_q\}$, we argue that $\{Q^j_i | i \in [q], j \in [t]\}$ are obtained by using (A_0, A_1, B_0, B_1) to make queries. Therefore, compared to the old model in Sect. 3.1, we no longer use oracle responses in $\boldsymbol{S}$ to compute input wire labels $A_\alpha \parallel \alpha$ and $B_\beta \parallel \beta$. Moreover, considering practical schemes, we enforce the same correlation of wire labels, e.g., the same XOR-difference in free-XOR.

4.2 New Claim

Now we consider different parts of some vectors. We maintain the division of $\boldsymbol{V}^i_{\alpha,\beta}$ and $\boldsymbol{T}$ into public parts and private parts, and get the following equations:

$$\boldsymbol{C}^i_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)} = \boldsymbol{Z}^i_{\alpha,\beta} + \boldsymbol{V}^{prv,i}_{\alpha,\beta} \times \mathbb{G}_{a,b}, i \in [\kappa].$$

When the context is clear, we use $\boldsymbol{C}^i_{\alpha,\beta}$ to represent $\boldsymbol{C}^i_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}$ for simplicity. We can find that some entries in these vectors are used to multiply wire labels in $GF(2^\kappa)$, while others are used to multiply oracle responses in $\{0,1\}$. We divide each of these vectors into a wire label part and an oracle response part. Considering oracle responses, we need to ensure:

$$\boldsymbol{C}^{res,i}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)} = \boldsymbol{Z}^{res,i}_{\alpha,\beta} + \boldsymbol{V}^{prv,i}_{\alpha,\beta} \times \mathbb{G}^{res,i}_{a,b}, i \in [\kappa].$$

The superscript res denotes the part which corresponds to oracle responses. We also get the oracle response part of $\boldsymbol{S}$, i.e., $\boldsymbol{S}^{res} = (Q^1_1, \ldots, Q^t_q)$. Entries of these vectors are in $\{0,1\}$. By this means, we can make use of our observation.

Let us consider $\boldsymbol{Z}^{res,i}_{\alpha,\beta}$ more carefully. Actually, $\boldsymbol{Z}^{res,i}_{\alpha,\beta}$ represents how the evaluator with (A_α, B_β) acts on her oracle responses linearly when she computes the i-th bit of the output wire label. All possible sets of uq oracle responses in

Eval are subsets of the set of tq oracle responses in Garble. We can measure all of them by vectors of length tq in which every q entries corresponds to a type of oracle response. Hence, we can view these vectors as the concatenation of t vectors of length q.

As an example, we represent a type of oracle response $H(A_0)$ as follows:

$$\boldsymbol{Q}^1 = (H(A_0)_1, \ldots, H(A_0)_q).$$

We define $\boldsymbol{Z}^i_{\alpha,\beta,1} \in \{0,1\}^q$ to represent the actions on $\boldsymbol{Q}^1$. Concretely, $E_{\alpha,\beta}$ computes $\langle \boldsymbol{Z}^i_{\alpha,\beta,1}, \boldsymbol{Q}^1 \rangle$ when computing the i-th bit of the output label.

Note that $H(A_0)$ is associated with $\{E_{0,0}, E_{0,1}\}$. If the evaluator $E_{\alpha,\beta}$ is outside of this set, she sets $\boldsymbol{Z}^i_{\alpha,\beta,1}$ to $\mathbf{0}$ since she has no access to this type of response. To simplify notation, we define $\boldsymbol{Z}^{res,i}_{\alpha,\beta} = (\boldsymbol{Z}^i_{\alpha,\beta,1}, \ldots, \boldsymbol{Z}^i_{\alpha,\beta,t})$ be a vector of length tq, where for $j \in [t]$, each component $\boldsymbol{Z}^i_{\alpha,\beta,j}$ is a vector of length q corresponding to a type of oracle response.

Claim 2 from the old model restricts the set $\{\boldsymbol{Z}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$. One may be curious whether the set $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}, i \in [\kappa]\}$ is limited by any condition. After defining two necessary functions, we propose a new claim.

As described in Sect. 3.4, all oracle responses satisfy one of these forms: $H(A_i)$, $H(B_j)$, $H(A_i \oplus B_j)$, and $H(A_i, B_j)$. We rule out $H(A_i, B_j)$ in this situation, and prove its appropriateness later in Sect. 4.3. Then, we get $t = 6$ and arrange

$$\boldsymbol{Z}^{res,i}_{\alpha,\beta} = (\boldsymbol{Z}^i_{\alpha,\beta,1}, \ldots, \boldsymbol{Z}^i_{\alpha,\beta,6}),$$

in which components correspond to forms $H(A_0), H(A_1), H(B_0), H(B_1), H(A_0 \oplus B_0), H(A_0 \oplus B_1)$.

We define a sign function, denoted as $v : \mathbb{Z}_2^q \to \mathbb{Z}_2$, as follows:

$$v(\boldsymbol{V}) = \begin{cases} 0, & \text{if } \boldsymbol{V} = \mathbf{0}; \\ 1, & \text{otherwise.} \end{cases}$$

Briefly speaking, this function is used to indicate whether a vector of length q is a zero vector. Let $\boldsymbol{Z}^{res,i}_{\alpha,\beta} = (\boldsymbol{Z}^i_{\alpha,\beta,1}, \ldots, \boldsymbol{Z}^i_{\alpha,\beta,6})$, we define another function $sum : \mathbb{Z}_2^{6q} \to \mathbb{Z}$ as follows:

$$sum(\boldsymbol{Z}^{res,i}_{\alpha,\beta}) = \sum_{j=1}^{6} v(\boldsymbol{Z}^i_{\alpha,\beta,j}).$$

It is easy to find that sum is used to indicate how many types of oracle responses are used. On the basis of this function, we propose *Claim* 4 as follows:

- *Claim* 4: Given a pair (α, β), any vector $\boldsymbol{L}$ constructed by a non-trivial linear combination of vectors in $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta} | i \in [\kappa]\}$ satisfies $sum(\boldsymbol{L}) \geq 2$.

Proof. Without loss of generality, let's assume that the evaluator with (A_0, B_0) only uses an oracle response $H(A_0)_i$ to compute the i-th bit of the output wire label. Hence, $sum(\boldsymbol{Z}_{0,0}^{res,i}) = 1$, because $\langle \boldsymbol{Z}_{0,0}^{res,i}, \boldsymbol{S}^{res} \rangle_\kappa = H(A_0)_i$. Suppose there exist κ coefficients $y_1, y_2, \ldots, y_\kappa$ such that for all $j \in [\kappa]$, $y_j \in \{0,1\}$. $E_{0,0}$ uses $\bigoplus_{j=1}^{\kappa} y_j B_0^j$ to compute the i-th bit of her output wire label. Certainly, we require that $\sum_{j=0}^{\kappa-1} y_j > 0$. Otherwise, for all $j \in [\kappa]$, $y_j = 0$. Then, $E_{0,1}$ can get this output bit as well.

We can see that there is a 50% chance that the output wire label of $E_{0,0}$ $(C_{0,0})$ is equal to that of $E_{0,1}$ $(C_{0,1})$. However, this does not mean that $C_{0,0}^i = C_{0,1}^i$ holds with a 50% chance. Even if $C_{0,0}$ and $C_{0,1}$ encode different logic values, the i-th bits of them are still independently distributed. Hence, $C_{0,0}^i = C_{0,1}^i$ holds with a 75% chance. $E_{0,1}$ can get her own output bit $C_{0,1}^i$, and get correct $C_{0,0}^i$ with a 75% chance. Note that $E_{0,0}$ linearly acts on $H(A_0)_i$, ciphertexts and some bits of A_0, B_0 to get $C_{0,0}^i$. Since B_0 is uniformly sampled from $\{0,1\}^\kappa$, $\bigoplus_{j=0}^{\kappa-1} y_j B_0^j$ should be uniformly distributed on $\{0,1\}$. By using ciphertexts and the knowledge about A_0, $E_{0,1}$ has a 75% chance to get $\bigoplus_{j=0}^{\kappa-1} y_j B_0^j$. Therefore, it is insecure because she should only have a 50% chance. $\square$

$E_{\alpha,\beta}$ has three types of oracle responses: $H(A_\alpha)$, $H(B_\beta)$ and $H(A_\alpha \oplus B_\beta)$. Vectors which correspond to $H(A_{1-\alpha})$, $H(B_{1-\beta})$ and $H(A_\alpha \oplus B_{1-\beta})$ are all zero. $\boldsymbol{Z}_{\alpha,\beta}^{res,i}$ is used to compute the i-th bit of the output wire label $C_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}$. Therefore, non-trivial linear combinations of vectors in $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | i \in [\kappa]\}$ can be used to compute non-trivial linear combinations of κ bits of $C_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}$. Based on *Claim* 4, at least two types of oracle responses are required to compute any non-trivial linear combination of κ bits of $C_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}$.

4.3 Proof of a Lower Bound in the First Model

With the help of *Claim* 4, we can prove a lower bound of our model. Precisely speaking, we consider a large class of practical garbling schemes, which work on arbitrary AND gates and are compatible with free-XOR. Note that our proof is *independent* of the dicing technique, as long as *Claim* 4 holds.

We concentrate on a set of 4κ vectors $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}, i \in [\kappa]\}$. *Claim* 4 points out that elements in $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}\}$ are pairwise linearly independent. Note that when free-XOR is supported, the output wire labels C_0 and C_1 satisfy $C_1 = C_0 \oplus \Delta$. Hence, given permute bits a, b and i, elements in the set $\{\boldsymbol{C}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}^{res,i} | \alpha, \beta \in \{0,1\}\}$ are the same. Therefore, the garbler needs to arrange 1-bit ciphertexts to ensure that evaluators with different input wire labels can transform different $\boldsymbol{Z}_{\alpha,\beta}^{res,i}$ into the same $\boldsymbol{C}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}^{res,i}$. [5]

In the half-gates garbling scheme, we find that for a given i, elements in $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}\}$ could be linear dependent. (We shake of trivial combina-

[5] Actually, ciphertexts are used to transform $\langle \boldsymbol{Z}_{\alpha,\beta}^{res,i}, \boldsymbol{S}^{res} \rangle$ into the same $\langle \boldsymbol{C}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}^{res,i}, \boldsymbol{S}^{res} \rangle$. Our statements have been simplified for brevity.

tion.) As noted above, the "three-halves" garbling scheme also shows linear dependence between elements of $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\}\}$ and $\{\boldsymbol{Z}^{res,j}_{\alpha,\beta}|\alpha,\beta\in\{0,1\}\}$ for $i\neq j$. Crucially, both situations lead to the saving of ciphertexts.

Lemma 1. *In the model of Bitwise Linear Garbling Schemes, suppose free-XOR is supported. If the rank of the set* $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\},i\in[\kappa]\}$ *is* rk*, then* $m\geq rk-\kappa$.

Proof. The evaluator $E_{\alpha,\beta}$ can only compute κ vectors in $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|i\in[\kappa]\}$ with her input wire labels. With less than $rk-\kappa$ ciphertexts, she fails to learn all the vectors in $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\},i\in[\kappa]\}$. This means at least one $\boldsymbol{Z}^{res,i}_{\alpha',\beta'}$ can not be computed, where $(\alpha',\beta')\neq(\alpha,\beta)$, and $i\in[\kappa]$. In other words, $\boldsymbol{Z}^{res,i}_{\alpha,\beta}$ and $\boldsymbol{Z}^{res,i}_{\alpha',\beta'}$ can not be transformed into the same $\boldsymbol{C}^{res,i}_{a,b,(a\oplus\alpha)\wedge(b\oplus\beta)}$. Therefore, required ciphertexts are at least $rk-\kappa$ bits. □

One can check this by placing the "three-halves" garbling scheme in our model. The set $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\},i\in[\kappa]\}$ constructed in that way is of rank $\frac{5}{2}\kappa$, while this scheme uses $\frac{3}{2}\kappa$ 1-bit ciphertexts. Hence, the key point of our proof is the lower bound of rk.

To begin our proof of Theorem 1, we present Lemma 2 below.

Lemma 2. *For a given set of linearly independent vectors* $\boldsymbol{Y}_i, i\in[l]$ *with entries in* $\{0,1\}$*, where* l *is a positive integer, suppose a set of vectors* $\mathcal{Z}$ *such that* $\forall i\in[l]$, $\boldsymbol{Y}_i\in\mathcal{Z}$. *If there exists a vector* $\boldsymbol{L}$ *with entries in* $\{0,1\}$ *such that* $\bigoplus_{i=0}^{l-1}\boldsymbol{Y}_i=\boldsymbol{L}$ *where* $\oplus$ *denotes addition modulo 2, then replacing any vector* $\boldsymbol{Y_i}$, $i\in[l]$ *with* $\boldsymbol{L}$ *or adding* $\boldsymbol{L}$ *into* $\mathcal{Z}$ *does not change the rank of* $\mathcal{Z}$.

Proof. Note that $\boldsymbol{L}$ can be linearly represented by vectors in $\mathcal{Z}$. Adding $\boldsymbol{L}$ into $\mathcal{Z}$ does not introduce new linearly independent vector. Therefore, the rank of $\mathcal{Z}$ does not change in this situation.

Without loss of generality, we replace $\boldsymbol{Y}_1$ with $\boldsymbol{L}$ and obtain $\mathcal{Z}'=(\mathcal{Z}\cup\{\boldsymbol{L}\})\setminus\{\boldsymbol{Y}_1\}$. If $\boldsymbol{Y}_1$ can be linearly represented by vectors in $\mathcal{Z}\setminus\{\boldsymbol{Y}_1\}$, then $\boldsymbol{L}$ can also be linearly represented, because $\bigoplus_{i=1}^{l}\boldsymbol{Y}_i=\boldsymbol{L}$. Otherwise, neither $\boldsymbol{Y}_1$ nor $\boldsymbol{L}$ can be linearly represented. Hence, $rank(\mathcal{Z}')=rank(\mathcal{Z})$. □

Theorem 1. *In the model of Bitwise Linear Garbling Schemes, suppose free-XOR is supported. Let* rk *be the rank of* $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\},i\in[\kappa]\}$*, the lower bound of* rk *is* $\frac{5}{2}\kappa$*, and therefore* $m\geq\frac{3}{2}\kappa$.

Proof. We compute the lower bound of rk by counting, and this is where *Claim* 4 comes into play. We use a set $\mathcal{Z}$ to include vectors in $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta\in\{0,1\},i\in[\kappa]\}$ gradually. Note that $\boldsymbol{Z}^{res,i}_{\alpha,\beta}$ has a length of $6q$, with entries in $\{0,1\}$. Recall that components of $\boldsymbol{Z}^{res,i}_{\alpha,\beta}=(\boldsymbol{Z}^{i}_{\alpha,\beta,1},\ldots,\boldsymbol{Z}^{i}_{\alpha,\beta,6})$ correspond to oracle responses of the forms $H(A_0),H(A_1),H(B_0),H(B_1),H(A_0\oplus B_0),H(A_0\oplus B_1)$.

1) We add κ vectors in $\{\boldsymbol{Z}^{res,i}_{0,0}|i\in[\kappa]\}$ into the set $\mathcal{Z}$ to obtain rank κ. Based on *Claim* 4, these vectors are linearly independent.

2) Now κ vectors in $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$ are also added into $\mathcal{Z}$. Note that

$$\boldsymbol{Z}_{0,0}^{res,i} = (\boldsymbol{Z}_{0,0,1}^{i}, \mathbf{0}, \boldsymbol{Z}_{0,0,3}^{i}, \mathbf{0}, \boldsymbol{Z}_{0,0,5}^{i}, \mathbf{0}),$$

while

$$\boldsymbol{Z}_{0,1}^{res,i} = (\boldsymbol{Z}_{0,1,1}^{i}, \mathbf{0}, \mathbf{0}, \boldsymbol{Z}_{0,1,4}^{i}, \mathbf{0}, \boldsymbol{Z}_{0,1,6}^{i}).$$

If $\boldsymbol{Z}_{0,0}^{res,i}$ can be linearly represented by vectors in $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$, then there exist coefficients $y_1, y_2, \ldots, y_\kappa \in \{0,1\}$ such that $\bigoplus_{k=1}^{\kappa} y_k \boldsymbol{Z}_{0,1}^{res,k} = \boldsymbol{Z}_{0,0}^{res,i}$. Moreover,

$$\bigoplus_{k=1}^{\kappa} y_k \boldsymbol{Z}_{0,1}^{res,k} = (\bigoplus_{k=1}^{\kappa} y_k \boldsymbol{Z}_{0,1,1}^{k}, \mathbf{0}, \mathbf{0}, \bigoplus_{k=1}^{\kappa} y_k \boldsymbol{Z}_{0,1,4}^{k}, \mathbf{0}, \bigoplus k = 1^{\kappa} y_k \boldsymbol{Z}_{0,1,6}^{k}).$$

Hence, we can find that only q entries of $\boldsymbol{Z}_{0,0,1}^{i}$ in $\boldsymbol{Z}_{0,0}^{res,i}$ can be nonzero. Therefore, $sum(\boldsymbol{Z}_{0,0}^{res,i}) \leq 1$ and this violates *Claim* 4. So $\boldsymbol{Z}_{0,0}^{res,i}$ can not be linearly represented by vectors in $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$. Similarly, we can check that any non-trivial linear combination of vectors in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ can not be linearly represented. Therefore, the rank of this set is 2κ.

3) We have to consider $\{\boldsymbol{Z}_{1,0}^{res,i} | i \in [\kappa]\}$ now. However, adding κ to the rank of this set again can not be accepted, because the "three-halves" garbling scheme already achieves rank $\frac{5}{2}\kappa$. We have to add some vectors without the increase of the rank, so we need $\boldsymbol{Z}_{1,0}^{res,i}$ which can be linearly represented by vectors in $\mathcal{Z}$. Given that

$$\boldsymbol{Z}_{1,0}^{res,i} = (\mathbf{0}, \boldsymbol{Z}_{1,0,2}^{i}, \boldsymbol{Z}_{1,0,3}^{i}, \mathbf{0}, \mathbf{0}, \boldsymbol{Z}_{1,0,6}^{i}),$$

$\boldsymbol{Z}_{1,0,2}^{i}$, $\boldsymbol{Z}_{1,0,3}^{i}$ and $\boldsymbol{Z}_{1,0,6}^{i}$ respectively correspond to oracle responses of the forms $H(A_1)$, $H(B_0)$ and $H(A_0 \oplus B_1)$. We consider $\boldsymbol{Z}_{1,0}^{res,i}$ where $\boldsymbol{Z}_{1,0,2}^{i} = \mathbf{0}$, because $\mathcal{Z}$ does not contain nonzero vectors corresponding to $H(A_1)$. Hence, suppose that γ vectors satisfy $\boldsymbol{Z}_{1,0,2}^{i} = \mathbf{0}$, while the remaining $(\kappa - \gamma)$ vectors still have nonzero $\boldsymbol{Z}_{1,0,2}^{i}$. Note that if these nonzero $\boldsymbol{Z}_{1,0,2}^{i}$ are linearly dependent, then we can construct another $\boldsymbol{Z}_{1,0}^{res,i}$ such that $\boldsymbol{Z}_{1,0,2}^{i}$ is zero by Lemma 2. Hence, these nonzero $\boldsymbol{Z}_{1,0,2}^{i}$ are linearly independent. Certainly, those $(\kappa - \gamma)$ vectors are going to increase the rank by $(\kappa - \gamma)$. Now, we consider these γ vectors with zero $\boldsymbol{Z}_{1,0,2}^{i}$. Because of *Claim* 4,

$$\boldsymbol{Z}_{1,0}^{res,i} = (\mathbf{0}, \mathbf{0}, \boldsymbol{Z}_{1,0,3}^{i}, \mathbf{0}, \mathbf{0}, \boldsymbol{Z}_{1,0,6}^{i})$$

must have nonzero $\boldsymbol{Z}_{1,0,3}^{i}$ and $\boldsymbol{Z}_{1,0,6}^{i}$. We require that vectors in $\mathcal{Z}$ can linearly represent them. The only way is to use $(\boldsymbol{Z}_{0,0,1}^{i}, \mathbf{0}, \boldsymbol{Z}_{1,0,3}^{i}, \mathbf{0}, \mathbf{0}, \mathbf{0})$ and $(\boldsymbol{Z}_{0,0,1}^{i}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{Z}_{1,0,6}^{i})$. By Lemma 2, we might as well assume that there are γ vectors of each form in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ and $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$. Consequently, $rank(\mathcal{Z}) = 3\kappa - \gamma$ after we put these vectors into $\mathcal{Z}$.

4) Finally, we need to add $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$. Given

$$\boldsymbol{Z}_{1,1}^{res,i} = (\boldsymbol{0}, \boldsymbol{Z}_{1,1,2}^{i}, \boldsymbol{0}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{0}),$$

we can also classify κ vectors in $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$ based on whether $\boldsymbol{Z}_{1,1,2}^{i}$ is nonzero or not. We suppose that δ vectors satisfy $\boldsymbol{Z}_{1,1,2}^{i} = \boldsymbol{0}$, while the remaining $(\kappa - \delta)$ vectors have nonzero $\boldsymbol{Z}_{1,1,2}^{i}$.
We get δ vectors of the form:

$$\boldsymbol{Z}_{1,1}^{res,i} = (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{0}).$$

The two nonzero vectors $\boldsymbol{Z}_{1,1,4}^{i}$ and $\boldsymbol{Z}_{1,1,5}^{i}$ correspond to $H(B_1)$ and $H(A_0 \oplus B_0)$. To ensure that these $\boldsymbol{Z}_{1,1}^{res,i}$ do not increase the rank, we hope that there are already δ vectors of the form $(\boldsymbol{Z}_{0,0,1}^{i}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{0})$ in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$, and $(\boldsymbol{Z}_{0,0,1}^{i}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,1,4}^{i}, \boldsymbol{0}, \boldsymbol{0}))$ in $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$. However, in step 3), there are γ vectors of the form $(\boldsymbol{Z}_{0,0,1}^{i}, \boldsymbol{0}, \boldsymbol{Z}_{1,0,3}^{i}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$ in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ and $(\boldsymbol{Z}_{0,0,1}^{i}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,0,6}^{i})$ in $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$.
Let us think about these $(\kappa - \delta)$ vectors which have nonzero $\boldsymbol{Z}_{1,1,2}^{i}$ corresponding to $H(A_1)$. We glance at the set $\mathcal{Z}$ and find that it is only possible to use $(\kappa - \gamma)$ vectors

$$\boldsymbol{Z}_{1,0}^{res,j} = (\boldsymbol{0}, \boldsymbol{Z}_{1,0,2}^{j}, \boldsymbol{Z}_{1,0,3}^{j}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,0,6}^{j})$$

where $\boldsymbol{Z}_{1,0,2}^{j} \neq \boldsymbol{0}$. Hence, to avoid the raise of $rank(\mathcal{Z})$, we require that $\boldsymbol{Z}_{1,1,2}^{i}$ can be linearly represented by these nonzero $\boldsymbol{Z}_{1,0,2}^{j}$. Meanwhile, even if $\boldsymbol{Z}_{1,1,2}^{i} = \boldsymbol{Z}_{1,0,2}^{j}$, we still have to consider

$$\boldsymbol{Z}_{1,1}^{res,i} \oplus \boldsymbol{Z}_{1,0}^{res,j} = (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,0,3}^{j}, \boldsymbol{Z}_{1,1,4}^{i}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{Z}_{1,0,6}^{j}).$$

However, dealing with vectors of this form in the following analysis is complex, so we assume that they either do not affect $rank(\mathcal{Z})$ or they increase $rank(\mathcal{Z})$, and directly check that they can be linearly represented when $rank(\mathcal{Z})$ reaches its minimum. In this way, omitting these vectors does not influence correctness.

Combining the above, we can find that $rank(\mathcal{Z})$ consists of three parts: $3\kappa - \gamma$, those $(\kappa - \delta)$ vectors with nonzero $\boldsymbol{Z}_{1,1,2}^{i}$ and the remaining δ vectors with zero $\boldsymbol{Z}_{1,1,2}^{i}$.

Firstly, let us consider all $(\kappa - \delta)$ vectors with nonzero $\boldsymbol{Z}_{1,1,2}^{i}$. Note that there are $(\kappa - \gamma)$ vectors with nonzero $\boldsymbol{Z}_{1,0,2}^{j}$ in $\mathcal{Z}$. This means that if $\kappa - \delta \leq \kappa - \gamma$, it is possible that these $(\kappa - \delta)$ vectors do not change $rank(\mathcal{Z})$. Otherwise, $\gamma > \delta$, and these vectors increase the rank by at least $(\kappa - \delta) - (\kappa - \gamma) = \gamma - \delta$. Note that these uncertainties come from neglected $(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{Z}_{1,0,3}^{j}, \boldsymbol{Z}_{1,1,4}^{i}, \boldsymbol{Z}_{1,1,5}^{i}, \boldsymbol{Z}_{1,0,6}^{j})$.

Secondly, we need 2κ vectors in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ and $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$ to linearly represent both δ vectors with zero $\boldsymbol{Z}_{1,1,2}^{i}$ and γ vectors with zero $\boldsymbol{Z}_{1,0,2}^{i}$.

This means that if $\delta+\gamma \le \kappa$, these vectors do not change $rank(\mathcal{Z})$. Otherwise, $\delta+\gamma > \kappa$, and these vectors increase the rank by $\delta+\gamma-\kappa$.

1. If $\kappa-\delta \le \kappa-\gamma$ and $\delta+\gamma \le \kappa$, then $rank(\mathcal{Z}) \ge 3\kappa-\gamma$. Since $\gamma \le \delta$ and $\delta+\gamma \le \kappa$, we get $\gamma \le \frac{1}{2}\kappa$. Hence, when $\gamma=\delta=\frac{1}{2}\kappa$, $rank(\mathcal{Z})$ reaches its minimum $\frac{5}{2}\kappa$ in this case. We can easily check that when $rank(\mathcal{Z})=\frac{5}{2}\kappa$, those $(\kappa-\delta)$ vectors with nonzero $\boldsymbol{Z}^i_{1,1,1}$ can be linearly represented.
2. If $\kappa-\delta \le \kappa-\gamma$ and $\delta+\gamma > \kappa$, then $rank(\mathcal{Z}) \ge (3\kappa-\gamma)+(\delta+\gamma-\kappa) \ge 2\kappa+\delta$. Since $\gamma \le \delta$ and $\delta+\gamma > \kappa$, we get $\delta > \frac{1}{2}\kappa$. Hence, $rank(\mathcal{Z}) > \frac{5}{2}\kappa$ in this case.
3. If $\kappa-\delta > \kappa-\gamma$ and $\delta+\gamma \le \kappa$, then $rank(\mathcal{Z}) \ge (3\kappa-\gamma)+(\gamma-\delta) \ge 3\kappa-\delta$. Since $\gamma > \delta$ and $\delta+\gamma \le \kappa$, we get $\delta < \frac{1}{2}\kappa$. $rank(\mathcal{Z}) > \frac{5}{2}\kappa$ in this case.
4. If $\kappa-\delta > \kappa-\gamma$ and $\delta+\gamma > \kappa$, then $rank(\mathcal{Z}) \ge (3\kappa-\gamma)+(\gamma-\delta)+(\delta+\gamma-\kappa) \ge 2\kappa+\gamma$. Since $\gamma > \delta$ and $\delta+\gamma > \kappa$, we get $\gamma > \frac{1}{2}\kappa$. $rank(\mathcal{Z}) > \frac{5}{2}\kappa$ in this case.

All in all, we prove a $\frac{5}{2}\kappa$ lower bound of $rank(\mathcal{Z})$, and a $\frac{3}{2}\kappa$-bit lower bound of ciphertexts by Lemma 1. □

Removed Forms. We consider the forms $\{H(A_i,B_j)|i,j \in \{0,1\}\}$ which have been ruled out before we propose *Claim* 4. We insist that we let $\{H(A_i,B_j)|i,j \in \{0,1\}\}$ associate with $\{E_{i,j}\}$, rather than requiring that the random oracle must be queried in this form.

To consider them in the proof of Theorem 1, we should choose $t=10$ and consider 10 types of oracle responses. In order not to affect *Claim* 4, we slightly modify the sign function v. Obviously, computing $H(A_i,B_j)$ requires two wire labels A_i and B_j. Hence, $H(A_i,B_j)$ is different from oracle responses $H(A_i)$ and $H(B_j)$. Suppose there is only a nonzero vector $\boldsymbol{V}$ in $\boldsymbol{Z}^{res,k}_{i,j}$ corresponding to $H(A_i,B_j)$ where $i,j \in \{0,1\}$ and $k \in [\kappa]$. Because both $H(A_i,B_j)$ and $H(A_i) \oplus H(B_j)$ can only be computed by $\{E_{i,j}\}$, it is reasonable that $sum(\boldsymbol{Z}^{res,k}_{i,j})=2$. Therefore, $sum(\boldsymbol{Z}^{res,k}_{i,j})=v(\boldsymbol{V})=2$.

Intuitively, considering $\{H(A_i,B_j)|i,j \in \{0,1\}\}$ allows us to use $H(A_i,B_j)_k$ to replace $H(A_i)_k \oplus H(B_j)_k$. $\boldsymbol{Z}^{res,k}_{i,j}$ satisfying $\langle \boldsymbol{Z}^{res,k}_{i,j}, \boldsymbol{S}^{res}\rangle = H(A_i)_k \oplus H(B_j)_k$ can be linearly represented by other vectors. For example, $H(A_i) \oplus H(B_j) = [H(A_i) \oplus H(A_i \oplus B_{i-j})] \oplus [H(B_j) \oplus H(A_{1-i} \oplus B_j)]$ in which $A_i \oplus B_{i-j} = A_{1-i} \oplus B_j$. However, it is easy to find that $(\boldsymbol{Z}^{res,k}_{i,j})'$ satisfying $\langle (\boldsymbol{Z}^{res,k}_{i,j})', \boldsymbol{S}^{res}\rangle = H(A_i,B_j)_k$ can not be linearly represented.

According to the proof of Theroem 1, when rk reaches its minimum, every vector in $\{\boldsymbol{Z}^{res,i}_{\alpha,\beta}|\alpha,\beta \in \{0,1\}, i \in [\kappa]\}$ can be linearly represented by other vectors. Since $(\boldsymbol{Z}^{res,k}_{i,j})'$ can not be linearly represented, considering the forms $\{H(A_i,B_j)|i,j \in \{0,1\}\}$ does not beat the $\frac{3}{2}\kappa$-bit lower bound.

4.4 The Second Model: Bitwise Garbling Schemes

In this subsection, we define the model of Bitwise Garbling Schemes.

Garble: This algorithm is parameterized by integers m, r, q, t, vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$ and the **mapping** function Map. Map breaks down into 4κ functions $\{Map^k_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$. Each vector has length r, with entries in $GF(2^\kappa)$. Meanwhile, vectors $\{\boldsymbol{Z}^{res,k}_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ are of length tq, with entries in $GF(2)$.

1. For $i \in [r]$, choose $R_i \xleftarrow{\$} GF(2^\kappa)$ to get $\boldsymbol{R} = \{R_1, \ldots, R_r\}$.
2. For $i \in \{0,1\}$, compute $A_i = \langle \boldsymbol{A}_i, \boldsymbol{R} \rangle$, $B_i = \langle \boldsymbol{B}_i, \boldsymbol{R} \rangle$.
3. Choose two permute bits $a, b \xleftarrow{\$} \{0,1\}$ for two input wires.
4. For t types of oracle responses, make tq distinct queries to the random oracle and get tq bits $Q^i_1, \ldots, Q^i_q$, $i \in [t]$. We place these responses in $\boldsymbol{S} = (R_1, \ldots, R_r, Q^1_1, \ldots, Q^t_q)$. Note that $\boldsymbol{S}^{res} = (Q^1_1, \ldots, Q^t_q)$.
5. For $i, j \in \{0,1\}, k \in [\kappa]$, we compute $Z^k_{i,j} = \langle \boldsymbol{Z}^{res,k}_{i,j}, \boldsymbol{S}^{res} \rangle$. For the sake of discussion, let $\mathcal{Z} = \{Z^k_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$.
6. Find m 1-bit ciphertexts in $\boldsymbol{G} = (G_1, G_2, \ldots, G_m)$ such that

$$Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G}) = \begin{pmatrix} C_{0,0} \\ C_{0,1} \\ C_{1,0} \\ C_{1,1} \end{pmatrix},$$

in which $Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G})$ is defined as

$$Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G}) \triangleq \begin{pmatrix} Map^1_{0,0}(Z^1_{0,0}, \boldsymbol{S}, \boldsymbol{G}) \parallel \cdots \parallel Map^\kappa_{0,0}(Z^\kappa_{0,0}, \boldsymbol{S}, \boldsymbol{G}) \\ Map^1_{0,1}(Z^1_{0,1}, \boldsymbol{S}, \boldsymbol{G}) \parallel \cdots \parallel Map^\kappa_{0,1}(Z^\kappa_{0,1}, \boldsymbol{S}, \boldsymbol{G}) \\ Map^1_{1,0}(Z^1_{1,0}, \boldsymbol{S}, \boldsymbol{G}) \parallel \cdots \parallel Map^\kappa_{1,0}(Z^\kappa_{1,0}, \boldsymbol{S}, \boldsymbol{G}) \\ Map^1_{1,1}(Z^1_{1,1}, \boldsymbol{S}, \boldsymbol{G}) \parallel \cdots \parallel Map^\kappa_{1,1}(Z^\kappa_{1,1}, \boldsymbol{S}, \boldsymbol{G}) \end{pmatrix}$$

and $\{C_{i,j} | i, j \in \{0,1\}\}$ are valid output wire labels. For the sake of brevity, we let $\mathcal{C} = \{C^k_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ and write as $\mathcal{C} = Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G})$.

Encode: On input $x_a, x_b \in \{0,1\}$, set color bits $\alpha := x_a \oplus a$ and $\beta := x_b \oplus b$. The evaluator gets $A_\alpha \parallel \alpha$ and $B_\beta \parallel \beta$.
Eval: Parameterized by mapping function Map, m, q, u and vectors $\{\boldsymbol{V}^i_{\alpha,\beta} | \alpha, \beta \in \{0,1\}, i \in [\kappa]\}$ of length uq.

1. The evaluator has input wire labels $A_\alpha \parallel \alpha$, $B_\beta \parallel \beta$, and ciphertexts $G_1, \ldots, G_m$.
2. Define a function $f : [u] \to [t]$. For u types of oracle responses, make uq distinct queries to the random oracle and get responses $Q^{f(j)}_1, \ldots, Q^{f(j)}_q$, where $j \in [u], f(j) \in [t]$. We place these values in $\boldsymbol{T} = (A_\alpha, B_\beta, Q^{f(1)}_1, \ldots, Q^{f(u)}_q)$. Note that $\boldsymbol{T}^{res} = (Q^{f(1)}_1, \ldots, Q^{f(u)}_q)$.
3. For $i \in [\kappa]$, compute $V^i_{\alpha,\beta} = \langle \boldsymbol{V}^i_{\alpha,\beta}, \boldsymbol{T}^{res} \rangle$.
4. Compute $C^k_{\alpha,\beta} = Map^k_{\alpha,\beta}(V^k_{\alpha,\beta}, \boldsymbol{T}, \boldsymbol{G})$ for $k \in [\kappa]$. Take $C_{\alpha,\beta} = C^1_{\alpha,\beta} \parallel \cdots \parallel C^\kappa_{\alpha,\beta}$ as the output wire label.

Compared to the model of Bitwise Linear Garbling Schemes, this model utilizes mapping functions to compute the output wire label. Clearly, our first model is already included in this model when we restrict that mapping functions are linear, e.g., bitwise XOR operations. After removing the limitation of linearity, our model deals with non-linear mapping.

Considering two algorithms Garble and Eval, $\{Map^k_{\alpha,\beta}(V^k_{\alpha,\beta}, \boldsymbol{T}, \boldsymbol{G}) | \alpha, \beta \in \{0,1\}, k \in \{0,1\}\}$ are consistent with $\{Map^k_{i,j}(Z^k_{i,j}, \boldsymbol{S}, \boldsymbol{G}) | i, j \in \{0,1\}, k \in [\kappa]\}$. Therefore, $E_{i,j}$, who computes $Map^k_{i,j}(Z^k_{i,j}, \boldsymbol{S}, \boldsymbol{G})$, does not use values which are in $\boldsymbol{S}$ but not in $\boldsymbol{T}$.

4.5 Proof of a Lower Bound in the Second Model

In Sect. 4.3, we prove that $\{\boldsymbol{Z}^{res,k}_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ has a rank of at least $\frac{5}{2}\kappa$. Since $\boldsymbol{S}^{res}$ is uniformly distributed, $\mathcal{Z}$ has at least 2.5κ degrees of freedom.

Lemma 3. *In the model of Bitwise Garbling Schemes, for a given $\boldsymbol{G}$, $\mathcal{C} = Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G})$ has at least 2.5κ degrees of freedom.*

Proof. Based on the proof of Sect. 4.3, $\mathcal{Z}$ has at least 2.5κ degrees of freedom. Otherwise, *Claim* 4 is violated and privacy is broken.

Because $\boldsymbol{G}$ is already given, $E_{i,j}$ who is able to compute $Z^k_{i,j}$ can also compute $Map^k_{i,j}(Z^k_{i,j}, \boldsymbol{S}, \boldsymbol{G})$. It is absurd that $\{\boldsymbol{Z}^{res,k}_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ should have a rank of at least $\frac{5}{2}\kappa$ to keep secure, while output wire labels $\mathcal{C} = Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G})$ have less than 2.5κ degrees of freedom. $\square$

Remark 1. Note that mapping functions $\{Map^k_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ taking $\boldsymbol{S}$ as input allow non-linear actions between wire labels and oracle responses. Some practical garbling schemes [25] based on polynomial interpolation perform standard addition and subtraction on wire labels, in which carries may be generated. Hence, mapping functions provide coverage of them.

Theorem 2. *In the model of Bitwise Garbling Schemes, suppose free-XOR is supported. The lower bound of m is $\frac{3}{2}\kappa$.*

Proof. According to Lemma 3, when $\boldsymbol{G}$ is given, $\mathcal{C} = Map(\mathcal{Z}, \boldsymbol{S}, \boldsymbol{G})$ has at least 2.5κ degrees of freedom. When free-XOR is supported, output wire labels encoding logic values 0 and 1 keep a global XOR-difference Δ which is previously sampled. Therefore, $\mathcal{C}$ has κ degrees of freedom. Obviously, $\boldsymbol{G}$ should have a length of at least 1.5κ, so $m \geq \frac{3}{2}\kappa$. $\square$

Remark 2. We do not restrict that $\{Map^k_{i,j} | i, j \in \{0,1\}, k \in [\kappa]\}$ must be linear. Therefore, we immediately realize that we can include multiplication which is rather common in garbling scheme design. We can perform multiplication between wire labels, oracle responses and ciphertexts. Of course, not necessarily limited by multiplication, the second model is actually more inclusive.

While we assume that the evaluator knows how to compute her output wire label, the "three-halves" garbling scheme [26] is literally outside of our first model. Precisely speaking, this scheme follows *Claim* 4, so the $\frac{3}{2}\kappa$-bit lower bound holds. However, it deviates from the description of this model, because this scheme performs unfixed actions on wire labels, while $\boldsymbol{Z}^{lab,i}_{\alpha,\beta}$ in this model (which represent actions on wire labels and consist of entries in $\boldsymbol{Z}^{i}_{\alpha,\beta}$ but not in $\boldsymbol{Z}^{res,i}_{\alpha,\beta}$) are fixed. To be specific, the evaluator needs additional bits apart from ciphertexts to decide her actions, in consideration of the dicing technique.

Our second model includes the "three-halves" garbling scheme, since this model only requires that oracle responses associated with the evaluator must be produced. Furthermore, this model includes the dicing technique. The dicing technique allows the garbler to transmit additional bits, which can now be regarded as ciphertexts, to control the actions of the evaluator. In the setting about practical garbling schemes, this can only achieved by the random oracle. In other words, different from the first model, the evaluator uses wire labels and oracle responses in $\boldsymbol{S}$ and ciphertexts $(G_1, \ldots, G_m)$ to decide her actions $\boldsymbol{V}^{i}_{\alpha,\beta}$. Obviously, the mapping function Map includes this idea.

5 Compatibility with Free-XOR

We already prove the $\frac{3}{2}$-bit lower bound of our models when considering free-XOR, and find that the XOR-difference Δ plays a crucial role in reducing the rank of $\mathcal{Z}$. However, the output labels C_0 and C_1 are also restricted by free-XOR, i.e., given i, all the elements in $\{\boldsymbol{C}^{res,i}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ are the same. If we do not use the free-XOR technique, the output wire labels are not required to keep the same XOR-difference. Hence, these elements are not necessarily the same. Contrast with Lemma 1, we may only need $rk - 2\kappa$ ciphertexts. Consequently, we explore whether giving up compatibility with free-XOR is a necessary sacrifice.

5.1 Similarity to Free-XOR

First of all, we must explain how to sacrifice compatibility with free-XOR. When free-XOR is used, $A_0 \oplus B_1 = A_1 \oplus B_0$. Note that $H(A_i \oplus B_j)$ is associated with $\{E_{i,j}, E_{1-i,1-j}\}$ for $i, j \in \{0,1\}$. One may think that if $A_0 \oplus A_1 \neq B_0 \oplus B_1$, free-XOR is forbidden. However, a garbling scheme may ensure that $A_1 = A_0 + d$ and $B_0 = B_1 - d$, where $d \xleftarrow{\$} \{0,1\}^\kappa$ and "+" or "-" denotes standard addition or subtraction. In this case, $A_0 + B_1 = A_1 + B_0$ and $A_0 - B_0 = A_1 - B_1$. Similarly, we can construct forms $H(A_0 - B_0)$ and $H(A_0 + B_1)$ which are associated with $\{E_{0,0}, E_{1,1}\}$ and $\{E_{0,1}, E_{1,0}\}$ respectively. It is easy to find that the lower bound of this garbling scheme is also $\frac{3}{2}\kappa$ bits. Even though XOR gates are not free, we still argue that this construction is similar to free-XOR.

To get rid of free-XOR, a garbling scheme should ensure that $H(l(A_i, B_j))$ associated with $\mathcal{E}_{l(A_i,B_j)} = \{E_{0,0}, E_{1,1}\}$ (and $\{E_{0,1}, E_{1,0}\}$) does not exist. Therefore, we propose Definition 3 for garbling schemes supporting quasi-free-XOR or not. [6]

Definition 3. *In a garbling scheme, for an arbitrary AND gate with input wire labels (A_0, A_1) and (B_0, B_1), if and only if there are oracle responses associated with $\{E_{0,0}, E_{1,1}\}$ and $\{E_{0,1}, E_{1,0}\}$, this scheme supports quasi-free-XOR.*

5.2 Lower Bound without Quasi-Free-XOR

We manage to give the lower bound of our first model without quasi-free-XOR. It seems that when C_0 and C_1 are independent, the number of ciphertexts is not necessarily $rk-\kappa$, because we can use different linear combinations of oracle responses to compute C_0^i and C_1^i.

Lemma 4. *In the model of Bitwise Linear Garbling Schemes, suppose quasi-free-XOR is forbidden. For given permute bits $a, b \in \{0,1\}$, if the rank of the set $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | (\alpha, \beta) \in \{0,1\}^2 \setminus \{(1-a, 1-b)\}, i \in [\kappa]\}$ is rk, then $m \geq rk - \kappa$.*

Proof. Given permute bits a, b, evaluators in $\{E_{i,j} | (i,j) \in \{0,1\}^2 \setminus \{(1-a, 1-b)\}\}$ get the output wire label encoding logic value 0. Similar to Lemma 1, for a given i, $\{\boldsymbol{Z}_{\alpha,\beta}^{res,i} | (\alpha, \beta) \in \{0,1\}^2 \setminus \{(1-a, 1-b)\}\}$ must be transformed into the same vector. Hence, at least $rk - \kappa$ ciphertexts are needed. □

Without loss of generality, suppose $a = 0$ and $b = 0$. The output wire label of (A_1, B_1) is C_1. $E_{0,0}$, $E_{0,1}$ and $E_{1,0}$ get the same output. Consequently, given $i \in [\kappa]$, $\boldsymbol{C}_{0,0}^{res,i} = \boldsymbol{C}_{0,1}^{res,i} = \boldsymbol{C}_{1,0}^{res,i}$. We still rule out oracle responses of the form $H(A_i, B_j)$. Without quasi-free-XOR, oracle responses assoicated with $\{E_{0,0}, E_{1,1}\}$ and $\{E_{0,1}, E_{1,0}\}$ do not exist. Therefore, $t = 4$ and we arrange

$$\boldsymbol{Z}_{\alpha,\beta}^{res,i} = (\boldsymbol{Z}_{\alpha,\beta,1}^{i}, \ldots, \boldsymbol{Z}_{\alpha,\beta,4}^{i})$$

which corresponds to forms $H(A_0), H(A_1), H(B_0), H(B_1)$. It is obvious that *Claim* 4 still holds. We can prove the 2κ-bit lower bound of ciphertexts, still by counting.

Theorem 3. *In the model of Bitwise Linear Garbling Schemes, suppose quasi-free-XOR is forbidden. Then, $m \geq 2\kappa$.*

Proof. We use the set $\mathcal{Z}$ to include vectors.

1) Add κ vectors in $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ into the set $\mathcal{Z}$, to obtain rank κ.
2) $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$ are also added into $\mathcal{Z}$. For the same reason as the proof of Theorem 1, the rank of this set is now 2κ.

[6] We can modify Theorem 1 and 2 by supposing quasi-free-XOR (instead of free-XOR) is supported.

3) We have to consider $\{\boldsymbol{Z}_{1,0}^{res,i} | i \in [\kappa]\}$ now. Note that

$$\boldsymbol{Z}_{1,0}^{res,i} = (\mathbf{0}, \boldsymbol{Z}_{1,0,1}^{i}, \mathbf{0}, \boldsymbol{Z}_{1,0,3}^{i}).$$

Based on *Claim* 4, $\boldsymbol{Z}_{1,0,1}^{i}$ and $\boldsymbol{Z}_{1,0,3}^{i}$ are nonzero. We realize that $rank(\mathcal{Z})$ is 3κ. Reducing this rank is impossible, because these κ linearly independent vectors $\boldsymbol{Z}_{1,0,1}^{i}$, which correspond to $H(A_1)$, are absent in the first two steps. Compared to Sect. 4.3, we lack a form $H(l(A_i, B_j))$ associated with $\mathcal{E}_{l(A_i,B_j)} = \{E_{0,1}, E_{1,0}\}$.

4) Finally, it makes no difference whether vectors in $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$ can be linearly represented by vectors in $\mathcal{Z}$ or not, because the garbler can arrange that $\boldsymbol{C}_{1,1}^{i} = \boldsymbol{Z}_{1,1}^{i}$. The evaluator $E_{1,1}$ needs no ciphertext to compute her output wire label. (Certainly, it is also easy to check that $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$ can be linearly represented by vectors in $\mathcal{Z}$.) Consequently, we only consider $rank(\mathcal{Z}) = 3\kappa$ at the end of step 3).

Based on Lemma 4, we need $rank(\mathcal{Z}) - \kappa$ ciphertexts, so $m \geq 2\kappa$. This result is true for any possible (a, b). Hence, the lower bound of m is 2κ. □

This proof can be regarded as the answer to another question in [26]: it is helpless to sacrifice compatibility with free-XOR.

Bitwise Garbling Schemes. Again, we extend this result into our second model.

Theorem 4. *In the model of Bitwise Linear Garbling Schemes, suppose quasi-free-XOR is forbidden. The lower bound of* m *is* 2κ*.*

Proof. Still, without loss of generality, assume $a = 0$ and $b = 0$. On this occasion, $\{Z_{i,j}^{k} | (i,j) \in \{0,1\}^2 \setminus \{(1,1)\}, k \in [\kappa]\}$ has at least 3κ degrees of freedom. Similar to Lemma 3, given $\boldsymbol{G}$, $\{Map_{i,j}(Z_{i,j}^{k}, \boldsymbol{S}, \boldsymbol{G}) | (i,j) \in \{0,1\}^2 \setminus \{(1,1)\}, k \in [\kappa]\}$ has at least 3κ degrees of freedom. Meanwhile, $E_{0,0}, E_{0,1}, E_{1,0}$ have the same output wire label. Hence, $\{C_{i,j}^{k} | (i,j) \in \{0,1\}^2 \setminus \{(1,1)\}, k \in [\kappa]\}$ has κ degrees of freedom. Consequently, $m \geq 2\kappa$ when quasi-free-XOR is forbidden. □

5.3 Gate-Hiding Garbling Schemes

Gate-hiding garbling schemes, which hide the type of gates from the evaluator, play a role in private function evaluation [15,21]. The evaluator is only allowed to know the circuit topology, while all gate functions remain unknown. Of course, these garbling schemes need to support both AND and XOR gates, where the evaluator's actions do not differ. Some garbling schemes support more types of gates, e.g., constant gates. In our model, the process of garbling an arbitrary kind of gate has been well-defined, with or without quasi-free-XOR. Consequently, we aim to propose a lower bound for gate-hiding garbling schemes.

Schemes with Quasi-Free-XOR. First of all, we consider gate-hiding garbling schemes that support quasi-free-XOR constructions. Since garbling an AND gate requires at least 1.5κ ciphertexts, the lower bound of these garbling schemes must be at least 1.5κ. Then we check whether our proof for AND gates in Sect. 4.3 still holds when garbling an XOR gate. Note that *Claim* 4 does not hold on exposed XOR gates, because the evaluator $E_{i,j}$ is allowed to know the output wire label of $E_{\bar{i},\bar{j}}$. However, in the gate-hiding setting, *Claim* 4 still holds.

With quasi-free-XOR, we require that all the elements in $\{\boldsymbol{C}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}\}$ be the same for a given i. Since we only focus on the oracle response part in Sect. 4.3, the construction achieving 1.5κ ciphertexts also works on XOR gates. Hence, the lower bound in this case is $\frac{3}{2}\kappa$ bits, even if all types of gates are considered.

Schemes without Quasi-Free-XOR. We now assume that quasi-free-XOR is forbidden. For a given i, elements in $\{\boldsymbol{C}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}\}$ may be different. Without loss of generality, we assume $a = 0$ and $b = 0$. When garbling an AND gate, the garbler needs to ensure that for all $i \in [\kappa]$, $\boldsymbol{C}_{0,0}^{res,i} = \boldsymbol{C}_{0,1}^{res,i} = \boldsymbol{C}_{1,0}^{res,i}$. However, the garbler has to guarantee that the output labels of $E_{0,0}$ and $E_{1,1}$ are the same while the output labels of $E_{0,1}$ and $E_{1,0}$ are the same, when garbling an XOR gate. That is to say, $\boldsymbol{C}_{0,0}^{res,i} = \boldsymbol{C}_{1,1}^{res,i}$ and $\boldsymbol{C}_{0,1}^{res,i} = \boldsymbol{C}_{1,0}^{res,i}$. When garbling an AND (resp. XOR) gate, let $\mathcal{Z}_{\text{AND0}}$ (resp. $\mathcal{Z}_{\text{XOR0}}$) and $\mathcal{Z}_{\text{AND1}}$ (resp. $\mathcal{Z}_{\text{XOR1}}$) include vectors of C_0 and C_1.

Theorem 5. *In the model of Bitwise Linear Garbling Schemes, suppose quasi-free-XOR is forbidden. Under the gate-hiding assumption, the lower bound of m is 2κ.*

Proof. Note that $a = 0$ and $b = 0$.

1) *AND:* We add κ vectors $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ into the set $\mathcal{Z}_{\text{AND0}}$, to obtain rank κ.
   XOR: $\{\boldsymbol{Z}_{0,0}^{res,i} | i \in [\kappa]\}$ are put into $\mathcal{Z}_{\text{XOR0}}$, $rank(\mathcal{Z}_{\text{XOR0}}) = \kappa$.
2) *AND:* For the same reason as the proof in Sect. 5.2, $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$ are also added into $\mathcal{Z}_{\text{AND0}}$ and the rank of this set is now 2κ.
   XOR: However, to store $\{\boldsymbol{Z}_{0,1}^{res,i} | i \in [\kappa]\}$ when garbling an XOR gate, we need $\mathcal{Z}_{\text{XOR1}}$ instead of $\mathcal{Z}_{\text{XOR0}}$. $rank(\mathcal{Z}_{\text{XOR1}}) = \kappa$.
3) *AND:* After adding $\{\boldsymbol{Z}_{1,0}^{res,i} | i \in [\kappa]\}$ into $\mathcal{Z}_{\text{AND0}}$, $rank(\mathcal{Z}_{\text{AND0}}) = 3\kappa$. At least 2κ ciphertexts are necessary.
   XOR: The set $\mathcal{Z}_{\text{XOR1}}$ does not change anymore after containing $\{\boldsymbol{Z}_{1,0}^{res,i} | i \in [\kappa]\}$, so it requires κ ciphertexts.
4) *AND:* Finally, add $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$ into the new $\mathcal{Z}_{\text{AND1}}$. $\mathcal{Z}_{\text{AND1}}$ can be viewed as free in terms of ciphertexts.
   XOR: $\mathcal{Z}_{\text{XOR0}}$ containing $\{\boldsymbol{Z}_{1,1}^{res,i} | i \in [\kappa]\}$ is of rank 2κ. Hence, both $\mathcal{Z}_{\text{XOR0}}$ and $\mathcal{Z}_{\text{XOR1}}$ require κ ciphertexts.

We need 2κ ciphertexts to garble an AND gate. Even if $\mathcal{Z}_{\text{XOR0}}$ and $\mathcal{Z}_{\text{XOR1}}$ use the same κ ciphertexts, we still need 2κ ciphertexts to keep the gate function private.

However, one shall notice that garbling an AND gate may require ciphertexts in a different step from garbling an XOR gate. We need to ensure that the evaluator always has a view independent of gate functions. Roughly speaking, we ensure that in the evaluator's view, κ ciphertexts are always used in step 1) or 4), and the other κ ciphertexts are always used in step 2) or 3). After that, the 2κ-bit lower bound can be reached. $\square$

We omit our second model here, due to its similarity with the first model under the circumstances. It is easy to prove the 2κ-bit lower bound of ciphertexts.

6 Methods within the Specific Setting

There are still some methods [3,17,28] outside of the old model to be discussed. Note that we require the specific setting to analyze these methods. Before that, we propose another relevant claim of our first model, regardless of quasi-free-XOR:

- *Claim* 5: In the model of Bitwise Linear Garbling Schemes, suppose an AND gate is garbled. Given $i \in [\kappa]$, all the vectors in $\{\boldsymbol{V}^{prv,i}_{\alpha,\beta} | \alpha, \beta \in \{0,1\}\}$ are distinct.

Proof. Suppose it is violated by $\boldsymbol{V}^{prv,i}_{\alpha,\beta} = \boldsymbol{V}^{prv,i}_{\alpha',\beta'}$ where $(\alpha',\beta') \neq (\alpha,\beta)$. Considering different permute bits a and b, both $C_{\alpha,\beta} = C_{\alpha',\beta'}$ and $C_{\alpha,\beta} \neq C_{\alpha',\beta'}$ can happen. If $C_{\alpha,\beta} = C_{\alpha',\beta'}$, then $\boldsymbol{C}^{res,i}_{\alpha,\beta} = \boldsymbol{C}^{res,i}_{\alpha',\beta'}$ and $\boldsymbol{Z}^{res,i}_{\alpha,\beta} = \boldsymbol{Z}^{res,i}_{\alpha',\beta'}$. Without loss of generality, we suppose $\alpha = \alpha'$. We find that B_β and $B_{\beta'}$ can not be used to compute the output wire label, because $E_{\alpha,\beta}$ (resp. $E_{\alpha',\beta'}$) does not have $B_{\beta'}$ (resp. B_β). If B_β or $B_{\beta'}$ are used, then $E_{\alpha',\beta'}$ or $E_{\alpha,\beta}$ can not get the output. If $C_{\alpha,\beta} \neq C_{\alpha',\beta'}$, we still disallow $E_{\alpha,\beta}$ and $E_{\alpha',\beta'}$ to use B_β and $B_{\beta'}$. Otherwise, $E_{\alpha,\beta}$ and $E_{\alpha',\beta'}$ can realize that $C_{\alpha,\beta} \neq C_{\alpha',\beta'}$, and learn information about permute bits. Hence, $C^i_{\alpha,\beta} \oplus C^i_{\alpha',\beta'}$ is equal to a value computed by A_α. Apparently, this is insecure. $\square$

The Specific Setting. The above proof is potentially based on the setting of the "three-halves" garbling scheme: AND gates absorb upstream NOT gates [26]. We call this the general setting. There is another specific setting in which the AND gate here always realizes $g(x_a, x_b) = x_a \wedge x_b$.[7]

We focus on the proof of Theorem 3, where one might get an idea: *Can we try to reduce the rank of $\mathcal{Z}$ before step 3) ends?*

Given $i \in [\kappa]$, to make this idea take effect, we have to make sure $\boldsymbol{Z}^{res,i}_{0,1} = \boldsymbol{Z}^{res,i}_{1,0}$. Through *Claim* 5, we have to consider the specific setting to continue our discussion. [8]

[7] Precisely speaking, this setting only considers symmetric gates whose gate function satisfies $g(0,1) = g(1,0)$. For simplicity, we only consider AND gates. One can easily check that this consideration is general.

[8] We refer readers to [3,17,28] to understand how and why these methods work in this specific setting.

Suppose $A_1 = A_0 + d$ and $B_1 = B_0 + d$, $d \overset{\$}{\leftarrow} \{0,1\}^\kappa$. Then, $l_1(A_0, B_1) = A_0 + B_1$ and $l_2(A_1, B_0) = A_1 + B_0$. To ensure $\boldsymbol{Z}_{0,1}^{res,i} = \boldsymbol{Z}_{1,0}^{res,i}$, $E_{0,1}$ and $E_{1,0}$ both use $H(A_0 + B_1)$ to get the same output. Moreover, $E_{0,0}$ and $E_{1,1}$ use $H(A_0 + B_0)$ and $H(A_1 + B_1)$ to keep the same actions as $E_{0,1}$ and $E_{1,0}$. Note that $H(A_0 + B_0)$ and $H(A_1 + B_1)$ are associated with $\{E_{0,0}\}$ and $\{E_{1,1}\}$ respectively. Hence, it is easy that this idea does not break privacy.

The rank of $\mathcal{Z}$ is reduced to 2κ before step 3) ends. According to Lemma 4, this allows us to use κ ciphertexts. Meanwhile, the remaining κ vectors in step 4) do not seem to affect the number of ciphertexts, because $\boldsymbol{C}_{1,1}^{res,i}$ can be different from $\boldsymbol{C}_{0,0}^{res,i}$. Therefore, only κ ciphertexts are required on this single AND gate. Those known methods [3,17,28] are all based on this idea. The method of [3] seems not distinct, because it realizes this idea by exploiting two gates. The oracle response $H(A_i + B_j)$ is computed in the first addition-mod-3 gate, while ciphertexts are used in the second unary "projection" gate.

Clearly, this idea still supports quasi-free-XOR. However, output wire labels fail to keep this correlation, since $\boldsymbol{C}_{1,1}^{res,i}$ are different. If output wire labels (C_0, C_1) still keep this correlation, elements in $\{\boldsymbol{C}_{\alpha,\beta}^{res,i} | \alpha, \beta \in \{0,1\}\}$ are forced to be the same. This gate still requires 2κ ciphertexts. Hence, it becomes irrelevant when we restrict that the correlation of input wire labels must be maintained.

7 Discussion

There are several points we discuss here.

Privacy-Free Garbling. In [26], Rosulek and Roy challenged the possibility of achieving a communication cost of less than κ bits in privacy-free garbling [10], because the idea of the half-gates garbling schemes [30] points out κ bits are needed when only authenticity is guaranteed. However, less than κ bits is quite counterintuitive. In our models, this looks like there exist some output bits are garbled without ciphertexts. Given $a = 0$ and $b = 0$, suppose the outputs of $\{E_{0,0}, E_{0,1}, E_{1,0}\}$ are 0. As we emphasize in *Claim* 4, we can not make them obtain the same oracle responses without ciphertexts. Hence, privacy-free garbling still needs κ bits in our models.

Mapping with Rejcetion. We exploit an idea of [9] to deal with non-linear mapping. It is worth noting that another model dealing with mapping with rejection [1] is also considered in [9], but it seems that we do not mention such an idea before. This is because this idea can be included quite naturally and do not affect our lower bounds. According to the proof of Theorem 2, if we reduce the number of ciphertexts below the lower bound, some bits of the output wire label have a probability to be invalid, that is, mapping with rejection. To produce an entire valid output wire label, we increase the length of original output label to ensure enough valid bits. It is easy to check the number of ciphertexts matches lower bounds, when we increase the length of the output label.

Extensibility. Suppose the existence of malicious adversaries in 2PC, the idea of "three-halves" garbling scheme seems blocked. In the semi-honest setting, those control bits are sampled by using garbled truth tables (influenced by permute bits and absorbed NOT gates). The garbler can predict and arrange all the possibilities, so he can independently sample these bits. In the malicious setting however, both parties get involved in the garbling. It remains uncertain whether this idea can be extended to the malicious setting, and if so, at what cost. Till now, known authenticated garbling 2PC protocols against malicious adversaries are only compatible with the half-gates construction [7,8,16,27], rather than the "three-halves" construction. Hence, we are curious to see if our model can be extended into the malicious setting by appropriate limitation.

Meanwhile, we also question the feasibility of increasing the number of wire labels slightly, because we can manage to apply a further slicing. For example, we can use 6 $\frac{\kappa}{3}$-bit ciphertexts to garble a 3-input gate whose truth table is of even parity. Specifically, for the evaluator with three input wire labels (A_i, B_j, C_k) where $i, j, k \in \{0,1\}$, she uses $H(A_i \oplus B_j) \oplus H(B_j) \oplus H(C_k)$, $H(A_i \oplus C_k) \oplus H(A_i) \oplus H(B_j)$ and $H(B_j \oplus C_k) \oplus H(A_i) \oplus H(B_j)$ as the oracle responses of the left, middle and right parts of the output wire label. Unfortunately, this thought alone is impractical, because we only care odd parity.

References

1. Acharya, A., Ashur, T., Cohen, E., Hazay, C., Yanai, A.: A new approach to garbled circuits. Cryptology ePrint Archive, Paper 2021/739 (2021). https://eprint.iacr.org/2021/739
2. Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-Interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol 8441, pp. 387-404. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_22
3. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for Boolean and arithmetic circuits. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 565-577. ACM Press, October 2016
4. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC, pp. 503–513. ACM Press, May 1990
5. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, pp. 784–796. ACM Press, October 2012.
6. Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.-S: On the security of the "Free-XOR" Technique. In: Cramer, R. (eds.) TCC 2012. LNCS, vol 7194, pp. 39–53. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_3
7. Cui, H., Wang, X., Yang, K., Yu, Y.: Actively secure half-gates with minimum overhead under duplex networks. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 35–67. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30617-4_2
8. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Authenticated garbling from simple correlations. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol 13510, pp. 57-87. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_3

9. Fan, L., Lu, Z., Zhou, H.: Column-wise garbling, and how to go beyond the linear model. Cryptology ePrint Archive, Paper 2024/415 (2024). https://eprint.iacr.org/2024/415
10. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 191–219. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_7
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A. (eds.) 19th ACM STOC, pp. 218–229. ACM Press, May 1987
12. Heath, D., Kolesnikov, V.: Stacked garbling. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol 12171, pp. 763–792. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_27
13. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol 12107, pp. 569–598. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_19
14. Heath, D., Kolesnikov, V., Peceny, S.: MOTIF: (almost) free branching in GMW. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol 12493, pp. 3–30. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_1
15. Katz, J., Malka, L.: Constant-round private function evaluation with linear complexity. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 556–571. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_30
16. Katz, J., Ranellucci, S., Rosulek, M., Wang, X.: Optimizing authenticated garbling for faster secure two-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 365–391. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_13
17. Kempka, C., Kikuchi, R., Suzuki, K.: How to circumvent the two-ciphertext lower bound for linear garbling schemes. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 967–997. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_32
18. Kolesnikov, V.: Free IF: how to omit inactive branches and implement $\mathcal{S}$-universal garbled circuit (almost) for free. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 34–58. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_2
19. Kolesnikov, V., Mohassel, P., Rosulek, M.: FleXOR: flexible garbling for XOR gates that beats free-XOR. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 440–457. Springer, Heidelberg (2014) https://doi.org/10.1007/978-3-662-44381-1_25
20. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
21. Mohassel, P., Sadeghian, S.: How to hide circuits in MPC an efficient framework for private function evaluation. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 557–574. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_33
22. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce, New York, NY, USA, pp. 129–139. ACM (1999)

23. Patra, A., Ravi, D.: On the exact round complexity of secure three-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol 10992, pp. 425-458. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_15
24. Patra, A., Ravi, D.: Beyond honest majority: the round complexity of fair and robust multi-party computation. In: Galbraith, S., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol 11921, pp. 456-487. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_17
25. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (eds.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009) https://doi.org/10.1007/978-3-642-10366-7_15
26. Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 94–124. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_5
27. Wang, X., Ranellucci, S., Katz, J.: Authenticated garbling and efficient maliciously secure two-party computation. In: ACM CCS 2017, pp. 21–37. ACM Press (2017)
28. Wang, Y., Malluhi, Q.M.: Reducing garbled circuit size while preserving circuit gate privacy. Cryptology ePrint Archive, Report 2017/041 (2017). https://eprint.iacr.org/2017/041
29. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE Computer Society Press, November 1982
30. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole-reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8