Can We Beat Three Halves Lower Bound?
(Im)Possibility of Reducing Communication Cost for Garbled Circuits*

Chunghun Baek and Taechan Kim**

Samsung Research, Seoul, Korea
{chun.baek, taechan.kim}@samsung.com

Abstract. Recent improvements to garbled circuits are mainly focused on reducing their size. The state-of-the-art construction of Rosulek and Roy (Crypto 2021) requires 1.5κ bits for garbling AND gates in the free-XOR setting. This is below the previously proven lower bound 2κ in the linear garbling model of Zahur, Rosulek, and Evans (Eurocrypt 2015). Whether their construction is optimal in a more inclusive model than the linear garbling model still remains open.
This paper begins by providing a comprehensive model for a large class of practical garbling schemes and proves the lower bound for the size of the garbled AND gates in our model. We show that garbled AND gates require at least 1.5κ bits in our new model with the free-XOR setting.
It is remarkable to see that the construction by Rosulek and Roy is already optimal despite the fact that our model possibly captures any potential extension of their construction.

1 Introduction

Garbled Circuits (GC) are one of major techniques for secure two-party computation, which allows two mistrusting parties to jointly compute functions on their private inputs while revealing only the outputs of the functions and nothing else. Since their concept was first introduced by Yao [15], one line of recent research [2, 12, 10, 13, 9, 6, 16, 14] has been dedicated to reducing the size of the garbled circuit ciphertexts that should be sent from one party, the garbler, to the other party, the evaluator.

The current state-of-the-art construction for garbled circuits is due to Rosulek and Roy [14] (dubbed as RR21 throughout the paper), where they consider a gate-by-gate garbling of Boolean circuits expressed using XOR and AND gates. In their scheme, the size of the garbled AND gates is 1.5κ bits (κ is the security parameter), while no communication is required for XOR gates. Their result surpassed the previous lower bound (2κ bits for AND gates with free-XOR) for the size of garbled circuits, which is obtained in a model called *linear garbling* defined by Zahur, Rosulek, and Evans [16]. Their optimization was made possible

* In this paper, we prove the lower bound of linear garbling scheme. Concurrently to our work, Fan, Lu, and Zhou [4] also presented a lower bound proof.

** Corresponding author

by a new technique, called *slicing-and-dicing*, that is beyond the definition of the linear garbling model.

Thus arises the following natural question whether the recent result is optimal. As posed by [14]:

> *Is* 1.5κ *bits optimal for garbled AND gates in a more inclusive model than the one in [16]? . . . Conversely, can one do better – say,* $4\kappa/3$ *bits per AND gate?*

1.1 Our Contributions

In this paper, we answer the aforementioned question. We prove that 1.5κ bits is optimal for garbled AND gates in our new model under the free-XOR setting. It should be noted that our model successfully captures all existing techniques not only the ones covered by the linear garbling model, but also the RR21 construction. Moreover, it captures any possible potential extension of RR21. To be precise, we summarize our contributions as follows:

- **Define a New Model for Linear Garbling Schemes.** The original definition of linear garbling model assumes that parties can only use symmetric-key primitives and linear operations. For practicality, we still use the same assumption. However, the original model implicitly assumes that an output wire label is computed by applying only *one* linear combination. Motivated by RR21's construction, we extend this assumption so that wire labels can be derived from applying *multiple* linear combinations.
  Based on this idea, we define a new model for garbling schemes. We believe our model covers a broad class of garbling scheme that one can construct using only symmetric-key primitives and linear operations.
- **Prove a Lower Bound.** We prove a lower bound of garbling schemes in our model. Precisely, we show that any secure garbled AND gates should have size of at least 1.5κ bits under the free-XOR setting [10]. Otherwise, garbled circuits might leak some information on private inputs other than the function's output. It is interesting to see that RR21's construction is already optimal in our sense, although our model encompasses any possible potential extension of their construction.

Our proof is purely based on algebraic techniques. The following items present the technical flow of our proof.

- **Formalize Garbling Equations.** Our main idea is to write garbling schemes as algebraic equations of two variables related to the evaluator's view, which we call *garbling equations*. This makes the representation of the garbling scheme simpler. Indeed, the linear-algebraic representation of [14] can be considered as a collection of linear equations each of which is obtained by evaluating our garbling equations at the evaluator's possible choices of inputs.

- **Find Coefficients.** Having formalized the garbling equations, we observe that the construction of a garbling scheme is simply reduced to finding all coefficients in the garbling equations so that the equality holds for all possible inputs of the evaluator. We provide several algebraic conditions for those coefficients and find solutions satisfying such conditions. We note that all existing constructions can be understood as particular solutions for the garbling equations in the above sense.
- **Check Privacy Leakage.** Once we have found the solutions for the garbling equations, it is necessary to check whether the corresponding garbling scheme is secure. Interestingly, we observe that the evaluator can infer information on the private inputs whenever the number of ciphertexts is smaller than $\frac{3}{2}s$. Here s is the number of slices and each of the ciphertexts is of length κ/s bits. Thus, any garbled AND gates of size $< 1.5\kappa$ bits violate the privacy property.

Organizations. The remainder of the paper is organized as follows. In Section 2, we review the formalism of garbling schemes and the definition of the linear garbling model. We describe our algebraic perspective of garbling scheme and interpret the existing garbling scheme from our perspective in Section 3. In Section 4, we present a new model that extends the linear garbling model. We then provide a notion of the garbling equation and prove a lower bound on garbled circuits in our garbling model in Section 5. Finally, we conclude by suggesting some open questions in Section 6.

2 Preliminaries

Throughout the paper, we will work over finite fields $\mathbb{K}$ of characteristic 2 and the bivariate polynomial ring $\mathbb{K}[x, y]$. We write $x + y$ / xy for Boolean operations XOR/AND of $x, y \in \mathbb{F}_2$, respectively. We denote a vector and its entries as $\overrightarrow{v} = (v_1, \ldots, v_n)$. Matrices are written in the bold capital characters such as $\mathbf{M}$.

2.1 Garbling Schemes

We use the garbling scheme abstraction introduced by Bellare, Hoang, and Rogaway [3]. In particular, as in [16], we concentrate on garbling circuits rather than garbling any form of computation. A garbling scheme consists of the following algorithms:

- Gb: On input 1^κ and a Boolean circuit f, outputs (F, e, d), where F is a *garbled circuit*, e is encoding information, d is decoding information.
- En: On input (e, x), where e is as above and x is an input suitable for f, outputs a *garbled input* X.
- Ev: On input (F, X), outputs a *garbled output* Y.
- De: On input (d, Y), returns an output y.

Correctness. A garbling scheme defined as above is *correct*, if $(F, e, d) \leftarrow \mathsf{Gb}(1^\kappa, f)$, $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x)$ holds all but negligible probability.

Privacy. Informally, we say a garbling scheme satisfies *privacy*, if (F, X, d) reveals no information about x other than $f(x)$. In our discussion, it is enough to consider the privacy property. For further details on other security properties such as *obliviousness* and *authenticity*, refer to Bellare, Hoang, and Rogaway [3].

2.2 Linear Garbling Schemes

Zahur, Rosulek, and Evans [16] defined a model of linear garbling to capture all existing practical techniques for garbling a single AND gate at the time. A garbling scheme for an AND gate is called *linear* if it satisfies the following form:

- Gb: Parameterized by integers m, n and r and $(m+n)$-dimensional vectors $\overrightarrow{A^0}, \overrightarrow{A^1}, \overrightarrow{B^0}, \overrightarrow{B^1}, \{\overrightarrow{C}^0_{\alpha,\beta} : \alpha, \beta \in \{0,1\}\}, \{\overrightarrow{C}^1_{\alpha,\beta} : \alpha, \beta \in \{0,1\}\}$ and $\{\overrightarrow{g}_{\alpha,\beta,i} : \alpha, \beta \in \{0,1\}, i \in \{1,\ldots,r\}\}$.
  1. (Choose random values) Choose $R_i \leftarrow \mathbb{F}_{2^\kappa}$ for $i = 1, \ldots, m$
  2. (Random oracle queries) Make n distinct queries to a random oracle (which can be chosen as a deterministic function of the R_i values). Let $H_1, \ldots, H_n$ denote the responses to these queries. Define $\overrightarrow{S} := (R_1, \ldots, R_m, H_1, \ldots, H_n)^t$.
  3. (Choose masking bits) Choose random masking bits $\alpha, \beta \leftarrow \{0,1\}$.
  4. (Input wire labels) For $i = 0, 1$, compute $A^i = \overrightarrow{A^i} \cdot \overrightarrow{S}$ and $B^i = \overrightarrow{B^i} \cdot \overrightarrow{S}$. Then $(A^0 \parallel 0, A^1 \parallel 1)$ and $(B^0 \parallel 0, B^1 \parallel 1)$ are taken as the input wire labels with A^α and B^β corresponding to **false**. Here, the superscripts denote the public color bits.
  5. (Output wire labels) For $i = 0, 1$, compute $C^i = \overrightarrow{C}^i_{\alpha,\beta} \cdot \overrightarrow{S}$. Here, C^0 corresponds to **false**.
  6. (Ciphertexts) For $i \in \{1, \ldots, r\}$, compute $G_i = \overrightarrow{g}_{\alpha,\beta,i} \cdot \overrightarrow{S}$. Then, $(G_1, \ldots, G_r)$ comprise the garbled circuit.
- En: On input $a, b \in \{0,1\}$, set $x = a + \alpha$ and $y = b + \beta$, where α and β are the masking bits chosen above. Output $A^x \parallel x$ and $B^y \parallel y$.
- Ev: Parameterized by integer n and binary vectors $\{\overrightarrow{\forall}_{x,y} : x, y \in \{0,1\}\}$, where each vector is of length $n + r + 2$.
  1. (Inputs) The input are wire labels $A^x \parallel x$ and $B^y \parallel y$, tagged with their corresponding color bits, and the garbled circuit $G_1, \ldots, G_r$.
  2. (Random oracle queries) Make n distinct oracle queries to the random oracle (chosen as a deterministic function of the input wire labels). Denote the responses to the queries by $H'_1, \ldots, H'_n$. Define $\overrightarrow{T} := (A^x, B^y, H'_1, \ldots, H'_n, G_1, \ldots, G_r)^t$.
  3. Output the value $\overrightarrow{\forall}_{x,y} \cdot \overrightarrow{T}$.

Zahur, Rosulek, and Evans proved the lower bound in the model of linear garbling schemes.

Theorem 1. *[16, Theorem 3.] Every secure linear garbling scheme for AND gates should have $r \geq 2$. In other words, the garbled gate consists of at least 2κ bits.*

3 An Algebraic Perspective of Garbling Schemes

In this section, we present our algebraic perspective of garbling schemes. It simplifies our understanding on how the previous garbling schemes were constructed and have been improved over the years. Based on this idea, in the forthcoming section, we investigate whether we can further improve the communication costs for garbling schemes.

To begin with, in Section 3.1 we introduce the notion of a truth table of a Boolean function. This notion will be useful to understand our algebraic interpretation of garbling schemes in Section 3.2.

3.1 Truth Table and Boolean Functions

A truth table of a n-variable Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as follows.

Definition 1 (Truth Table). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function. The truth table for f, denoted by $\mathcal{T}(f)$, is a 2^n-bit vector such that:*

$$\mathcal{T}(f) := (f(0,\ldots,0), f(0,\ldots,1), \ldots, f(1,\ldots,1))^\top \in \mathbb{F}_2^{2^n}.$$

In other words, the k-th entry of $\mathcal{T}(f)$ is $f(k_0,\ldots,k_{n-1})$, where $(k_0 k_1 \ldots k_{n-1})_2$ is a binary representation of an integer $k = \sum_{i=0}^{n-1} k_i 2^{n-1-i} \in \{0,\ldots,2^n-1\}$.

As an example, let us consider a bivariate Boolean function $\mathbf{x} : \mathbb{F}_2^2 \to \mathbb{F}_2$ defined as $\mathbf{x}(x,y) = x$, i.e. $\mathbf{x}$ is a projection function that maps to the first coordinate. By the definition, $\mathcal{T}(\mathbf{x})$ is a vector in $\mathbb{F}_2^4$ given by $\mathcal{T}(\mathbf{x}) = (\mathbf{x}(0,0), \mathbf{x}(0,1), \mathbf{x}(1,0), \mathbf{x}(1,1)) = (0,0,1,1)$.

To put it differently, the operator $\mathcal{T}$ maps a n-variate Boolean function f to a unique 2^n-dimensional binary vector. We also observe that $\mathcal{T}$ is homomorphic, i.e. $\mathcal{T}(f+g) = \mathcal{T}(f) + \mathcal{T}(g)$ for Boolean functions f and g.

Conversely, given any 2^n-bit vector $\overrightarrow{v}$, one can find a unique n-variate Boolean polynomial f such that whose truth table $\mathcal{T}(f)$ is the same as $\overrightarrow{v}$.

Definition 2 (Sum-of-Product). *Let $\overrightarrow{v} = (v_0,\ldots,v_{2^n-1})$ be a vector in $\mathbb{F}_2^{2^n}$. The sum-of-product (SoP) expression for $\overrightarrow{v}$, denoted by $\mathcal{F}_{\overrightarrow{v}} : \mathbb{F}_2^n \to \mathbb{F}_2$, is a n-variate Boolean function such that:*

$$\mathcal{F}_{\overrightarrow{v}}(x_0,\ldots,x_{n-1}) := \sum_{k=0}^{2^n-1} v_k \prod_{i=0}^{n-1} (x_i + k_i + 1),$$

where $(k_0 k_1 \ldots k_{n-1})_2$ is a binary representation of an integer $k = \sum_{i=0}^{n-1} k_i 2^{n-1-i} \in \{0,\ldots,2^n-1\}$.

By the definition, we can check that $\mathcal{T}(\mathcal{F}_{\overrightarrow{v}}) = \overrightarrow{v}$. For instance, let us take an example of $\overrightarrow{v} = (0,0,1,1)^\top$. Consider a bivariate Boolean function $f(x,y)$ such that $\mathcal{T}(f) = \overrightarrow{v}$. The sum-of-product expression of the truth table $\overrightarrow{v}$ provides the Boolean function f such that $f(x,y) = x(y+1) + xy = x$ (note that the Boolean function $x(y+1)$ outputs 1 iff $(x,y) = (0,1)$ and xy outputs 1 iff $(x,y) = (1,1)$). In other words, we have $\mathcal{F}_{\overrightarrow{v}}(x,y) = \mathbf{x}(x,y) = x$ and $\mathcal{T}(\mathcal{F}_{\overrightarrow{v}}) = \mathcal{T}(\mathbf{x}) = \overrightarrow{v}$.

Throughout the paper, we are mainly interested in the case of $n = 2$, i.e. bivariate Boolean polynomials. Denote $\mathbb{F}_2[x,y]$ by the bivariate polynomial ring over $\mathbb{F}_2$ with the variables x and y. There exists one-to-one correspondence between the quotient ring $\mathbb{F}_2[x,y]/\langle x^2+x, y^2+y\rangle$ and $\mathbb{F}_2^4$ by the operators $\mathcal{T}$ and $\mathcal{F}$.[1] We provide a list of examples in Table 1 presenting this one-to-one correspondence. Here, $\mathcal{T}(f) = \overrightarrow{v}$ and $\mathcal{F}_{\overrightarrow{v}} = f$ for $f \in \mathbb{F}_2[x,y]$ and $\overrightarrow{v} \in \mathbb{F}_2^4$.

We remark one more interesting property of the operator $\mathcal{F}$. Let us consider the Hadamard product $\overrightarrow{v} \circ \overrightarrow{w}$, the element-wise product. We assume $\overrightarrow{v}$ and $\overrightarrow{w}$ is of the same dimension. Then, interestingly, we observe that $\mathcal{F}_{\overrightarrow{v}\circ\overrightarrow{w}} = \mathcal{F}_{\overrightarrow{v}} \cdot \mathcal{F}_{\overrightarrow{w}}$. For instance, we see that $(0,0,1,1) \circ (0,1,0,1) = (0,0,0,1)$ and $\mathcal{F}_{(0,0,0,1)^\top} = \mathcal{F}_{(0,0,1,1)^\top} \cdot \mathcal{F}_{(0,1,0,1)^\top}$. It is not hard to verify this property, so we leave this to the readers.

f	$\overrightarrow{v}^\top$	f	$\overrightarrow{v}^\top$	f	$\overrightarrow{v}^\top$	f	$\overrightarrow{v}^\top$
0	$(0,0,0,0)$	1	$(1,1,1,1)$	xy	$(0,0,0,1)$	$xy+1$	$(1,1,1,0)$
x	$(0,0,1,1)$	$x+1$	$(1,1,0,0)$	$x(y+1)$	$(0,0,1,0)$	$x(y+1)+1$	$(1,1,0,1)$
y	$(0,1,0,1)$	$y+1$	$(1,0,1,0)$	$(x+1)y$	$(0,1,0,0)$	$(x+1)y+1$	$(1,0,1,1)$
$x+y$	$(0,1,1,0)$	$x+y+1$	$(1,0,0,1)$	$(x+1)(y+1)$	$(1,0,0,0)$	$(x+1)(y+1)+1$	$(0,1,1,1)$

Table 1: Examples of Boolean functions and their truth table when $n = 2$. It satisfies $\mathcal{T}(f) = \overrightarrow{v}$ and $\mathcal{F}_{\overrightarrow{v}} = f$.

3.2 From Linear-Algebraic View to Algebraic Perspective

In this section, we briefly review the construction of the existing garbling schemes together with RR21's linear algebraic representation. Then we re-interpret RR21's linear algebraic representation from our algebraic perspective using the truth table and the SoP expression explained in Section 3.1. Once it has been done, we will observe that constructing a garbling scheme reduces to find suitable algebraic equations.

Throughout this paper, let κ be the security parameter. We consider garbling an AND gate with input wires a and b. To garble the gate, the garbler chooses two κ-bit strings as wire labels per each input wire, say, A^0 and A^1 (resp. B^0

[1] This choice of the quotient ring is natural, since we only consider the case that the variables x and y take the values at $\mathbb{F}_2$. Throughout the paper, polynomials in $\mathbb{F}_2[x,y]$ are regarded as the elements in this quotient ring.

and B^1) are wire labels for the input wire a (resp. b). The garbler secretly knows which wire label corresponds to **false** of each input wire. To be precise, the wire labels A^α and B^β correspond to **false**, where the values of the bits α and β are only known to the garbler. We shall call these two bits the *permute bits*.

We also apply the point-and-permute technique [2]. In other words, when the evaluator holds exactly one wire label for each wire, say A^x and B^y for some $x, y \in \mathbb{F}_2$, we assume that the superscripts x and y are publicly known to her. These bits are often referred to as *color bits*. Typically, the least significant bit of the wire label is used as its color bit.

We often use both (x, y) and (i, j) to indicate the pair of the color bits. In particular, (x, y) is mainly used to denote unspecified color bits and plays as variables. On the other hand, (i, j) is especially used as a specific choice of the color bits.

From now on, we will consider the free-XOR setting [10]. In this setting, the garbler chooses a common global bit string Δ and the wire labels are chosen so that $A^0 + A^1 = B^0 + B^1 = \Delta$ (recall that the addition is over $\mathbb{F}_2$, so it is the same as the XOR operation). The value of Δ should be kept secret by the garbler.

In the following, we assume that H is a random oracle of the output length κ unless specified otherwise.

3.2.1 Yao's Garbled Circuit

In Yao's garbled circuit construction, the garbler generates 4 ciphertexts, say $G_{0,0}, G_{0,1}, G_{1,0}, G_{1,1}$, corresponding to each of the evaluator's color bits combinations.

When the evaluator holds the wire labels A^x and B^y for some $x, y \in \mathbb{F}_2$, she computes the output wire label by decrypting the (x, y)-th ciphertext as $G_{x,y} + H(A^x, B^y)$. In order for the evaluator to correctly compute the output wire label for the AND gate, the garbler sets an output wire label C that corresponds to **false** so that the evaluator can output $C + \Delta$ (that corresponds to **true**) if and only if she has the wire labels $A^{\alpha+1}$ and $B^{\beta+1}$. In short, it should satisfy the following:

$$C = G_{x,y} + H(A^x, B^y) \text{ if and only if } (x, y) \neq (\alpha + 1, \beta + 1);$$

and

$$C + \Delta = G_{x,y} + H(A^x, B^y) \text{ if and only if } (x, y) = (\alpha + 1, \beta + 1).$$

RR21's Linear Algebraic Perspective. In RR21, they rearranged the above equations into a linear-algebraic relation as follows.

$$\begin{bmatrix} 1&1&0&0&0 \\ 1&0&1&0&0 \\ 1&0&0&1&0 \\ 1&0&0&0&1 \end{bmatrix} \begin{bmatrix} C \\ G_{0,0} \\ G_{0,1} \\ G_{1,0} \\ G_{1,1} \end{bmatrix} = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} \begin{bmatrix} H(A^0, B^0) \\ H(A^0, B^1) \\ H(A^1, B^0) \\ H(A^1, B^1) \end{bmatrix} + \overrightarrow{t}\Delta. \tag{1}$$

Here, $\overrightarrow{t} = (t_{0,0}, t_{0,1}, t_{1,0}, t_{1,1})^\top$ where $t_{x,y} = 1$ if and only if $(x, y) = (\alpha+1, \beta+1)$ and $t_{x,y} = 0$ otherwise.

In the above equation, the values Δ and $H(A^x, B^y)$ on the right hand side are predetermined by the input values. Depending on these values, the garbler has to compute the values on the left hand side, the output wire label C and the ciphertexts $G_{x,y}$'s, so that the above equation holds.

Our Algebraic Perspective. In this paragraph, we present our idea of representing Yao's construction from our algebraic perspective. From now on, we consider the evaluator's color bits x and y as the variables in $\mathbb{F}_2$.

Since the function $(x + \alpha)(y + \beta) \in \mathbb{F}_2[x, y]$ parameterized by α and $\beta \in \mathbb{F}_2$ outputs 1 if and only if $(x, y) = (\alpha + 1, \beta + 1)$, we can rewrite the equation of Yao's garbled circuit as

$$C + (x + \alpha)(y + \beta)\Delta = G_{x,y} + H(A^x, B^y).$$

By the similar idea, we can rewrite $G_{x,y}$ and $H(A^x, B^y)$ as quadratic polynomials over $\mathbb{F}_{2^\kappa}$ such that $G_{x,y} = (x + 1)(y + 1)G_{0,0} + (x + 1)yG_{0,1} + x(y + 1)G_{1,0} + xyG_{1,1}$ and $H(A^x, B^y) = (x + 1)(y + 1)H(A^0, B^0) + (x + 1)yH(A^0, B^1) + x(y + 1)H(A^1, B^0) + xyH(A^1, B^1)$.

Let us set $\mathbf{M} = \begin{bmatrix}(x+1)(y+1) & (x+1)y & x(y+1) & xy\end{bmatrix}$, an (1×4)-matrix over $\mathbb{F}_2[x, y]$. Summarizing and rearranging all above, Yao's construction can be represented by a quadratic equation over $\mathbb{F}_{2^\kappa}$ as follows:

$$C + \mathbf{M}\overrightarrow{G} = \mathbf{M}\overrightarrow{H} + (x + \alpha)(y + \beta)\Delta, \tag{2}$$

where the coefficient vectors are defined as $\overrightarrow{G} = (G_{0,0}, G_{0,1}, G_{1,0}, G_{1,1})^\top$ and $\overrightarrow{H} = (H(A^0, B^0), H(A^0, B^1), H(A^1, B^0), H(A^1, B^1))^\top$.

Then the garbler's task now becomes equivalent to choosing the coefficients, C and $G_{i,j}$'s, in the left-hand side in a way that *the identity holds in the equation* (2) *for arbitrary choices of* $(x, y) \in \mathbb{F}_2 \times \mathbb{F}_2$, where the coefficients on the right-hand side are already determined by the input values.

For instance, comparing the constant term in both sides yields $C + G_{0,0} = H(A^0, B^0) + \alpha\beta\Delta$. Note that this relation is the same as the top row in (1). Moreover, comparing x-coefficient in both sides, we have $G_{0,0} + G_{1,0} = H(A^0, B^0) + H(A^1, B^0) + \beta\Delta$. Summing this relation to the former (the one from the constant term) provides the relation in the third row in (1). In a similar way, one can check that Equation (1) and (2) are equivalent.

It is notable that there are 5 unknowns, C and $G_{i,j}$'s, to be determined whereas there are 4 relations from each coefficients of $1, x, y$ and xy. This system of linear equation is underdetermined, and hence there are infinitely many solutions and it is easy to solve.

Relationships between the Two Representations. It is interesting to see how our algebraic representation relates to RR21's linear algebraic representation. From its construction, it is obvious that the $(2i + j + 1)$-th row of Equation (1)

is obtained by evaluating Equation (2) at $(x, y) = (i, j)$ for $i, j \in \{0, 1\}$. Here, be aware that the integer $2i + j + 1 \in \{1, \ldots, 4\}$ is computed by viewing i and j as integers.

On the other hand, applying the operator $\mathcal{F}$ (see Section 3.1) to each column of the binary matrices in Equation (1), we can derive Equation (2) from Equation (1). For instance, we have

$$\begin{bmatrix} 1&1&0&0&0 \\ 1&0&1&0&0 \\ 1&0&0&1&0 \\ 1&0&0&0&1 \end{bmatrix} \xrightarrow{\mathcal{F}} \begin{bmatrix} 1 & (x+1)(y+1) & (x+1)y & x(y+1) & xy \end{bmatrix} = [1 \mid \mathbf{M}].^{2}$$

By the definition of $\overrightarrow{t}$, one can also check that $\mathcal{F}_{\overrightarrow{t}} = (x + \alpha)(y + \beta)$. Plugging them into Equation (1) leads us to have Equation (2).

In a nutshell, applying $\mathcal{T}$ to the linear-algebraic representation gives our algebraic representation, and applying $\mathcal{F}$ to our algebraic representation translates to the linear-algebraic representation.

This idea is seemingly very simple, but novel. As we shall see below, it shall help us to understand other complicated constructions, such as RR21, more easily.

3.2.2 Row Reduction Techniques

Row Reduction techniques reduce the number of ciphertexts from 4 to 3 by forcing $G_{0,0} = \overrightarrow{0^\kappa}$ (i.e. all κ bits are zero). It is equivalent to set the output wire label as $C = H(A^0, B^0) + \alpha\beta\Delta$ in Yao's construction.

RR21's Linear Algebraic Perspective. From RR21's perspective, the linear relation can be written as follows,

$$\begin{bmatrix} 1&0&0&0 \\ 1&1&0&0 \\ 1&0&1&0 \\ 1&0&0&1 \end{bmatrix} \begin{bmatrix} C \\ G_{0,1} \\ G_{1,0} \\ G_{1,1} \end{bmatrix} = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} \begin{bmatrix} H(A^0, B^0) \\ H(A^0, B^1) \\ H(A^1, B^0) \\ H(A^1, B^1) \end{bmatrix} + \overrightarrow{t}\,\Delta.$$

Our Algebraic Perspective. As before, applying $\mathcal{F}$ to the linear-algebraic equation provides us

$$C + (x+1)yG_{0,1} + x(y+1)G_{1,0} + xyG_{1,1} = \mathbf{M}\overrightarrow{H} + (x+\alpha)(y+\beta)\Delta,$$

where $\mathbf{M}$ and $\overrightarrow{H}$ are the same as in Yao's construction.

If we explain the row reduction technique from our algebraic perspective, we see that 4 unknowns $C, G_{0,1}, G_{1,0}$ and $G_{1,1}$ are determined using 4 relations by comparing the coefficients of $1, x, y$ and xy. Recall that in Yao's construction we had an underdetermined system of 4 relations in 5 unknowns, where one of five unknowns could be chosen freely. The row reduction technique, however, reduces the number of the ciphertexts by removing this redundancy.

[2] Recall that we focus on bivariate case as mentioned in Section 3.1. So, we have $\mathcal{F}_{(1,1,1,1)^\top} = 1$, $\mathcal{F}_{(1,0,0,0)^\top} = (x+1)(y+1)$, and so on (see Table 1).

3.2.3 Half-Gate Garbling Scheme

The half-gate garbling scheme [16] reduces the number of ciphertexts to 2. Describing the scheme in short, on input A^i and B^j, the evaluator obtains the output label by computing

$$C + (i+\alpha)(j+\beta)\Delta = (iG_0 + jG_1) + H(A^i) + H(B^j) + jA^i. \tag{3}$$

As before, α and β are the permute bits and $(i+\alpha)(j+\beta)$ is the actual value of the output by the AND gate.

RR21's Linear Algebraic Perspective. In [14], they represented this as follows:

$$\begin{bmatrix} 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{bmatrix} \begin{bmatrix} C \\ G_0 \\ G_1 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 1\,0\,0\,1 \\ 0\,1\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix} \begin{bmatrix} H(A^0) \\ H(A^1) \\ H(B^0) \\ H(B^1) \end{bmatrix} + \begin{bmatrix} 0\,0 \\ 1\,0 \\ 0\,0 \\ 1\,1 \end{bmatrix} \begin{bmatrix} A^0 \\ \Delta \end{bmatrix} + \overrightarrow{t}\,\Delta, \tag{4}$$

where $\overrightarrow{t}$ is the same as Equation (1).

Our Algebraic Perspective. Again, we apply $\mathcal{F}$ to Equation (4). From Section 3.1, recall that $\mathcal{F}_{(1,1,1,1)^\top} = 1$, $\mathcal{F}_{(0,0,1,1)^\top} = x$, $\mathcal{F}_{(0,1,0,1)^\top} = y$, and $\mathcal{F}_{(0,0,0,1)^\top} = xy$.

Plugging these into (4), we have

$$\begin{bmatrix} 1\ x\ y \end{bmatrix} \begin{bmatrix} C \\ G_0 \\ G_1 \end{bmatrix} = \begin{bmatrix} x+1\ x\ y+1\ y \end{bmatrix} \begin{bmatrix} H(A^0) \\ H(A^1) \\ H(B^0) \\ H(B^1) \end{bmatrix} + \begin{bmatrix} y\ xy \end{bmatrix} \begin{bmatrix} A^0 \\ \Delta \end{bmatrix} + \mathcal{F}_{\overrightarrow{t}}\Delta.$$

Let us define $\mathbf{M} = \begin{bmatrix} x+1\ x\ y+1\ y \end{bmatrix}$ and $\overrightarrow{H} = (H(A^0), H(A^1), H(B^0), H(B^1))^\top$. Then the above equation can be rewritten as

$$C + xG_0 + yG_1 = \mathbf{M}\overrightarrow{H} + y(A^0 + x\Delta) + (x+\alpha)(y+\beta)\Delta. \tag{5}$$

It is remarkable to see that this is exactly the same representation as Equation (3).

Let us explain this equation from our algebraic perspective. As before, the garbler should choose C, G_0, and G_1 by comparing the coefficients in both sides. In addition, note that the only quadratic term, $xy\Delta$, in the right-hand side is cancelled out, so both sides are *linear* polynomials in x and y over $\mathbb{F}_{2^\kappa}$. This is considerably crucial in reducing the number of ciphertexts: the equation is only linear, so there are only 3 coefficients to consider (1, x, and y), whereas the previous constructions should consider 4 coefficients including the quadratic term xy.

We observe that the use of random oracle queries of the form $H(A^i)$ and $H(B^j)$ instead of $H(A^i, B^j)$ was at the core of this improvement. Due to this change, one can observe that $\mathbf{M}\overrightarrow{H}$ contains only linear polynomials. This reduces

the number of coefficients to be determined by the garbler, thus reducing the number of ciphertexts.

However, as one might observe, there always exists the quadratic polynomial $(x+\alpha)(y+\beta)\Delta$ whenever we garble the AND gate. In order to only consider linear terms, one might attempt to remove the quadratic term $xy\Delta$. Indeed, in the half-gate scheme, this term was cancelled out by introducing a term $yA^x = yA^0 + xy\Delta$. Note that this term depends only on the color bits x and y so that it is computable by the evaluator.

3.2.4 RR21's Garbling Scheme

In this subsection, we review the RR21's construction which reduces the size of the ciphertexts to $1.5\kappa + O(1)$ bits. Roughly speaking, they constructed a scheme with 3 ciphertexts each of which is of $\kappa/2$ bits.

Their idea is based on a technique called *slicing-and-dicing*. The slicing technique refers to an idea that slices wire labels into halves and use them to compute each half of the output label. For instance, in their scheme, the input labels are written as $A^i = \left(A^i_L \| A^i_R\right)$ and $B^j = \left(B^j_L \| B^j_R\right)$, where each halves are of $\kappa/2$ bits. As before, their construction assumes the free-XOR setting, so $A^0 + A^1 = B^0 + B^1 = \Delta$, where $\Delta = \left(\Delta_L \| \Delta_R\right)$. They also use a random oracle H whose range is the set of $\kappa/2$-bit strings, i.e. $H : \{0,1\}^* \to \{0,1\}^{\kappa/2}$. Then, each of C_L and C_R, the left and right half of the output label, is computed using ciphertexts of bit length $\kappa/2$, random oracle queries using A^i and B^j, and a linear function of A^i_L, A^i_R, B^j_L and B^j_R.

Based on this idea, the authors construct a system of linear equations similar to Equation (1) and (4) so that C_L or C_R can be represented as linear combinations using each halves of input labels, their random oracle queried values and ciphertexts. In order to achieve a non-trivial improvement, they introduced to use random oracle queries on inputs of the form $A^i + B^j$ besides A^i and B^j. Precisely, they used the random oracles of the form of $H(A^i)$, $H(B^j)$, and $H(A^i + B^j)$.[3] Then, on input A^i and B^j, the construction lets the evaluator compute the left/right half of the output label for (i,j) as

$$C_L + (i+\alpha)(j+\beta)\Delta_L = H(A^i) + H(A^i + B^j) + \cdots,$$
$$C_R + (i+\alpha)(j+\beta)\Delta_R = H(B^j) + H(A^i + B^j) + \cdots.$$

RR21's Linear Algebraic Perspective. In the dotted part of the above equations, there must be linear combinations of the ciphertexts and the wire labels. RR21 proposed a construction using three ciphertexts G_0, G_1, and G_2. Similarly as before, rearranging the equations so that only the values to be determined by the garbler are located on the left-hand side, the construction can be re-written in a linear-algebraic form into Equation (6).

Here, the $(2i+j+1)$-th row of Equation (6) represents the equation for the left half of the output label when the color bits combination is (i,j). Similarly,

[3] With the free-XOR setting, recall that $H(A^0 + B^0) = H(A^1 + B^1)$ and $H(A^0 + B^1) = H(A^1 + B^0)$.

the $(2i+j+5)$-th row is for the right-half of the label when the color bits are (i,j). Let us define $\overrightarrow{t} = (t_{0,0}, t_{0,1}, t_{1,0}, t_{1,1})$ as before. It can be written[4]

$$\left[\begin{array}{c|c} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,1 \\ 1\,0\,1\,0\,1 \\ 1\,0\,1\,0\,0 \\ \hline 0\,1\,0\,0\,0 \\ 0\,1\,0\,1\,1 \\ 0\,1\,0\,0\,1 \\ 0\,1\,0\,1\,0 \end{array}\right] \begin{bmatrix} C_L \\ C_R \\ G_0 \\ G_1 \\ G_2 \end{bmatrix} = \left[\begin{array}{c|c} 1\,0\,0\,0\,1\,0 \\ 1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,1 \\ 0\,1\,0\,0\,1\,0 \\ \hline 0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,1\,1\,0 \end{array}\right] \begin{bmatrix} H(A^0) \\ H(A^1) \\ H(B^0) \\ H(B^1) \\ H(A^0+B^0) \\ H(A^0+B^1) \end{bmatrix} + \mathbf{R}_{lin} \begin{bmatrix} A^0_L \\ A^0_R \\ B^0_L \\ B^0_R \\ \Delta_L \\ \Delta_R \end{bmatrix} + \begin{bmatrix} \overrightarrow{t} & \mathbf{0} \\ \mathbf{0} & \overrightarrow{t} \end{bmatrix} \begin{bmatrix} \Delta_L \\ \Delta_R \end{bmatrix}, \quad (6)$$

where the last term is the same as $\left(t_{0,0}\Delta_L, \ldots, t_{1,1}\Delta_L \mid t_{0,0}\Delta_R, \ldots, t_{1,1}\Delta_R\right)^\top$.

Denote the 8×6 binary matrix on the right-hand side by $\mathbf{M}_{lin}$ and the 8×5 binary matrix on the left-hand side by $\mathbf{V}_{lin}$. The matrix $\mathbf{M}_{lin}$ is directly decided by which type of oracle response is to be used. Once it has been settled, [14] observed that the columns of $\mathbf{M}_{lin}$ and $\mathbf{V}_{lin}$ should span the same column space in order to have the equality. This is why they set $\mathbf{V}_{lin}$ so that its columns consist of a basis of the column space of $\mathbf{M}_{lin}$. Since $\mathbf{M}_{lin}$ is of rank 5 by its construction, $\mathbf{V}_{lin}$ has 5 columns resulting in having 3 ciphertexts.

Then the work by [14] is mainly devoted to find out $\mathbf{R}_{lin}$. Observe that $\mathbf{R}_{lin}$ plays a role in providing linear combinations that the evaluator can actually do with the available wire label. For instance, if the evaluator has wire label $A^1 = A^0 + \Delta$, she cannot include only one of A^0 and Δ in the linear combination. From this observation, $\mathbf{R}_{lin}$ in RR21 [14, Equation (6)] can be written as the form of

$$\mathbf{R}_{lin} = \left[\begin{array}{cccc|cc} \lambda_{00A,L} & \lambda_{00A,R} & \lambda_{00B,L} & \lambda_{00B,R} & 0 & 0 \\ \lambda_{01A,L} & \lambda_{01A,R} & \lambda_{01B,L} & \lambda_{01B,R} & \lambda_{01B,L} & \lambda_{01B,R} \\ \lambda_{10A,L} & \lambda_{10A,R} & \lambda_{10B,L} & \lambda_{10B,R} & \lambda_{10A,L} & \lambda_{10A,R} \\ \lambda_{11A,L} & \lambda_{11A,R} & \lambda_{11B,L} & \lambda_{11B,R} & \lambda_{11A,L}+\lambda_{11B,L} & \lambda_{11A,R}+\lambda_{11B,R} \\ \hline \rho_{00A,L} & \rho_{00A,R} & \rho_{00B,L} & \rho_{00B,R} & 0 & 0 \\ \rho_{01A,L} & \rho_{01A,R} & \rho_{01B,L} & \rho_{01B,R} & \rho_{01B,L} & \rho_{01B,R} \\ \rho_{10A,L} & \rho_{10A,R} & \rho_{10B,L} & \rho_{10B,R} & \rho_{10A,L} & \rho_{10A,R} \\ \rho_{11A,L} & \rho_{11A,R} & \rho_{11B,L} & \rho_{11B,R} & \rho_{11A,L}+\rho_{11B,L} & \rho_{11A,R}+\rho_{11B,R} \end{array}\right],$$

where all entries are binary.[5] They obtained candidates for $\mathbf{R}_{lin}$ that satisfies (6) through computer search among these matrices.

A cumbersome point here is that $\mathbf{R}_{lin}$ generally depends on the permute bits α and β. On the contrary, recall that in the previous constructions the matrices are independent of the choices of the permute bits.

[4] We remark that we rearranged Eq.(3) in [14] so that the top/bottom-half rows represent the equations for the left/right-part, respectively. Compare this with the original paper where they arranged the rows in the order of left and right alternately for each (i,j), i.e., $(0,0)$-left, $(0,0)$-right, $\ldots$, $(1,1)$-left, $(1,1)$-right.

[5] $\lambda_{ij,X,Z}$ (resp. $\rho_{ij,X,Z}$) contributes as a coefficient of X^k_Z when computing the left-half (resp. right-half) of output labels, where $X \in \{A,B\}$, $Z \in \{L,R\}$. k is set to i if $X = A$ and j otherwise.

Knowing an information on $\mathbf{R}_{lin}$ is crucial for the evaluator to compute the output label, while knowing the permute bits directly reveals the secret real values to the evaluator. Thus the authors in [14] discuss how to overcome this issue using the technique called *dicing*. In the next paragraph, we shall briefly explain the dicing technique from our algebraic perspective. We believe that our algebraic representation is simpler than the original description to understand the technique.

Our Algebraic Perspective. Similarly as before, we apply our technique using the sum-of-product expression (Definition 2). However, observe that we have the column of length 8 unlike the previous cases. In this case, we remark that it is natural to consider the top/bottom 4-dimensional vector separately instead of applying $\mathcal{F}$ directly to the 8-dimensional vector. This is because each of the 4-dimensional vectors contributes to each of the left/right part depending on 4 possible choices of color bits combination.

For instance, let us consider the vector $(1,1,0,0 \mid 0,0,0,0)^\top$. Applying $\mathcal{F}$ to each halves, we have $(x+1 \mid 0)^\top$, where 0 is considered as a zero function. We also observe that, conversely, $(\mathcal{T}(x+1) \mid \mathcal{T}(0))^\top = (1,1,0,0 \mid 0,0,0,0)^\top$.

Based on this idea, we again rewrite Equation (6) into our algebraic form. To begin with, we consider the matrix $\mathbf{R}_{lin}$. Then, applying $\mathcal{F}$ to each upper/lower column of $\mathbf{R}_{lin}$, we can express this as the form of

$$\mathbf{R} = \big[\mathbf{R}_A \mid \mathbf{R}_B \mid x\mathbf{R}_A + y\mathbf{R}_B\big],$$

where $\mathbf{R}_A$ and $\mathbf{R}_B$ are 2×2 matrices of polynomials in $\mathbb{F}_2[x,y]$. This is derived from the property that the Hadamard product is preserved as the polynomial product via $\mathcal{F}$ (see Section 3.1). For instance, among the upper part of odd columns, we have a relation $(0, \lambda_{01B,L}, \lambda_{10A,L}, \lambda_{11A,L}+\lambda_{11B,L}) = (\lambda_{00A,L}, \lambda_{01A,L}, \lambda_{10A,L}, \lambda_{11A,L}) \circ (0,0,1,1) + (\lambda_{00B,L}, \lambda_{01B,L}, \lambda_{10B,L}, \lambda_{11B,L}) \circ (0,1,0,1)$. Therefore, we obtain $\mathcal{F}_{(0,\lambda_{01B,L},\lambda_{10A,L},\lambda_{11A,L}+\lambda_{11B,L})^\top} = x \cdot [\mathbf{R}_A]_{11} + y \cdot [\mathbf{R}_B]_{11}$, where $[\mathbf{A}]_{ij}$ denotes the (i,j)-th element of a matrix $\mathbf{A}$.

Now, it is interesting to see that

$$\begin{aligned}\mathbf{R} \cdot \left[A_L^0, A_R^0, B_L^0, B_R^0, \Delta_L, \Delta_R\right]^\top &= \mathbf{R}_A \begin{bmatrix} A_L^0 \\ A_R^0 \end{bmatrix} + \mathbf{R}_B \begin{bmatrix} B_L^0 \\ B_R^0 \end{bmatrix} + (x\mathbf{R}_A + y\mathbf{R}_B) \begin{bmatrix} \Delta_L \\ \Delta_R \end{bmatrix} \\ &= \mathbf{R}_A \begin{bmatrix} A_L^x \\ A_R^x \end{bmatrix} + \mathbf{R}_B \begin{bmatrix} B_L^y \\ B_R^y \end{bmatrix}.\end{aligned}$$

Thus, it actually reflects the behaviour of the evaluator so that she can exploit only the values of A^x and B^y.

We are now ready to provide our algebraic form of RR21's construction. We applied $\mathcal{F}$ to $\mathbf{V}_{lin}$ and $\mathbf{M}_{lin}$, and obtained, respectively,

$$\mathbf{V} = \left[\begin{array}{cc|ccc} 1 & 0 & x & 0 & x+y \\ 0 & 1 & 0 & y & x+y \end{array}\right] \text{ and } \mathbf{M} = \left[\begin{array}{cc|cc|cc} x+1 & x & 0 & 0 & x+y+1 & x+y \\ 0 & 0 & y+1 & y & x+y+1 & x+y \end{array}\right].$$

Let us write $\overrightarrow{H} = (H(A^0), H(A^1), H(B^0), H(B^1), H(A^0+B^0), H(A^0+B^1))^\top$, $\overrightarrow{C} = (C_L, C_R)^\top$, and $\overrightarrow{G} = (G_0, G_1, G_2)^\top$. Then, we can rewrite Equation (6) as

follows:

$$\mathbf{V}\begin{bmatrix}\overrightarrow{C}\\ \overrightarrow{G}\end{bmatrix} = \mathbf{M}\overrightarrow{H} + \mathbf{R}_A\left(\begin{bmatrix}A_L^0\\ A_R^0\end{bmatrix} + x\begin{bmatrix}\Delta_L\\ \Delta_R\end{bmatrix}\right) + \mathbf{R}_B\left(\begin{bmatrix}B_L^0\\ B_R^0\end{bmatrix} + y\begin{bmatrix}\Delta_L\\ \Delta_R\end{bmatrix}\right) + (x+\alpha)(y+\beta)\begin{bmatrix}\Delta_L\\ \Delta_R\end{bmatrix}. \tag{7}$$

Note that the column space of $\mathbf{M}$ and $\mathbf{V}$, consisting of vectors over $\mathbb{F}_2[x,y]$, are the same.

In order to construct a garbling scheme from this equation, one has to fill out $\mathbf{R}_A$ and $\mathbf{R}_B$ so that the equality holds for arbitrary choice of (x,y). Although the original paper found the matrix $\mathbf{R}_{lin}$ exhaustively, we can provide a simple algebraic way to find out $\mathbf{R}_A$ and $\mathbf{R}_B$ based on our representation. At the beginning, we observe that, on the left-hand side,

$$\mathbf{V}\begin{bmatrix}\overrightarrow{C}\\ \overrightarrow{G}\end{bmatrix} = \begin{bmatrix}C_L + (G_0+G_2)x + G_2 y\\ C_R + G_2 x + (G_1+G_2)y\end{bmatrix}.$$

Check that the y-coefficient of the upper polynomial and the x-coefficient of the lower polynomial should be the same. Moreover, in order to hold the equality in (7), the right-hand side should satisfy the same condition. Since $\mathbf{M}$ generates the same space as $\mathbf{V}$, it is readily checked that $\mathbf{M}\overrightarrow{H}$ satisfies the desired condition. So, it remains to choose $\mathbf{R}_A$ and $\mathbf{R}_B$ in a way that the remaining part satisfies the same condition. Also, it should be noted that $\mathbf{R}_A$ and $\mathbf{R}_B$ are chosen so that they cancel out the quadratic terms $xy\Delta_L$ and $xy\Delta_R$, since the left-hand side of (7) only consists of linear polynomials.

In the subsequent sections, we shall describe a general way to find such matrices satisfying the above mentioned conditions. For instance, we shall provide $\mathbf{R}_A$ and $\mathbf{R}_B$ in Example 1, Section 5.3. Here, we observe that $\mathbf{R}_A$ and $\mathbf{R}_B$ are dependent of the permute bits α and β.

Dicing Technique. As mentioned above, the garbler should send an information on $\mathbf{R}_A$ and $\mathbf{R}_B$ obliviously to the evaluator in order not to reveal α and β. They dealt with this issue using the dicing technique, which we shall explain in terms of our algebraic perspective.

In order for the evaluator to compute the output label correctly, it is sufficient to have only the evaluated value of $\mathbf{R}$ at the corresponding color bits instead of the entire $\mathbf{R}$. Here, recall that $\mathbf{R}_A$ (similarly for $\mathbf{R}_B$) is a matrix with entries in $\mathbb{F}_2[x,y]$. So, we denote $\mathbf{R}_A(i,j)$ (resp. $\mathbf{R}_B(i,j)$) by the matrix with all of its entries are evaluated at $(x,y)=(i,j)$. We call the value $(\mathbf{R}_A(i,j), \mathbf{R}_B(i,j))$ the *marginal view* at (i,j).

Because the garbler does not know which (i,j) will be held by the evaluator, the marginal views are sent in a garbled form in a way that the evaluator can only obtain the marginal view at her input (i,j). For instance, the garbler generates auxiliary ciphertexts, say $(z_1,\ldots,z_t)$. Using z_k's, the evaluator should be able to recover only $\mathbf{R}_A(i,j)$ and $\mathbf{R}_B(i,j)$ based on her input A^i and B^j. Then, she

uses this information to recover the output label. Refer to Appendix A for more details on the dicing technique.

Where The Improvement Comes From. In this paragraph, we provide an intuition on how RR21 obtained its improvement over the half-gate scheme in terms of communication costs. We emphasize that our algebraic approach is more intuitive than its original description to understand what changes have led RR21 to achieve such improvement. It would be helpful to know whether the idea of RR21 can be further generalized.

Recall that $\overrightarrow{C}$ and $\overrightarrow{G}$ are determined by Equation (7), where the equation is a system of two linear equations. Observe that, as described above, these two equations are dependent on each other. Precisely, looking at the left-hand side, there are two linear polynomials $C_L+(G_0+G_2)x+G_2y$ and $C_R+G_2x+(G_1+G_2)y$, where G_2 is shared in common as y-coefficient and x-coefficient, respectively. In other words, these two linear polynomials can be determined by five free variables, C_L, C_R, G_0, G_1 and G_2.

Indeed, the number of free variables in the left-hand side is closely related to the number of ciphertexts. If there were two independent linear equations, we would have six free variables in total. This will not help us to improve against the half-gate scheme. On the other hand, it can be seen that RR21 essentially reduces the number of ciphertexts by forcing a system of two linear equations to have a certain relation.

Note that the number of coefficients to be determined is actually the same as the rank of the matrix $\mathbf{V}_{lin}$ in terms of RR21's representation. We believe that our algebraic representation is more intuitive to understand what is going on than considering the rank, especially when we try to generalize the idea of RR21.

When it comes to generalization of RR21's idea, it might be natural to consider a system of multiple linear equations with certain number of relations. If there were s linear equations with ℓ relations, then we would have $3s-\ell$ free variables. Among them, s variables are used to determine s-chunks of the output label $C=(C_1,\ldots,C_s)$. So, there will be $2s-\ell$ ciphertexts in total, each of which is of κ/s bits, and so on and so forth. However, as we shall see in Section 5, this approach will not provide us an improved construction.

4 Our Garbling Model

In this section, we provide a comprehensive model that extends the linear garbling model by [16]. We argue that this is a natural generalization of the previous modeling, and our model captures all existing practical garbling schemes. Based on this, we will provide a lower bound of garbling schemes in the next section.

4.1 Motivations

Our main objective of this paper is to provide a lower bound for garbling schemes. In order to have a meaningful bound, we should specify a methodology of garbling

constructions. Above all, we would like to focus on constructions that exploit only symmetric-key primitives and linear operations. Our model assumes parties can make polynomially many queries to a random oracle. In other words, our lower bound is obtained in the world of Minicrypt [7] in which only symmetric-key primitives exist and no public-key primitives exist.

Observe that the original linear garbling model also assumes the same as above. So, one might not expect to have a new model that differs from the original one. However, we shall explain below that there is a very natural way to extend the linear garbling model without leaving the world of Minicrypt.

Let us consider how each party in a garbling scheme plays their role by using only symmetric-key primitives and linear operations. We assume that the linear operations are carried out over fields of characteristic 2. And we do not allow the operations to be performed outside of that given field.

- *Garbler's Behaviour (Basic).* We first describe a basic approach that one can imagine for the garbler's behaviour. On the side of the garbler, he takes the input wire labels as inputs and makes queries to a random oracle using them. Based on these values, he returns the output wire labels along with the ciphertexts by applying suitable linear operations. The output labels and the ciphertexts may vary depending on the choice of the permute bits. Hence, which linear operations to apply would be dependent on the permute bits.
- *Evaluator's Behaviour (Basic).* The evaluator uses her input wire labels to make queries to a random oracle. Using those values together with the ciphertexts received from the garbler, she applies suitable linear combinations to get the output wire label that corresponds to the evaluator's input wire labels. Since the permute bits are to be hidden to the evaluator, her linear combinations should be only dependent on the color bits.

Let A^x and B^y be the evaluator's input wire labels for some $x, y \in \mathbb{F}_2$. The evaluator should obtain $C^{(x+\alpha)(y+\beta)}$ for the permute bits (α, β). Here, C^0 represents **false**. The prescribed behaviour can be formally written as

$$C^{(x+\alpha)(y+\beta)} = \mathbf{V}_{x,y}(G_1, \ldots, G_r, \widetilde{H}_1, \ldots, \widetilde{H}_{\widetilde{n}}, A^x, B^y)^\top,$$

where $G_1, \ldots, G_r$ are the ciphertexts and $\widetilde{H}_1, \ldots, \widetilde{H}_{\widetilde{n}}$ are random oracle responses using A^x and B^y. Moreover, $\mathbf{V}_{x,y}$ is a $1 \times (\widetilde{n} + r + 2)$ matrix and its entries have their values in $\mathbb{F}_{2^\kappa}$ which are decided by the color bits x and y, i.e. $\mathbf{V}_{x,y}$ is defined over $\mathbb{F}_{2^\kappa}[x, y]$. Parsing $\mathbf{V}_{x,y}$ suitably, one might notice that Equation (2) and (5) are the representation equivalent to the above.

Indeed, the above discussion has no difference with the consideration in the linear garbling model. To extend the linear garbling model in a non-trivial way, we consider the following.

In the above, observe that the output label is determined by only *one* equation, i.e. $\mathbf{V}_{x,y}$ consists of one row. Then it is natural to consider the following case as a next step: What if the output label is determined by a system of *multiple* equations?

- *Extension 1 (Slicing).* We assume that the wire labels are split into s parts for an integer s. The garbler makes random oracle queries on *unsliced* wire labels to get random κ/s-bit strings. He uses these random oracle queried values and sliced input wire labels (which is also κ/s-bit) to compute the sliced output labels and ciphertexts by applying s linear combinations. On the other hand, the evaluator applies s linear combinations using her input labels and ciphertexts to have the output label.

In this case, the evaluator's behaviour can be written as follows.

$$\overrightarrow{C}^{(x+\alpha)(y+\beta)} = \mathbf{V}_{x,y}(G_1, \ldots, G_r, \widetilde{H}_1, \ldots, \widetilde{H}_{\tilde{n}}, \overrightarrow{A^x}, \overrightarrow{B^y})^\top,$$

where each wire label is sliced into s-chunks of the same bit-length, $(A_1, \ldots, A_s)$, and written using vector notation as $\overrightarrow{A}$. In addition, $\mathbf{V}_{x,y}$ now has s rows.

At this point, a somewhat weird but plausible question can be raised: Should $\mathbf{V}_{x,y}$ be dependent only on the color bits? What if it also depends on the permute bits?[6] Of course, in such case, the evaluator cannot apply linear operations using $\mathbf{V}_{x,y}$ since she does not know the permute bits. However, what if we assume that the garbler can send auxiliary information so that the evaluator recovers $\mathbf{V}_{x,y}$ without knowing the permute bits?

- *Extension 2 (Dicing).* The garbler makes auxiliary random oracle queries using his wire labels. He also takes a part of $\mathbf{V}_{x,y}$ that depends on α, β as inputs. Apply linear combinations on them to obtain auxiliary ciphertexts, say $\overrightarrow{z} := (z_1, \ldots, z_t)$. On the other hand, the evaluator makes auxiliary random oracle queries with her input labels. Together with the auxiliary ciphertexts $\overrightarrow{z}$, she applies linear operations that only depend on her color bits (x, y). This will reveal the part of $\mathbf{V}_{x,y}$ that depends on α, β. Hence, she can exploit it to obtain the desired output label.

In the above description, observe that we only exploit random oracle queries and linear operations. Thus we are still in the world of Minicrypt.

One might observe that the RR21 construction is an example that both Extension 1 and 2 are applied. In particular, the evaluator applies Equation (7) after proper rearranging. The value $(\mathbf{R}_A, \mathbf{R}_B)$, the part that depends on α, β, is sent to the evaluator by using the dicing technique requiring a small number of ciphertexts.

We argue that the above extensions possibly capture a broader class of garbling constructions than the linear garbling model that one can imagine from the world of Minicrypt.

[6] For notational convention, we still think that $\mathbf{V}_{x,y}$ is defined over $\mathbb{F}_{2^\kappa}[x, y]$ instead of $\mathbb{F}_{2^\kappa}[x, y, \alpha, \beta]$. In such case, we think that their coefficients are parameterized by α and β.

4.2 Formalizing a Garbling Model

Based on the observation from the previous section, we formalize our new model for garbling schemes. We argue that our model encompasses all existing practical garbling schemes.

Let $\mathbb{K}$ be a subfield of $\mathbb{F}_{2^{\kappa/s}}$. We say that a garbling scheme is *s-linear over* $\mathbb{K}$ if it satisfies the following form:

- Gb: Parameterized by integers m, n, r, s, t, u, $s \times (sm+n)$ matrices $\mathbf{A}^0$, $\mathbf{A}^1$, $\mathbf{B}^0$, $\mathbf{B}^1$, $\{\mathbf{C}^0_{\alpha,\beta} : \alpha, \beta \in \{0,1\}\}$, $\{\mathbf{C}^1_{\alpha,\beta} : \alpha, \beta \in \{0,1\}\}$, $r \times (sm+n)$ matrix $\{\mathbf{G}_{\alpha,\beta} : \alpha, \beta \in \{0,1\}\}$, where the entries of all vectors/matrices are in $\mathbb{K}$. And also parameterized by an integer c, $t \times (\ell + u)$ matrix $\mathbf{z}$, and u-dimensional vector $\{\overrightarrow{r_{\alpha,\beta}} : \alpha, \beta \in \{0,1\}\}$, where all the entries are in $\mathbb{F}_{2^c}$.
  Set a parameter $\mathtt{dicing} \in \{0,1\}$.
  1. (Choose random values) Choose $R_i \leftarrow \mathbb{F}_{2^\kappa}$ for $i = 1, \ldots, m$. Parse R_i into $(R_{i,1} \| \cdots \| R_{i,s})$ where $R_{i,j} \in \mathbb{F}_{2^{\kappa/s}}$.
  2. (Random oracle queries, κ/s-bit part) Make n distinct queries to a random oracle with κ/s-bit output (which can be chosen as a deterministic function of the R_i values). Let $H^{\mathrm{kap}}_1, \ldots, H^{\mathrm{kap}}_n$ denote the responses to these queries. Define $\overrightarrow{S}^{\mathrm{kap}} := (R_{1,1}, \ldots, R_{m,s}, H^{\mathrm{kap}}_1, \ldots, H^{\mathrm{kap}}_n)^\top$.[7]
  3. (Random oracle queries, c-bit part) If $\mathtt{dicing} = 0$, then $\overrightarrow{S}^{\mathrm{con}} = \perp$. Otherwise, make ℓ distinct queries to a random oracle with c-bit outputs (which can be chosen as a deterministic function of the R_i values). Let $H^{\mathrm{con}}_1, \ldots, H^{\mathrm{con}}_\ell$ denote the responses to these queries. Define $\overrightarrow{S}^{\mathrm{con}} := (H^{\mathrm{con}}_1, \ldots, H^{\mathrm{con}}_\ell)^\top$.[7]
  4. (Choose masking bits) Choose random permute bits $\alpha, \beta \leftarrow \{0,1\}$.
  5. (Input wire labels) For $i = 0, 1$, compute s-dimensional vectors $\overrightarrow{A}^i = \mathbf{A}^i \overrightarrow{S}^{\mathrm{kap}}$, $\overrightarrow{B}^i = \mathbf{B}^i \overrightarrow{S}^{\mathrm{kap}}$ and define $A^i = (A^i_1 \| \ldots \| A^i_s)$, $B^i = (B^i_1 \| \ldots \| B^i_s)$. Then $(A^0 \parallel 0, A^1 \parallel 1)$ and $(B^0 \parallel 0, B^1 \parallel 1)$ are taken as the input wire labels with A^α and B^β corresponding to **false**. Here, the superscripts denote the public color bits.
  6. (Output wire labels) For $i = 0, 1$, compute s-dimensional vectors $\overrightarrow{C}^i = \mathbf{C}^i_{\alpha,\beta} \overrightarrow{S}^{\mathrm{kap}}$ and define $C^i = (C^i_1 \| \ldots \| C^i_s)$. Here, C^0 corresponds to **false**.
  7. (Ciphertexts) Compute $\overrightarrow{G} = \mathbf{G}_{\alpha,\beta} \overrightarrow{S}^{\mathrm{kap}}$. If $\mathtt{dicing} = 0$, then set $\overrightarrow{z} = \perp$. Otherwise, compute $\overrightarrow{z} = \mathbf{z}(\overrightarrow{r}_{\alpha,\beta} \| \overrightarrow{S}^{\mathrm{con}})$. Then, $(\overrightarrow{G}, \overrightarrow{z})$ comprise the garbled circuit. The former are called *main* ciphertexts and the latter are *auxiliary* ciphertexts.
- En: On input $a, b \in \{0,1\}$, set $x = a + \alpha$ and $y = b + \beta$, where α and β are the permute bits chosen above. Output $A^x \parallel x$ and $B^y \parallel y$.
- Ev: Parameterized by integer n, $s \times (n+r+2s)$ matrices $\{\mathbf{V}^{\mathrm{kap}}_{x,y} : x, y \in \{0,1\}\}$ and $s(n+r+2s) \times (n+t)$ matrices $\{\mathbf{W}^{\mathrm{con}}_{x,y} : x, y \in \{0,1\}\}$, where each of the entries is in $\mathbb{K}$.

[7] "kap" is the abbreviation for *kappa* for the sake of meaning that H^{kap}_i has "κ"/s-bit output. Similarly, "con" stands for *constant* where H^{con}_i has (a constant) c-bit output.

1. (Inputs) The input are wire labels $A^x \| x$ and $B^y \| y$, tagged with their corresponding color bits, and the garbled circuit $(\overrightarrow{G}, \overrightarrow{z}) = (G_1, \ldots, G_r, z_1, \ldots, z_t)$.
2. (Random oracle queries, κ/s-bit part) Make n distinct oracle queries to the random oracle (chosen as a deterministic function of the input wire labels). Denote the responses to the queries by $\widetilde{H}_1^{\mathrm{kap}}, \ldots, \widetilde{H}_n^{\mathrm{kap}}$. Define $\overrightarrow{T}^{\mathrm{kap}} := (A_1^x, \ldots, A_s^x, B_1^y, \ldots, B_s^y, \widetilde{H}_1^{\mathrm{kap}}, \ldots, \widetilde{H}_n^{\mathrm{kap}}, G_1, \ldots, G_r)^\top$.
3. (Random oracle queries, constant bit part) If $\overrightarrow{z} = \perp$, then skip this step. Otherwise, make ℓ distinct oracle queries to the random oracle (chosen as a deterministic function of the input wire labels). Denote the responses to the queries by $\widetilde{H}_1^{\mathrm{con}}, \ldots, \widetilde{H}_\ell^{\mathrm{con}}$. Define $\overrightarrow{T}^{\mathrm{con}} := (\widetilde{H}_1^{\mathrm{con}}, \ldots, \widetilde{H}_\ell^{\mathrm{con}}, z_1, \ldots, z_t)^\top$.
4. (Control bit recovery) If $\overrightarrow{z} = \perp$, then set $\mathbf{V}_{x,y}^{\mathrm{con}} = \mathbf{0}$. Otherwise, compute $\mathbf{W}_{x,y}^{\mathrm{con}} \overrightarrow{T}^{\mathrm{con}}$ and parse it into a $s \times (n + r + 2s)$ matrix $\mathbf{V}_{x,y}^{\mathrm{con}}$.
5. Output the value $(\mathbf{V}_{x,y}^{\mathrm{kap}} + \mathbf{V}_{x,y}^{\mathrm{con}}) \overrightarrow{T}^{\mathrm{kap}}$.

We also say that the garbling scheme in the above sense is *free-XOR compatible* if it satisfies the following:

- The parameters in Gb satisfy

$$\mathbf{A}^1 + \mathbf{A}^0 = \mathbf{B}^1 + \mathbf{B}^0 = \mathbf{C}_{\alpha,\beta}^1 + \mathbf{C}_{\alpha,\beta}^0 = [\mathbf{I}_s \parallel \mathbf{0}],$$

  where $\mathbf{I}_s$ is the s-dimensional identity matrix and $\mathbf{0}$ is the zero matrix of dimension $s \times (sm + n - s)$.
- In Step 1, Gb sets $\Delta := R_1$ and parses Δ into $(\Delta_1 \| \ldots \| \Delta_s)$.
- In Step 2, Gb defines $\overrightarrow{S}^{kap} := (\Delta_1, \ldots, \Delta_s, \ldots)^\top$.

From the above, the free-XOR compatibility assures that $\overrightarrow{A}^0 + \overrightarrow{A}^1 = \overrightarrow{B}^0 + \overrightarrow{B}^1 = \overrightarrow{C}^0 + \overrightarrow{C}^1 = \overrightarrow{\Delta}$, where $\overrightarrow{\Delta} = (\Delta_1, \ldots, \Delta_s)^\top$.[8]

In the definition of the s-linear garbling model, the parameter `dicing` indicates whether the garbler randomizes the control bits (if $\texttt{dicing} = 1$) or not. All schemes that belong to the linear garbling model have $(s, \texttt{dicing}) = (1, 0)$, i.e. the wire labels are not sliced and there is no control bit randomization applied. On the other hand, the RR21 construction has $(s, \texttt{dicing}) = (2, 1)$, i.e. the wire labels sliced into two parts and the control bit randomization technique is applied. In Appendix B, we show that RR21 is 2-linear in the above sense.

As a side note, we argue that the schemes by [8] and [1] are not s-linear in the above sense. Although, their constructions garble an AND gate with κ-bit ciphertexts, it is less useful since they can garble only a single gate in isolation. We think it is reasonable to exclude such schemes in our modeling in terms of practicality. In Appendix C, we explain why their constructions are not linear.

[8] We emphasize that our free-XOR compatibility requires that both input labels and output labels have the same correlation, i.e. the labels for **false** and **true** on each wire have the same offset Δ. It assures the scheme is composable with itself, i.e. output labels by previous garbled gates can be used as input labels to garble next gates.

Remark 1. We remark that the slicing and the dicing technique can be independently applicable. We argue that applying the dicing technique is only decided by whether the evaluator's linear combinations include parts that depend on the permute bits. For instance, RR21's construction requires the dicing since $\mathbf{R}_A$ and $\mathbf{R}_B$ are dependent on α and β. An interesting aspect is that it seems very unlikely to find $\mathbf{R}_A$ and $\mathbf{R}_B$ that do not depend on the permute bits when $s \geq 2$ in general. Thus it seems impossible to avoid the dicing technique when the slicing technique is applied. However, it may be possible to obtain a scheme with slicing but no dicing, although we are unaware of such constructions at this point of writing.

Remark 2. We observe that most garbling schemes that are free-XOR compatible actually employ $\mathbb{K} = \mathbb{F}_2$ instead of the entire field. Up to our knowledge, the only scheme in the linear garbling model that uses $\mathbb{K} = \mathbb{F}_{2^\kappa}$ is [13] which is based on the polynomial interpolation method. Note that this scheme is not free-XOR compatible.

5 Lower Bound on Garbled Circuit

In this section, we present a lower bound for garbled AND gates in our model of s-linear garbling scheme under several reasonable assumptions. From now on, we assume that garbling schemes are free-XOR compatible.

5.1 Garbling Equations

In this section, we provide a notion of garbling equations. Once it is defined, we realize that constructing a garbling scheme reduces to find a suitable form of garbling equations. Based on this observation, we shall prove our lower bound in the model of s-linear garbling.

In the s-linear garbling model, the algorithm Ev returns an output by computing $(\mathbf{V}^{\mathrm{kap}}_{x,y} + \mathbf{V}^{\mathrm{con}}_{x,y})\overrightarrow{T}^{\mathrm{kap}}$. Let us consider linear operations on the part of random oracle responses in $\overrightarrow{T}^{kap}$. For instance, assume that Ev applies the linear combination like $H(A^x) + \cdots$ on the input A^x. Then it can be rephrased as a polynomial like $(x+1)H(A^0) + xH(A^1) + \cdots$. Hence, we can rewrite this part of operations in the form of $\mathbf{M}\overrightarrow{H}$, where $\mathbf{M}$ is the part that only depends on x and y and $\overrightarrow{H}$ is a vector consisting of all possible random oracle responses that can be made by the parties. In other words, $\mathbf{M}$ decides which oracle responses to apply among all possibilities depending on x, y.

Suitably parsing the matrix $\mathbf{V}^{\mathrm{kap}}_{x,y} + \mathbf{V}^{\mathrm{con}}_{x,y}$, one can rewrite the algorithm Ev as follows

$$\overrightarrow{C} + (x+\alpha)(y+\beta)\overrightarrow{\Delta} = \mathbf{W}\overrightarrow{G} + \mathbf{M}\overrightarrow{H} + \mathbf{R}_A\overrightarrow{A^x} + \mathbf{R}_B\overrightarrow{B^y}, \tag{8}$$

where all the matrices $\mathbf{M}$, $\mathbf{W}$, $\mathbf{R}_A$ and $\mathbf{R}_B$ have s rows. If the garbling scheme is over $\mathbb{K}$, then all of these matrices are defined over $\mathbb{K}[x, y]$. We assume that polynomials in $\mathbb{K}[x, y]$ have coefficients parameterized by α, β.[6]

Finally, we observe that in order to construct a garbling scheme one should decide what matrices to apply in the above equation. After the matrices are determined, the algorithm Gb solves $\overrightarrow{C}$ and $\overrightarrow{G}$ in the following equation,

$$\mathbf{V}\begin{bmatrix}\overrightarrow{C}\\ \overrightarrow{G}\end{bmatrix} = \mathbf{M}\overrightarrow{H} + \mathbf{R}_A(\overrightarrow{A} + x\overrightarrow{\Delta}) + \mathbf{R}_B\ (\overrightarrow{B} + y\overrightarrow{\Delta}) + (x+\alpha)(y+\beta)\overrightarrow{\Delta}, \quad (9)$$

so that the equality holds for all choices of (x, y). Here, $\mathbf{V} = [\mathbf{I}_s|\mathbf{W}]$ and $\mathbf{I}_s$ is the identity matrix of the dimension s. We call the equation of the form (9) (or equivalently, (8) at evaluator's view) the *garbling equations*.

5.2 Conditions on the Matrices M and V

In the construction of the garbling equation, setting $\mathbf{M}$ and $\mathbf{V}$ is of prime importance to be considered first. This is because the number of the columns of $\mathbf{V}$, which is obtained as a column-reduced matrix of $\mathbf{M}$, is closely related to the number of ciphertexts.

Although our argument also holds for arbitrary $\mathbb{K}$, for the sake of simplicity, we focus on the case of s-linear garbling scheme over $\mathbb{F}_2$. In other words, the matrices $\mathbf{M}$, $\mathbf{V}$ and $\mathbf{R}$ are considered to be defined over $\mathbb{F}_2[x, y]$. For the general case, refer to Appendix D.

5.2.1 Observation on M

In this section, we discuss on how the matrix $\mathbf{M}$ looks like. Recall that each entries of the s-dimensional vector $\mathbf{M}\overrightarrow{H}$ is determined by (linear combinations of) responses to random oracle queries using the input labels A^x and B^y. Since the output by random oracles is solely determined by the color bits (x, y), it can be written as a function $\eta : \mathbb{F}_2 \times \mathbb{F}_2 \mapsto \mathbb{F}_{2^L}$, where L is the bit-length of the random oracle outputs (typically, $L = \kappa/s$ in our case). In other words, $\mathbf{M}\overrightarrow{H}$ can be represented like $(\dots, \sum_k \eta_k(x, y), \dots)^\top$.

Note that *any* function is completely decided by the values at all possible inputs, i.e. given any $h_0, h_1, h_2, h_3 \in \mathbb{F}_{2^L}$, a function η that has values h_{i+2j} at (i, j) can be written as a polynomial in $\mathbb{F}_{2^L}[x, y]/\langle x^2 + x, y^2 + y\rangle$ using the Lagrange interpolation:

$$\eta(x, y) = h_0 \cdot (x+1)(y+1) + h_1 \cdot (x+1)y + h_2 \cdot x(y+1) + h_3 \cdot xy, \quad (10)$$

where $h_{2i+j} = \eta(i, j)$.

Now we describe how the matrix $\mathbf{M}$ and the vector $\overrightarrow{H}$ can be represented. To ease the description, without loss of generality, we only consider the case when $\mathbf{M}\overrightarrow{H}$ is 1-dimensional, i.e. $s = 1$. For now, we assume that $\mathbf{M}\overrightarrow{H} = \eta(x, y)$ for some η. Given the polynomial $\eta \in \mathbb{F}_{2^L}[x, y]/\langle x^2 + x, y^2 + y\rangle$, we write $\eta(x, y) = \sum_{i,j\in\{0,1\}} h_{2i+j} \cdot (x + 1 - i)(y + 1 - j)$ as above. Then we set $\mathbf{M} = [(x+1)(y+1), \cdots, xy]$ and $\overrightarrow{H} = (h_0, \dots, h_3)^\top$.

However, depending on the values $h_0, \ldots, h_3$, one can further reduce the number of columns of $\mathbf{M}$. For instance, consider the case when $h_0 = h_1$ and $h_2 = h_3$. In this case, we have $\eta = h_0 \cdot (x+1) + h_2 \cdot x$ and set $\mathbf{M}\overrightarrow{H} = [x+1 \; x](h_0, h_2)^\top$. We observe that this case occurs when we use the random oracle queries of the form $H(A^x)$ where $h_0 = H(A^0)$ and $h_2 = H(A^1)$. Similarly, when $h_0 = h_2$ and $h_1 = h_3$, we have $\mathbf{M}\overrightarrow{H} = [y+1 \; y](h_0, h_1)^\top$. This corresponds to the case of using $H(B^y)$. In the case of $h_0 = h_3$ and $h_1 = h_2$, we can write $\mathbf{M}\overrightarrow{H} = [x+y+1 \; x+y](h_0, h_1)^\top$. This case occurs when we use $H(A^x + B^y)$ under the free-xor condition.

From the above observation, it is natural to ask whether one can use any other form of random oracle queries that would yield a better construction for garbling. In the following, we argue that we shall only consider the aforementioned forms of oracle queries. To begin with, we define the notion of *rank* of the bivariate polynomial η.

Definition 3. *Let $\eta \in \mathbb{F}_{2^L}[x, y]/\langle x^2 + x, y^2 + y\rangle$ as described in Eq.* (10)*. Let us consider the elements in $\mathbb{F}_{2^L}$ as L-dimensional vectors over $\mathbb{F}_2$. We shall define the rank of η, denoted by* $\mathrm{rk}(\eta)$*, by the rank of the $(4 \times L)$-dimensional matrix (over $\mathbb{F}_2$) whose rows are $h_0, \ldots, h_3$.*

For instance, in the above example of $\eta = h_0 \cdot (x + 1) + h_2 \cdot x$, we have $\mathrm{rk}(\eta) = 2$. Here, 4 coefficients, $h_0, \ldots, h_3$, are determined by two elements, h_0 and h_2. In general, η can be written as $\eta(x, y) = \sum_{i=0}^{\mathrm{rk}(\eta)-1} t_k \cdot f_k(x, y)$ for some $t_k \in \mathbb{F}_{2^L}$ and $f_k \in \mathbb{F}_2[x, y]$, where $\mathrm{rk}(\eta) \leq 4$ (note that f_k has its coefficients in $\mathbb{F}_2$ not $\mathbb{F}_{2^L}$). We also observe that the column rank of $\mathbf{M}$ is actually the same as the rank of the corresponding η.

In the following, we observe the following property.

Lemma 1. *Let η be given as Eq.* (10) *i.e. $\eta(i, j) = h_{2i+j}$ for $i, j \in \{0, 1\}$. Then, (wlog) the tuple of the coefficients (h_0, h_1, h_2, h_3) should be of one of the following forms (up to a proper permutation):*

1. $\mathrm{rk}(\eta) = 1 : (h_0, *, *, *)$*, where* $* \in \{0, h_0\}$*;*
2. $\mathrm{rk}(\eta) = 2 : (h_0, h_1, *, *)$*, where* $* \in \{0, h_0, h_1, h_0 + h_1\}$*;*
3. $\mathrm{rk}(\eta) = 3 : (h_0, h_1, h_2, *)$*, where* $* \in \{ih_0 + jh_1 + kh_2 : i, j, k \in \{0, 1\}\}$*;*
4. $\mathrm{rk}(\eta) = 4 : (h_0, h_1, h_2, h_3)$.

Proof. The proof is clear from the definition of the rank. □

The above lemma asserts that in some cases of η, corresponding garbling schemes require random oracle queries to be made in a restrictive way which is either useless or unlikely to hold in general. For instance, in the case of $(h_0, h_0, h_0, 0)$, it means that a random oracle H should satisfy $h_0 = H(A^i, B^j)$ for $(i, j) \neq (1, 1)$ and $0 = H(A^1, B^1)$. However, it seems unnatural to force a random oracle H to have the same value at all (A^i, B^j) except at (A^1, B^1).

In the case when a tuple contains $h_0 + h_1$ (where the rank is 2 or 3), it means that H should satisfy a homomorphic property, e.g. $H(A^1, B^0) = H(A^0, B^0) +$

$H(A^0, B^1)$. However, at this stage we are unaware whether such H exists in a way that provides a better garbling construction.

In the case of (h_0, h_1, h_1, h_2) (where the rank is 3), one has $h_1 = H(A^0, B^1) = H(A^1, B^0)$. The construction in [8] might be considered as a particular case of using this type of η. They defined $H(A^i, B^j) := H(A^i +_{\mathbb{Z}} B^j)$, where $+_{\mathbb{Z}}$ denotes the addition of bitstrings considered as integers (see Appendix C). However, this construction is only useful for garbling a single gate only, since it is not guaranteed to hold the same condition for consecutive gates.

Overall, we only consider the following types for the coefficients of η:

Definition 4. *Let η be given as Eq.* (10)*. We call that η* is useful *if it is one of the following forms (up to a proper permutation):*

1. $\mathrm{rk}(\eta) = 2$*:* (h_0, h_1, h_0, h_1)*;*
2. $\mathrm{rk}(\eta) = 3$*:* $(h_0, h_1, h_2, h_0 + h_1 + h_2)$*;*
3. $\mathrm{rk}(\eta) = 4$*:* (h_0, h_1, h_2, h_3)*;*

In the case of rank 2, all possible types of η can be obtained by considering the random oracles queries of the form either $H(A^x)$, $H(B^y)$, and $H(A^x + B^y)$ as we have seen above.

In the case of rank 3, all possible types of η can be obtained by linear combinations of the polynomials of rank 2. For instance, if the coefficients of η is $(h_0, h_1, h_2, h_0+h_1+h_2)$, then we can rewrite η as $s_0(x+y+1)+s_1(x+y)+t_0(x+1)+t_1x$, where $h_0 = s_0 + t_0$, $h_1 = s_1 + t_0$, and $h_2 = s_1 + t_1$. In other words, it corresponds to the case of using random oracles of the form $H(A^x+B^y)+H(A^x)$.

As we are interested in a smaller rank of $\mathbf{M}$ (for smaller ciphertexts), we focus our discussion on the case of $\mathrm{rk}(\eta) \lneq 4$, i.e. we only consider linear polynomials as the entries of $\mathbf{M}$.

In this way, we assume that we only consider the random oracles of the type either $H(A^x)$, $H(B^y)$, and $H(A^x + B^y)$, and their linear combinations. Then the matrix $\mathbf{M}$ and $\overrightarrow{H}$ can be chosen accordingly.

Note that we can use different kinds of random oracles on the same input, so the same type of polynomials can appear multiple times in $\mathbf{M}$. For instance, (although it might be less useful) one might set $\mathbf{M} = \begin{bmatrix} x+1 & x & 0 & 0 & \cdots \\ 0 & 0 & x+1 & x & \cdots \end{bmatrix}$ and $\overrightarrow{H} = (h_0, h_1, h'_0, h'_1, \dots)^\top$, where $h_i := H(A^i)$ and $h'_i := H'(A^i)$. In this case, it means that one might compute the first-half of the wire label by querying random oracle H with inputs A^i and the second-half is computed using another random oracle H'.

We also remark that any nonzero elements in the same column of $\mathbf{M}$ are uniquely determined by the input to the random oracle, which means that any nonzero elements in the same column of $\mathbf{M}$ must be the same. For instance $(x, y+1)^\top$ cannot be a column of $\mathbf{M}$, but $(0, y)^\top$ can.

Summarizing our observations, $\mathbf{M}$ can be written as a matrix of the form:

$$\mathbf{M} = \left[(\ell_1 + 1)\overrightarrow{u_1} \mid \ell_1\overrightarrow{u_1} \mid \cdots \mid (\ell_n + 1)\overrightarrow{u_n} \mid \ell_n\overrightarrow{u_n}\right], \tag{11}$$

where n is a positive integer, ℓ_i is an element in $\{x, y, x+y\}$, and $\overrightarrow{u_i}$ is a s-dimensional binary vector.

5.2.2 Observation on V

Since $\mathbf{V}$ is constructed as a column basis of $\mathbf{M}$, instead of choosing $\mathbf{V}$ directly, we rather impose conditions that $\mathbf{V}$ should satisfy on selecting the matrix $\mathbf{M}$. To begin with, as mentioned above, it is reasonable to assume that $\mathbf{V}$ is of the form $\mathbf{V} = [\mathbf{I}_s \mid \mathbf{W}]$. As $\mathbf{M}$ consists of only linear polynomials, it is obvious that the entries of $\mathbf{W}$ are also linear polynomials. In particular, we can always choose $\mathbf{W}$ so that its entries are all homogeneous (i.e. have a zero constant), since $\mathbf{V}$ contains $\mathbf{I}_s$. To see why, one might observe that any columns in $\mathbf{M}$ of the form $(\ell+1)\overrightarrow{u}$ and $\ell\overrightarrow{u}$ could be generated by $\overrightarrow{u}$ and $\ell\overrightarrow{u}$. As $\overrightarrow{u}$ being a binary vector, it is generated by $\mathbf{I}_s$. Then we might assume $\ell\overrightarrow{u}$ is generated from the columns in $\mathbf{W}$ whose entries have the zero constant terms.

Recalling that it is a column-reduced matrix of $\mathbf{M}$, we might write $\mathbf{V}$ as the form of

$$\mathbf{V} = [\mathbf{I}_s \mid \mathbf{W}] = [\mathbf{I}_s \mid \ell_1\overrightarrow{u_1} \mid \cdots \mid \ell_r\overrightarrow{u_r}]\,. \tag{12}$$

Here, the identity matrix $\mathbf{I}_s$ should be obtained from $\{\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_n}\}$ of $\mathbf{M}$ in Equation (11). Thus the vector $\overrightarrow{u_i}$'s should contain s linearly independent vectors. Denote r by the rank of the matrix $\left[\ell_1\overrightarrow{u_1} \mid \ell_2\overrightarrow{u_2} \mid \cdots \mid \ell_n\overrightarrow{u_n}\right]$. By proper reordering, assume that the set $\{\ell_1\overrightarrow{u_1}, \cdots, \ell_r\overrightarrow{u_r}\}$ is linearly independent.[9] Finally, we can write $\mathbf{V}$ as Equation (12). By the construction, note that we have $r \geq s$.

5.3 Solving the Garbling Equation

Now we are ready to discuss on how to construct the garbling equation for general s. Once we have chosen the matrices $\mathbf{M}$ and $\mathbf{V}$, it remains to find $(\mathbf{R}_A, \mathbf{R}_B)$ so that the equality in the garbling equation (8) holds for all inputs (x, y). To find solutions, we proceed as follows:

(Step 1) Compare the coefficients of the terms on both sides of (8) so that the equality holds;
(Step 2) Find relations between the x-coefficient and the y-coefficient in the left hand side;
(Step 3) Try to choose $(\mathbf{R}_A, \mathbf{R}_B)$ so that the right hand side satisfies the relations of (Step 2);
(Step 4) Check whether information on α and β is not revealed from the value of $(\mathbf{R}_A, \mathbf{R}_B)$ evaluated at some $(x, y) = (i, j)$.

Recall that this evaluated value in (Step 4) is the marginal view.

[9] Remark that it can be linearly independent even if the set $\{\overrightarrow{u_1}, \cdots, \overrightarrow{u_r}\}$ is linearly dependent.

Compare coefficients. We write $\mathbf{R}_X = \mathbf{R}_{X,0}x + \mathbf{R}_{X,1}y + \mathbf{R}_{X,2}$, where $X \in \{A, B\}$ and $\mathbf{R}_{X,i}$ is a $s \times s$ binary matrix. Substituting this to (8) yields

$$\begin{aligned}\overrightarrow{C} + \mathbf{W}\overrightarrow{G} = \mathbf{M}\overrightarrow{H} &+ \Big(\mathbf{R}_{A,0}\overrightarrow{A} + \mathbf{R}_{B,0}\overrightarrow{B} + \big(\mathbf{R}_{A,0} + \mathbf{R}_{A,2} + \beta\mathbf{I}_s\big)\overrightarrow{\Delta}\Big)\, x \\ &+ \Big(\mathbf{R}_{A,1}\overrightarrow{A} + \mathbf{R}_{B,1}\overrightarrow{B} + \big(\mathbf{R}_{B,1} + \mathbf{R}_{B,2} + \alpha\mathbf{I}_s\big)\overrightarrow{\Delta}\Big)\, y \\ &+ \big(\mathbf{R}_{A,1} + \mathbf{R}_{B,0} + \mathbf{I}_s\big)\, xy + (\mathit{constant}).\end{aligned} \tag{13}$$

Recall that the left-hand side is linear since $\mathbf{W}$ only contains linear terms. So, in order that the equality holds, the right-hand side should also be linear. This forces to choose $(\mathbf{R}_A, \mathbf{R}_B)$ satisfying

$$\mathbf{R}_{A,1} + \mathbf{R}_{B,0} + \mathbf{I}_s = 0. \tag{14}$$

Find relations. Prior to discussion on constructing the garbling equations, we take a more look into the relations between the coefficients. From now on, we assume that $r \leq 2s$, since the communication cost would be $(r/s)\kappa$ bits and we are interested in the case of $r/s \leq 2$.

Consider the term $\mathbf{W}\overrightarrow{G}$, the non-constant term, in the left hand side of the equation. We split $\mathbf{W}$ into $\mathbf{W}_1 x + \mathbf{W}_2 y$, where each $\mathbf{W}_i$ is a binary matrix.Then we remark that the $2s$-dimensional vector $(\overrightarrow{w_1}, \overrightarrow{w_2}) := (\mathbf{W}_1\overrightarrow{G}, \mathbf{W}_2\overrightarrow{G})$ is generated by r ciphertexts G_i's. Since there are r degree of freedom for $2s$-dimensional vectors, there must be $(2s-r)$ independent relations among the entries of $(\overrightarrow{w_1}, \overrightarrow{w_2})$. We write such relations as

$$\pi(\overrightarrow{w_1}, \overrightarrow{w_2}) := \mathbf{P}\overrightarrow{w_1} + \mathbf{Q}\overrightarrow{w_2} = 0, \tag{15}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are $(2s - r) \times s$-dimensional binary matrices and $\big[\mathbf{P} \mid \mathbf{Q}\big]$ is of full rank. Since the relation $\pi(\overrightarrow{w_1}, \overrightarrow{w_2}) = 0$ holds for any randomly chosen $\overrightarrow{G}$, we also have, by abusing the notation, $\pi(\mathbf{W}_1, \mathbf{W}_2) := \mathbf{P}\mathbf{W}_1 + \mathbf{Q}\mathbf{W}_2 = \mathbf{0}$.

Note that in the case of half-gate scheme, where $s = 1$ and $r = 2$, this step can be omitted because there are zero relations.

Find solutions. Now we describe how we construct the equation. In the previous paragraph, we saw that the x-coefficient matrix $\mathbf{W}_1$ and the y-coefficient matrix $\mathbf{W}_2$ of the left-hand side satisfy the relation $\pi(\mathbf{W}_1, \mathbf{W}_2) = \mathbf{0}$, where $\mathbf{P}$ and $\mathbf{Q}$ are determined by $\mathbf{V}$. On the right hand side, by the construction of $\mathbf{M}$ and $\mathbf{V}$, it is obvious that the coefficients of x and y in $\mathbf{M}\overrightarrow{H}$ satisfy the same relation. By the linearity, it remains to choose $(\mathbf{R}_A, \mathbf{R}_B)$ so that the relation holds. Since the relation should hold for each of arbitrarily chosen random $\overrightarrow{A}$, $\overrightarrow{B}$ and $\overrightarrow{\Delta}$, we deduce that $(\mathbf{R}_A, \mathbf{R}_B)$ satisfies the followings:

$$\begin{aligned}\pi(\mathbf{R}_{A,0}, \mathbf{R}_{A,1}) &= \mathbf{0}, \\ \pi(\mathbf{R}_{B,0}, \mathbf{R}_{B,1}) &= \mathbf{0}, \\ \pi\big(\mathbf{R}_{A,0} + \mathbf{R}_{A,2} + \beta\mathbf{I}_s,\ \mathbf{R}_{B,1} + \mathbf{R}_{B,2} + \alpha\mathbf{I}_s\big) &= \mathbf{0}.\end{aligned} \tag{16}$$

As a side note, we check that the solution space of $(\mathbf{R}_A, \mathbf{R}_B)$ is isomorphic to a $(3rs - s^2)$-dimensional binary space. To see this, among $6s^2$ binary entries of

$(\mathbf{R}_A, \mathbf{R}_B)$, we have s^2 relations from (14) and $(2s-r)\cdot s$ relations for each equation in (16). Thus, the solution space is of dimension $6s^2 - (3s(2s-r)+s^2) = 3sr - s^2$.

Example 1. Recall that $\mathbf{V} = [\ \mathbf{I}_2 \mid \mathbf{W}\ ] = \left[\begin{array}{cc|ccc} 1 & 0 & x & 0 & x+y \\ 0 & 1 & 0 & y & x+y \end{array}\right]$ in RR21's construction. Then we can write $\mathbf{W} = \begin{bmatrix} 1\,0\,1 \\ 0\,0\,1 \end{bmatrix} x + \begin{bmatrix} 0\,0\,1 \\ 0\,1\,1 \end{bmatrix} y = \mathbf{W}_1 x + \mathbf{W}_2 y$. In this case, we obtain $\mathbf{P} = \begin{bmatrix} 0\ 1 \end{bmatrix}$ and $\mathbf{Q} = \begin{bmatrix} 1\ 0 \end{bmatrix}$ satisfying $\pi(\mathbf{W}_1, \mathbf{W}_2) = \mathbf{0}$.

If we find $(\mathbf{R}_A, \mathbf{R}_B)$ using (14) and (16), we have

$$\mathbf{R}_{A,0} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \qquad \mathbf{R}_{A,1} = \begin{bmatrix} a_3 & a_4 \\ b_3 & b_4 \end{bmatrix}, \qquad \mathbf{R}_{A,2} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$
$$\mathbf{R}_{B,0} = \begin{bmatrix} a_3+1 & a_4 \\ b_3 & b_4+1 \end{bmatrix}, \mathbf{R}_{B,1} = \begin{bmatrix} b_3 & b_4+1 \\ e_3 & e_4 \end{bmatrix}, \mathbf{R}_{B,2} = \begin{bmatrix} f_1 & f_2 \\ f_3 & f_4 \end{bmatrix},$$

where $f_1 = a_3 + b_3 + c_3 + \alpha$ and $f_2 = a_4 + b_4 + c_4 + \beta + 1$ and the entries are all binary elements. One might observe that $(\mathbf{R}_A, \mathbf{R}_B)$ is 14-dimensional space (it coincides with the result in [14]). As an example, one might check that the matrix $\mathbf{R}_p$ in [14, Figure 4] is obtained by setting all free variables as zero except $b_4 = c_4 = 1$ when $(\alpha, \beta) = (1, 1)$.

5.4 Proof of Lower Bounds on Garbled Circuits

Previously, we have found the solutions $(\mathbf{R}_A, \mathbf{R}_B)$ satisfying (Step 1)$\sim$(Step 3). It remains to check whether the solution satisfies (Step 4). Our aim is to show that in order for the found solutions to satisfy (Step 4), it is necessary to have $3s \leq 2r$. This demonstrates that any garbled gate for 2-degree Boolean functions consists of at least 1.5κ bits, which supports the optimality of RR21's garbling scheme.

We argue that the marginal view at $(x, y) = (0, 0)$ leaks information on α and β when $3s \geq 2r$. This shows the necessary condition for (Step 4). The following theorem is a prerequisite on such necessary conditions of $\mathbf{P}$ and $\mathbf{Q}$.

Theorem 2. *Define the matrices* $\mathbf{M}$ *and* $\mathbf{V}$ *as Section 5.2.1. Consider Equation* (13) *that satisfies Equation* (14) *and* (16)*, where* π *is defined as Equation* (15)*. Assume that the matrices* $\mathbf{M}$*,* $\mathbf{V}$ *and therefore* $\mathbf{P}$*,* $\mathbf{Q}$ *are given. Given the matrices* $(\mathbf{R}_{A,2}, \mathbf{R}_{B,2})$*, if any of* $\mathbf{P}$*,* $\mathbf{Q}$ *and* $\mathbf{P} + \mathbf{Q}$ *are not of full rank, then either the bit* α*,* β*, or both* α *and* β*, can be computed.*

Proof. Suppose that $\mathbf{P}$ is not of full rank. Then, there exists a $(2s-r)$-dimensional nonzero binary vector $\overrightarrow{k_1}$ such that $\overrightarrow{k_1}^\top \mathbf{P} = 0$. The value $\overrightarrow{k_1}^\top \mathbf{Q}$ must be nonzero since $\begin{bmatrix} \mathbf{P} \mid \mathbf{Q} \end{bmatrix}$ is of full rank. Combining (14) and (16) implies

$$\pi\big(\mathbf{R}_{A,0} + \mathbf{R}_{A,1} + \mathbf{R}_{A,2} + (\beta+1)\mathbf{I}_s, \mathbf{R}_{B,2} + \alpha\mathbf{I}_s\big) = \mathbf{0}. \tag{17}$$

and yields $\overrightarrow{k_1}^\top \mathbf{Q}(\mathbf{R}_{B,2} + \alpha\mathbf{I}_s) = 0$ by multiplying $\overrightarrow{k_1}^\top$ in both sides. Henceforth, it holds that $\overrightarrow{k_1}^\top \mathbf{Q}\mathbf{R}_{B,2} = \alpha\big(\overrightarrow{k_1}^\top \mathbf{Q}\big)$. By computing the left-hand side, the bit α is determined by whether the result is zero or not.

By a similar argument, if $\mathbf{Q}$ is not of full rank, we have $\overrightarrow{k_2}^\top \mathbf{P}(\mathbf{R}_{A,2} + \beta \mathbf{I}_s) = 0$, where $\overrightarrow{k_2}$ is a nonzero vector such that $\overrightarrow{k_2}^\top \mathbf{Q} = 0$. It comes from that $\pi\big(\mathbf{R}_{A,2} + \beta \mathbf{I}_s, \mathbf{R}_{B,0} + \mathbf{R}_{B,1} + \mathbf{R}_{B,2} + (\alpha+1)\mathbf{I}_s\big) = \mathbf{0}$. In this case, the bit β is computed by checking whether $\overrightarrow{k_2}^\top \mathbf{P}\mathbf{R}_{A,2}$ is zero or not.

Finally, let us consider the case when $\mathbf{P} + \mathbf{Q}$ is not of full rank. Combining (17) with $\pi(\mathbf{R}_{A,0}, \mathbf{R}_{A,1}) = \mathbf{0}$ yields $\pi\big(\mathbf{R}_{A,1} + \mathbf{R}_{A,2} + (\beta+1)\mathbf{I}_s, \mathbf{R}_{A,1} + \mathbf{R}_{B,2} + \alpha \mathbf{I}_s\big) = \mathbf{0}$. Multiplying a nonzero vector $\overrightarrow{k_3}$ such that $\overrightarrow{k_3}^\top(\mathbf{P} + \mathbf{Q}) = 0$, we have $\overrightarrow{k_3}^\top \big(\mathbf{P}\big(\mathbf{R}_{A,2} + (\beta+1)\mathbf{I}_s\big) + \mathbf{Q}\big(\mathbf{R}_{B,2} + \alpha \mathbf{I}_s\big)\big) = 0$. Since we have $\overrightarrow{k_3}^\top \big(\mathbf{P}\big(\mathbf{R}_{A,2} + \mathbf{I}_s\big) + \mathbf{Q}\mathbf{R}_{B,2}\big) = \overrightarrow{k_3}^\top \big(\beta \mathbf{P} + \alpha \mathbf{Q}\big)$, comparing the left-hand side with the right-hand side for all possible $(\alpha, \beta) \in \{0,1\}^2$ allows us to reveal (α, β). □

Let us briefly describe the meaning of the above theorem in terms of garbling schemes. Consider a garbling construction with the garbling equation (13). Applying the dicing technique, if necessary, we may assume that the value of $(\mathbf{R}_A, \mathbf{R}_B)$ at (i, j) is known to the evaluator, where A^i and B^j are the input wire labels held by the evaluator. Then the above theorem asserts that the marginal view at $(0, 0)$ leaks private information on α and β whenever any of $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{P} + \mathbf{Q}$ are not full row rank. Thus, it violates the privacy property of the scheme.

Let us denote the rank of a matrix $\mathbf{A}$ by $\mathrm{rk}(\mathbf{A})$. We are now ready to show the desired result.

Theorem 3. *Let $\Pi := (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be a s-linear garbling scheme over $\mathbb{K}$ that is free-XOR compatible. For Π to satisfy the privacy property, it must satisfy $3s \leq 2r$. In other words, the garbled gate from Π is of at least $(3/2)\kappa + ct$ bits.*

Recall that ct is the size of the auxiliary ciphertexts from the dicing technique. In most case, a garbling scheme can be constructed so that $ct = O(1)$. If it is the case, one can say that any garbled gate of s-linear garbling schemes is of at least $1.5\kappa + O(1)$ bits.

Proof. For simplicity, this proof only focuses on the case when $\mathbb{K} = \mathbb{F}_2$. Refer to Appendix D for general $\mathbb{K}$.

For the correctness of Π, the algorithm Ev on input x and y should satisfy $\overrightarrow{C} + (x+\alpha)(y+\beta)\overrightarrow{\Delta} = (\mathbf{V}^{\mathrm{kap}}_{x,y} + \mathbf{V}^{\mathrm{con}}_{x,y})\overrightarrow{T}^{\mathrm{kap}}$. Since the scheme is s-linear, we may assume that it can be rewritten as Equation (8), where $\mathbf{M}$, $\mathbf{W}$, $\mathbf{R}_A$ and $\mathbf{R}_B$ are described as above.

In order for Π to satisfy the privacy, we may assume that the matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{P}+\mathbf{Q}$ are of full rank by Theorem 2, i.e. $\mathrm{rk}(\mathbf{P}) = \mathrm{rk}(\mathbf{Q}) = \mathrm{rk}(\mathbf{P}+\mathbf{Q}) = 2s-r$. Recall that $\mathbf{V} = [\mathbf{I}_s \mid \mathbf{W}] = [\mathbf{I}_s \mid \ell_1 \overrightarrow{u_1} \mid \cdots \mid \ell_r \overrightarrow{u_r}]$, where $\overrightarrow{u_i}$ is a s-dimensional binary vector and $\ell_i \in \{x, y, x+y\}$ for $i = 1, \ldots, r$. If we write $\ell_i = \delta_i x + \epsilon_i y$, we obtain $\mathbf{W} = \mathbf{W}_1 x + \mathbf{W}_2 y$ such that

$$\mathbf{W}_1 = \big[\delta_1 \overrightarrow{u_1} \mid \cdots \mid \delta_r \overrightarrow{u_r}\big] \text{ and } \mathbf{W}_2 = \big[\epsilon_1 \overrightarrow{u_1} \mid \cdots \mid \epsilon_r \overrightarrow{u_r}\big].$$

Consider the sets S_x, S_y and S_{x+y} defined by $S_f := \{i : \ell_i = f\}$ for $f \in \{x, y, x+y\}$. Then the set $\{S_x, S_y, S_{x+y}\}$ is a partition of $\{1, 2, \ldots, r\}$. Note that

the vectors in $\{\overrightarrow{u_i} : i \in S_x\}$ are linearly independent. This is due to the fact that $\{x\overrightarrow{u_i} : i \in S_x\}$ consists of column vectors of $\mathbf{V}$ and hence is linearly independent. Similarly, the sets $\{\overrightarrow{u_i} : i \in S_y\}$ and $\{\overrightarrow{u_i} : i \in S_{x+y}\}$ are linearly independent.

By the definition, it holds $\mathbf{PW}_1 + \mathbf{QW}_2 = 0$ and it is obvious that $\mathbf{P}(\delta_i \overrightarrow{u}_i) + \mathbf{Q}(\epsilon_i \overrightarrow{u}_i) = \mathbf{P}\overrightarrow{u}_i = 0$ for $i \in S_x$. In other words, each row of $\mathbf{P}$ belongs to an s-dimensional subspace orthogonal to the space spanned by $\{\overrightarrow{u_i} : i \in S_x\}$. Since $\{\overrightarrow{u_i} : i \in S_x\}$ is linearly independent, the dimension of the row space of $\mathbf{P}$ is less than or equal to $s - \#S_x$, where $\#S_x$ denotes the cardinality of S_x. Equivalently, $\mathrm{rk}(\mathbf{P}) = 2s - r \leq s - \#S_x$.

Similarly we have $\mathrm{rk}(\mathbf{Q}) \leq s - \#S_y$ and $\mathrm{rk}(\mathbf{P} + \mathbf{Q}) \leq s - \#S_{x+y}$. Summing up these three inequalities leads us $\mathrm{rk}(\mathbf{P}) + \mathrm{rk}(\mathbf{Q}) + \mathrm{rk}(\mathbf{P} + \mathbf{Q}) = 6s - 3r \leq 3s - \big(\#S_x + \#S_y + \#S_{x+y}\big) = 3s - r$, i.e. $3s \leq 2r$, the desired result. □

6 Conclusion and Further Discussions

We conclude by suggesting several open questions.

Garbling Multivariate High Degree Functions. In [11], they discussed on extension of linear garbling model and constructing garbling schemes for Boolean functions with high degree/multiple inputs. Precisely, they argued that garbling n-variate Boolean polynomials of degree d requires $\sum_{i=1}^{d-1} \binom{n}{i}\kappa$ bits. In the case of $n = d = 2$ (e.g. AND gates), it has the same size as the half-gate garbling scheme.

Our new model of linear garbling presented in Section 4.2 has been defined for two inputs, but can be naturally extended to cover high-degree/multi-input case. Also, the garbling equation can be easily extended for multivariate Boolean gates of degree $d > 2$. For instance, consider an example of three-variate Boolean gates of degree 3, such as xyz. Restricting our consideration on $s = 1$, we have a garbling equation for xyz in the following form:

$$\mathbf{V}\begin{pmatrix} C \\ \overrightarrow{G} \end{pmatrix} = \mathbf{M}\overrightarrow{H} + \mathbf{R}_A(A + x\Delta) + \mathbf{R}_B(B + y\Delta) + \mathbf{R}_\Gamma(\Gamma + z\Delta) + (x+\alpha)(y+\beta)(z+\gamma)\Delta.$$

By similar arguments in the previous sections, we may define $\overrightarrow{H}$ as an 18-dimensional vector using $H(A^i)$, $H(B^j)$, $H(\Gamma^k)$, $H(A^i, B^j)$, $H(B^j, \Gamma^k)$ and $H(\Gamma^k, A^i)$ for $i, j, k \in \{0, 1\}$ and set $\mathbf{M}$ as a matrix consisting of polynomials of degree ≤ 2 according to the choice of $\overrightarrow{H}$. We have $\mathbf{V} = \left[\, 1 \;\middle|\; x\ y\ z \;\middle|\; xy\ yz\ zx \,\right]$ and there are total 6 non-constant coefficients. Thus, we may have a garbling scheme with 6 ciphertexts by finding $\mathbf{R}_A$, $\mathbf{R}_B$ and $\mathbf{R}_\Gamma$ so that both sides of the above equation belongs to the same space. See that $\mathbf{R}_A = yz$ and $\mathbf{R}_B = \mathbf{R}_\Gamma = 0$ are one of such solutions.

However, one might see that this construction is no better than repeatedly applying the half-gate scheme (not even RR21 construction!) to a circuit composed of multiple AND/XOR gates that represents the given Boolean polynomial. Thus, we may ask several questions. Can we obtain any construction for garbling

multivariate/high-degree polynomials whose size is smaller than using RR21's construction? For instance, can we find solutions for the above garbling equation with $s \geq 2$ that yields a construction for secure garbling schemes? Or, can we prove a lower bound for garbling n-variate polynomials of degree d where $n \geq 2$ and $d \geq 2$?

Garbling General Arithmetic Functions. While our discussion mostly focused on Boolean functions, one might consider extension of our approach to garble arithmetic circuits, e.g. addition/multiplication modulo m for $m > 2$. Several works such as [1] and [11] observed that generalizing free-XOR technique directly allows a technique that garbles an addition modulo m for free. In addition, [11] considered a construction for garbling multiplication modulo m with ciphertexts of size $2(m-1)\kappa$ bits, whereas a typical algorithm such as the half-gate scheme would require $O(\log^2 m)\kappa$ bits.

As before, our approach can provide a simpler description of [11]. For instance, in the case of $m = 3$, we set $\mathbf{M} = \big[f_0(x)\ f_1(x)\ f_2(x) \mid f_0(y)\ f_1(y)\ f_2(y)\big]$, where f_i is a polynomial of degree 2 over $\mathbb{F}_3$ such that $f_i(x) = 1$ if $x = i$ and 0 otherwise. Then we have $\mathbf{V} = [1 \mid x\ x^2 \mid y\ y^2]$. We can easily find solutions for the garbling equation with the above matrices. As a result, this yields a garbling scheme with $4 = 2(m-1)$ ciphertexts.

Again, we might ask a question whether we can improve this construction with our approach using $s \geq 2$. For instance, after slicing the wire labels, can we reduce the size of garbled gates for arithmetic circuits? Can we have any result on the lower bound as before?

Privacy-Free Garbling. Frederiksen, Nielsen, and Orlandi [5] introduced privacy-free garbling schemes that only satisfy authenticity property, and do not require privacy property. They showed that such garbling schemes are useful in some special applications such as zero-knowledge protocols.

In this paper, our result on the lower bound is only meaningful to garbled circuits with privacy property. Is it possible to prove a lower bound for privacy-free garbling schemes? In our model, the size of garbled gates is at least κ bits anyway, since our setting intrinsically requires $r \geq s$. Then can we say that the current state-of-the-art privacy-free scheme described by [16] is optimal?

References

1. M. Ball, T. Malkin, and M. Rosulek. Garbling gadgets for boolean and arithmetic circuits. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 565–577. ACM, 2016.
2. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In H. Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990.

3. M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In T. Yu, G. Danezis, and V. D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.
4. L. Fan, Z. Lu, and H. Zhou. Column-wise garbling, and how to go beyond the linear model. *IACR Cryptol. ePrint Arch.*, page 415, 2024.
5. T. K. Frederiksen, J. B. Nielsen, and C. Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 191–219. Springer, 2015.
6. S. Gueron, Y. Lindell, A. Nof, and B. Pinkas. Fast garbling of circuits under standard assumptions. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 567–578. ACM, 2015.
7. R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.
8. C. Kempka, R. Kikuchi, and K. Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 967–997, 2016.
9. V. Kolesnikov, P. Mohassel, and M. Rosulek. Flexor: Flexible garbling for XOR gates that beats free-xor. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 440–457. Springer, 2014.
10. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.
11. T. Malkin, V. Pastro1, and abhi shelat. An algebraic approach to garbling. Unpublished manuscript, 2016. `https://simons.berkeley.edu/talks/tal-malkin-2015-06-10`.
12. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In S. I. Feldman and M. P. Wellman, editors, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 129–139. ACM, 1999.
13. B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009.

14. M. Rosulek and L. Roy. Three halves make a whole? beating the half-gates lower bound for garbled circuits. In T. Malkin and C. Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 94–124. Springer, 2021.
15. A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.
16. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer, 2015.

A How to Randomize the Control Bits

In this section, we explain how the dicing technique works. It turns out that any garbling construction from our algebraic perspective can be used to randomize the control bits as well. We note that the technique in [14] can be seen as a particular case of ours.

For the sake of readability, we mainly describe the technique with the example of RR21's construction, then we explain how this works in general case.

Let us recall Example 1. At the beginning of the dicing technique, the garbler chooses $(\mathbf{R}_A, \mathbf{R}_B)$ at random among 2^{14} possible choices. Assume that the choice is

$$\mathbf{R} = [\mathbf{R}_A|\mathbf{R}_B] = \left[\begin{array}{cc|cc} 0 & 0 & x+\alpha & y+\beta+1 \\ 0 & 0 & y & x \end{array}\right].$$

It is chosen by setting all the free variables zero except $e_3 = 1$.

To send the information on $\mathbf{R}$, the garbler encrypts it column by column. More precisely, say $\mathbf{R} = [\overrightarrow{r_1}, \ldots, \overrightarrow{r_4}]$, where $\overrightarrow{r_k}$ is the k-th column of $\mathbf{R}$. The algorithm Gb makes random oracle queries and define

$$\overrightarrow{S}^{con} := \left(H^c(A^0), H^c(A^1), H^c(B^0), H^c(B^1), H^c(A^0+B^0), H^c(A^0+B^1)\right)^\top,$$

where H^c is a random oracle that returns an 1-bit string (it is usually chosen as the least significant bit of outputs by the random oracle that has κ/s-bit range, i.e. H^{kap} in Item 1, Section 4.2).

Given the column $\overrightarrow{r_k}$ for each k, choose $\overrightarrow{z_k} := (z_{k1}, \ldots, z_{k5})^\top$ such that

$$\mathbf{V}\overrightarrow{z_k} = \mathbf{M}\overrightarrow{S}^{con} + \overrightarrow{r_k}, \tag{18}$$

where $\mathbf{V}$ and $\mathbf{M}$ are the same as Equation (7). Then it returns the vector $\overrightarrow{z_k}$ which comprises the ciphertexts encrypting $\overrightarrow{r_k}$.

For instance, let us take an example of $\overrightarrow{r_3} = (x + \alpha, y)^\top$. By comparing both sides, we have

$$\begin{aligned} z_{31} &= H^c(A^0) + H^c(A^0 + B^0) + \alpha \\ z_{32} &= H^c(B^0) + H^c(A^0 + B^0) \\ z_{33} &= H^c(A^0) + H^c(A^1) + 1 \\ z_{34} &= H^c(B^0) + H^c(B^1) + 1 \\ z_{35} &= H^c(A^0 + B^0) + H^c(A^0 + B^1). \end{aligned}$$

Let $\mathbf{V}_{ij}$ be the value of $\mathbf{V}$ evaluated at $(x, y) = (i, j)$. Upon receiving $\overrightarrow{z_k}$, on input A^i and B^j, the algorithm Ev computes

$$\overrightarrow{\widetilde{r}_k} = \mathbf{V}_{ij}\overrightarrow{z_k} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} H(A^i) \\ H(B^j) \\ H(A^i + B^j) \end{bmatrix}.$$

It is easily verified that $\overrightarrow{\widetilde{r}_k}$ is the value of $\overrightarrow{r_k}$ evaluated at $(x, y) = (i, j)$, i.e. the marginal view.

We observe that the above argument works in general not only for RR21's construction. Actually, the control bit randomization is carried out by encrypting each columns of $\mathbf{R}$, the randomly chosen control bits. Moreover, it is encrypted via *the same garbling equation* as that used for the original garbling construction. In other words, the matrices $\mathbf{M}$ and $\mathbf{V}$ in Equation (18) are the same as the original garbling equation. The only condition for the control bits encryption to work, it suffices to see whether $\overrightarrow{r_k}$ belongs to the same space spanned by the columns of $\mathbf{M}$ or $\mathbf{V}$. And it turns out to be equivalent that $\overrightarrow{r_k}$ satisfies the relation π in Section 5. Recall that $\overrightarrow{r_k}$ is the column of $\mathbf{R}$. By (16), we see that $\mathbf{R}$, thus each of its columns, satisfies the relation π which is the desired result. Henceforth, we argue that the control bit randomization is always possible with its original garbling equation.

To help readers' understanding, let us call back the previous example of RR21. In this case, the relation π is equivalent to say that the y-coefficient on the top is the same as the x-coefficient of the bottom. We see that, for each $\overrightarrow{r_k}$, it satisfies the condition.

Let us consider why this technique does not reveal the information on α and β. We see that the entire value of $\mathbf{R}$ will definitely disclose the permute bits. Observe that $\overrightarrow{z_k}$'s are encrypting the coefficients of the polynomials in $\mathbf{R}$ using $\overrightarrow{S}^{con}$. And the decryption only reveals the value of the polynomials in $\mathbf{R}$ evaluated at $(x, y) = (i, j)$. Without knowing the wire labels other than A^i and B^j, the evaluator cannot evaluate the polynomials outside of (i, j). Thus, it does not disclose the entire information on $\mathbf{R}$.

Remark 3. One might observe that our example of RR21 is slightly different from their original description. Their description is intrinsically reducing the number of ciphertexts $\overrightarrow{z}_k$'s encrypting the control bits by choosing the control bits $\mathbf{R}$ from a small subspace rather than the full space of dimension 14. By a tedious computation, we see that $\mathbf{R}$ in [14, Figure 3.] is actually determined by

only the first column. Setting $\overrightarrow{r}_1 := (r_{11}, r_{12})^\top = (ax + by + c, bx + dy + e)^\top$ for randomly chosen $a, b, c, d, e \in \mathbb{F}_2$, we check that the matrix $\mathbf{R}$ is the same as

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{12} + (x+1) & r_{11} + r_{12} \\ r_{12} & r_{11} + r_{12} + (y+1) & r_{11} + r_{12} & r_{11} \end{bmatrix}.$$

Thus, it is enough to send only the encryption of $\overrightarrow{r}_1$, instead of sending entire encryptions of all columns. Therefore, it reduces the number of ciphertexts garbling the control bits.

B RR21 is 2-Linear

In this section, we show that RR21's construction is 2-linear in the sense of Section 4.2. We remark that any projection map is a linear map. Thus, we shall simply write any projected element without full description of its linear map. For instance, in RR21 construction, we write wire labels as of the form $A = (A_L \| A_R)$. Given a matrix $\mathbf{R}$ over $\mathbb{F}_2[x, y]$, denote $\mathbf{R}^{(i,j)}$ by the value of $\mathbf{R}$ at $(x, y) = (i, j)$. The k-row of the matrix $\mathbf{R}$ is denoted by $[\mathbf{R}]_k$.

The following can be derived from the garbling equation (7). For instance, $(C_L^0, C_R^0)^\top$ is obtained by evaluating Equation (7) at $(x, y) = (0, 0)$. Similarly, (G_0, G_1) is derived by adding the value at $(0, 0)$ and at $(1, 1)$. The algorithm Gb can be written as follows:

$$\begin{bmatrix} A_L^x \\ A_R^x \\ B_L^y \\ B_R^y \\ \hline C_L^0 \\ C_R^0 \\ \hline C_L^1 \\ C_R^1 \\ \hline G_0 \\ G_1 \\ \hline G_2 \end{bmatrix} = \left[\begin{array}{cc|cc|cc|c} 1 & 0 & 0 & 0 & x & 0 & 0\,0\,0\,0\,0\,0 \\ 0 & 1 & 0 & 0 & 0 & x & 0\,0\,0\,0\,0\,0 \\ 0 & 0 & 1 & 0 & y & 0 & 0\,0\,0\,0\,0\,0 \\ 0 & 0 & 0 & 1 & 0 & y & 0\,0\,0\,0\,0\,0 \\ \hline \multicolumn{2}{c|}{\mathbf{R}_A^{(0,0)}} & \multicolumn{2}{c|}{\mathbf{R}_B^{(0,0)}} & \multicolumn{2}{c|}{\alpha\beta\mathbf{I}_2} & \begin{matrix}1\,0\,0\,0\,1\,0\\0\,0\,1\,0\,1\,0\end{matrix} \\ \hline \multicolumn{2}{c|}{\mathbf{R}_A^{(0,0)}} & \multicolumn{2}{c|}{\mathbf{R}_B^{(0,0)}} & \multicolumn{2}{c|}{(1+\alpha\beta)\mathbf{I}_2} & \begin{matrix}1\,0\,0\,0\,1\,0\\0\,0\,1\,0\,1\,0\end{matrix} \\ \hline \multicolumn{2}{c|}{\mathbf{R}_A^{(0,0)} + \mathbf{R}_A^{(1,1)}} & \multicolumn{2}{c|}{\mathbf{R}_B^{(0,0)} + \mathbf{R}_B^{(1,1)}} & \multicolumn{2}{c|}{\mathbf{R}_A^{(1,1)} + \mathbf{R}_B^{(1,1)} + (\alpha+\beta)\mathbf{I}_2} & \begin{matrix}1\,1\,0\,0\,0\,0\\0\,0\,1\,1\,0\,0\end{matrix} \\ \hline \multicolumn{2}{c|}{[\mathbf{R}_A^{(0,0)} + \mathbf{R}_A^{(0,1)}]_1} & \multicolumn{2}{c|}{[\mathbf{R}_B^{(0,0)} + \mathbf{R}_B^{(0,1)}]_1} & \multicolumn{2}{c|}{[\mathbf{R}_B^{(0,1)} + \alpha\mathbf{I}_2]_1} & 0\,0\,0\,0\,1\,1 \end{array}\right] \underbrace{\begin{bmatrix} A_L^0 \\ A_R^0 \\ B_L^0 \\ B_R^0 \\ \Delta_L \\ \Delta_R \\ H(A^0) \\ H(A^1) \\ H(B^0) \\ H(B^1) \\ H(A^0 + B^0) \\ H(A^0 + B^1) \end{bmatrix}}_{\overrightarrow{S}^{kap}}.$$

With the same notation as Section A, for each k, we can check that

$$\underbrace{\begin{bmatrix} z_{k1} \\ z_{k2} \\ z_{k3} \\ z_{k4} \\ z_{k5} \end{bmatrix}}_{\overrightarrow{z}\text{-part}} = \underbrace{\left[\begin{array}{cc|cc|cc|cc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]}_{\mathbf{z}\text{-part}} \underbrace{\left[\begin{array}{c} r_{k1}^{(0,0)} \\ r_{k2}^{(0,0)} \\ \hline r_{k1}^{(0,1)} \\ r_{k2}^{(0,1)} \\ \hline r_{k1}^{(1,0)} \\ r_{k2}^{(1,0)} \\ \hline r_{k1}^{(1,1)} \\ r_{k2}^{(1,1)} \\ \hline H^c(A^0) \\ H^c(A^1) \\ H^c(B^0) \\ H^c(B^1) \\ H^c(A^0+B^0) \\ H^c(A^0+B^1) \end{array}\right]}_{(\overrightarrow{r}\|\overrightarrow{S}^{con})\text{-part}}.$$

We leave it to readers to check that the algorithm Ev also satisfies our definition.

C Non s-Linear Schemes

In this section, we briefly describe why the scheme of [8] is not linear in our sense. In their scheme, the garbler chooses $A^\alpha, B^\beta, \Delta \in \mathbb{F}_{2^\kappa}$ at random. Here, A^α and B^β are the input wire labels that represent **false**. Then the other wire label is decided by $A^{\alpha+1} = A^\alpha +_{\mathbb{Z}} \Delta$ and $B^{\beta+1} = B^\beta +_{\mathbb{Z}} \Delta$, where $+_{\mathbb{Z}}$ denotes the addition over $\mathbb{Z}_{2^\kappa}$ and the operands are considered as elements in $\mathbb{Z}_{2^\kappa}$. The random oracle responses are $H_1 = H(A^\alpha +_{\mathbb{Z}} B^\beta)$, $H_2 = H(A^\alpha +_{\mathbb{Z}} B^\beta +_{\mathbb{Z}} \Delta)$, and $H_3 = H(A^\alpha +_{\mathbb{Z}} B^\beta +_{\mathbb{Z}} 2\Delta)$. They defined $\overrightarrow{S}^{kap} := (A^\alpha, B^\beta, \Delta, H_1, H_2, H_3) \in \mathbb{F}_{2^\kappa}^6$.

At this stage, we remark that there seems to be no efficient way to represent $A^{\alpha+1}$ as $\mathbf{A}^{\alpha+1}\overrightarrow{S}^{kap}$ for a matrix $\mathbf{A}^{\alpha+1}$ over $\mathbb{F}_{2^\kappa}$. It should be alerted that $(1,0,1,0,0,0)\overrightarrow{S}^{kap}$ is not a right way to represent $A^{\alpha+1}$, since the addition here is over $\mathbb{F}_{2^\kappa}$ while the desired operation should be carried out over $\mathbb{Z}_{2^\kappa}$. In this sense, it is misleading that the authors in [8, Section 6.2] represented as $A^{\alpha+1} = (1,0,1,0,0,0)\overrightarrow{S}^{kap}$.

Although their scheme is not linear in the above sense, we can still represent their scheme using our algebraic approach. Set $X = x + \alpha$ and $Y = y + \beta$. Define

$$\mathbf{M} = \left[(X+1)(Y+1)\ (X+1)Y\ X(Y+1)\ XY\right] \text{ and } \mathbf{V} = \left[XY+1\ XY\ f_{a,d}(X,Y)\right],$$

where $f_{a,d}(X,Y) := aXY + X + Y + d$ and $a, d \in \mathbb{F}_2$. Then the output wire label C^0, C^1 and the ciphertext G are chosen so that $\mathbf{V}(C^0, C^1, G)^\top = \mathbf{M}(H_1, H_2, H_2, H_3)^\top$. By comparing and properly eliminating both sides, we have

$$C^0 = dG + H_1;\ C^0 = (d+1)G + H_2;\ \text{and } C^1 = (a+d)G + H_3.$$

It means that when the evaluator holds A^α and B^β, then she can recover C^0 from the first equation. Similarly, when the input labels are $(A^\alpha, B^{\beta+1})$ or $(A^{\alpha+1}, B^\beta)$, then the output C^0 is obtained from the second equation. The last equation is used when the inputs are $(A^{\alpha+1}, B^{\beta+1})$.

Note that, in order for the evaluator to properly carry out the computation, she must know the value of either d, $d+1$, or $a+d$ accordingly with the input labels. Here is where the dicing technique is applied. Note that $d = f_{a,d}(0,0)$, $d+1 = f_{a,d}(0,1) = f_{a,d}(1,0)$, and $a+d = f_{a,d}(1,1)$. If the garbler computes and sends $z_{X,Y} := f_{a,d}(X,Y) + \mathrm{lsb}(H(A^{X+\alpha}, B^{Y+\beta}))$, then the evaluator is able to recover only the desired bit by decrypting $z_{X,Y}$ with her inputs A^x and B^y.

D Proof of Theorem 3 for general $\mathbb{K}$

Sketch of proof. In general case, we have a random oracle response followed by a multiplication by $\mathbb{K}$. For instance, one might compute a sliced wire labels by applying linear operations such as $a_1 H(A^i) + a_2 H(A^i + B^j) + \cdots$ for some $a_1, a_2 \in \mathbb{F}_{2^\kappa}$.

In such case, by using a similar argument to Section 5.2.1, we might write $\mathbf{M}$ as

$$\mathbf{M} = \left[a_1(\ell_1+1)\overrightarrow{u_1} \mid a_1\ell_1\overrightarrow{u_1} \mid \cdots \mid a_n(\ell_n+1)\overrightarrow{u_n} \mid a_n\ell_n\overrightarrow{u_n}\right], \tag{19}$$

where $a_i \in \mathbb{K}$ and ℓ_i and $\overrightarrow{u_i}$ are defined as Equation (11).

Now, let us consider $\mathbf{V}$. Recall that it is a basis of the column space of $\mathbf{M}$ over $\mathbb{K}$. One can check that the same $\mathbf{V}$ as Equation (12) generates the column space of $\mathbf{M}$. In the rest of arguments, since we only use $\mathbf{V}$, it still holds for this case.