Analysis on Sliced Garbling via Algebraic Approach [⋆]

Taechan Kim

Independent Researcher, Seoul, Korea
tckim1458@gmail.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). Recently, Ashur, Hazay, and Satish (eprint 2024/389) proposed a scheme that requires $4/3\kappa + O(1)$ bits for garbling AND gates. Precisely they extended the idea of *slicing* introduced by Rosulek and Roy to garble 3-input gates of the form $g(u, v, w) := u(v + w)$. By setting $w = 0$, it can be used to garble AND gates with the improved communication costs. However, in this paper, we observe that the scheme proposed by Ashur, Hazy, and Satish leaks information on the permute bits, thereby allowing the evaluator to reveal information on the private inputs. To be precise, we show that in their garbling scheme, the evaluator can compute the bits α and $\beta + \gamma$, where α, β, and γ are the private permute bits of the input labels A, B, and C, respectively.

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 [16], one line of recent research [4,13,12,14,11,7,17,15] 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 [15] (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

[⋆] In this work, we present an attack on a new garbling scheme proposed by Ashur, Hazay, and Satish [1]. Concurrently to our work, we noticed that Fan, Lu, and Zhou [6] also described an attack on their scheme.

the size of garbled circuits, which is obtained in a model called *linear garbling* defined by Zahur, Rosulek, and Evans [17]. Their optimization was made possible by a new technique, called *slicing-and-dicing*, that is beyond the definition of the linear garbling model.

Following to the previous works, Ashur, Hazay, and Satish [1] recently proposed a garbling scheme that garbles AND gates requiring communication costs of only $4/3\kappa + O(1)$ bits, thus improving upon the previous state-of-the-art construction. Their core idea is to extend the slicing-and-dicing technique by Rosulek and Roy. Precisely, they suggested to garble 3-input gates of the form $g(u, v, w) := u(v+w)$ (where the function is defined over the binary field $\mathbb{F}_2$) instead of directly garbling AND gates. They also suggested to slice the input labels into 3 pieces, whereas the RR21 construction uses 2-sliced input labels. By setting $w = 0$ in $g(u, v, w)$, one can use their garbling scheme to garble AND gates.

Our Contributions. However, in this paper, we show that their garbling scheme leaks private information on inputs, thereby jeopardizing the security guarantee that should be satisfied in the garbling scheme. Precisely, we prove that the evaluator can compute the bits α and $\beta + \gamma$, where α, β, and γ are private permute bits of the input labels A, B, and C, respectively.[1] The permute bits are used to mask the private inputs u, v, and w of the gate g, thus it should not be revealed to the evaluator to satisfy the privacy property of the garbling scheme.

Previous Works. Before describing our techniques, we briefly review prior approaches. Since its introduction by Yao [16], the core idea behind garbled circuits has centered on *encoding the truth table of a function.* For a function $g : \{0,1\}^n \to \{0,1\}$, the truth table contains $N = 2^n$ rows. Each i-th input is assigned an input label, represented as a κ-bit string, depending on the input's truth value. Likewise, the output is assigned either an output label. The garbling of g is performed by encrypting the output label corresponding to the value $v = g(u_1, \ldots, u_n)$, using the n input labels corresponding to u_i's as encryption keys. This naive approach results in a garbled function of size $N\kappa$ bits.

To fix idea, let us focus on garbling 2-input gates where each input wire has labels (A_0, A_1) and (B_0, B_1). From now on we assume the point-and-permute technique [4] is applied with the free-XOR setting. Precisely, let A_α and B_β be the labels corresponding to the logical value 0 on the respective input wire, where the masking bits α and β (a.k.a. the permute bits) are secretly known by the garbler. Equivalently, $A_{u+\alpha}$ and $B_{v+\beta}$ correspond to the values u and v respectively, where the addition of the subscripts is over $\mathbb{F}_2$. In the point-and-permute technique, if the evaluator holds one of $\{A_0, A_1\}$ and $\{B_0, B_1\}$, say A_x and B_y, then she would know their subscripts x and y (a.k.a. color bits and they are typically given as the least significant bit of labels). Here, we see that only the masked bits $x = u + \alpha$ and $y = v + \beta$ are revealed to the evaluator and the

[1] Each of the input labels is a κ-bit string that is assigned to each of the inputs of the gate depending on their logical values. The input labels A, B, and C correspond to the input u, v, and w, respectively.

truth values u and v are still hidden as the random mask α and β are only known by the garbler.

When garbling a Boolean circuit, a common technique is to decompose the circuit into a series of AND and XOR gates and apply a gate-by-gate garbling method. With the free-XOR technique [12], all wire labels share a global offset Δ, meaning $A_0 + A_1 = B_0 + B_1 = \Delta$, allowing XOR gates to be garbled for free. So, recent optimizations have focused on reducing the size of the ciphertexts required for garbling AND gates. Yao's original circuit requires 4κ bits per AND gate, but this was later reduced to 2κ bits using the half-gate garbling technique [17]. The current state-of-the-art achieves a size of $1.5\kappa + O(1)$ bits, as shown by Rosulek and Roy [15].

A key observation in [15] is that the encoded truth table can be viewed as a system of linear equations. To illustrate, consider Yao's garbled circuit under the free-XOR setting. The encoded truth table for an AND gate, consisting of four rows, is represented as follows:

$$\begin{aligned} C &= G_{0,0} + H(A_0, B_0) \\ C &= G_{0,1} + H(A_0, B_1) \\ C &= G_{1,0} + H(A_1, B_0) \\ C + \Delta &= G_{1,1} + H(A_1, B_1), \end{aligned}$$

where $G_{i,j}$ are the four ciphertexts corresponding to combinations of input labels (A_i, B_j), C is the output label corresponding to the truth value 0, and we assume the permute bits $(\alpha, \beta) = (1, 1)$. This can be rearranged into a system of four linear equations:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C \\ G_{0,0} \\ G_{0,1} \\ G_{1,0} \\ G_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H(A_0, B_0) \\ H(A_0, B_1) \\ H(A_1, B_0) \\ H(A_1, B_1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \Delta.$$

Here, given the input labels and the global offset, the garbler solves for the variables on the left-hand side. Notably, the matrix on the right-hand side has rank 4, and the number of ciphertexts required is determined by this rank. While the example suggests that four ciphertexts are necessary, one of the ciphertexts can be set as a zero string using a row-reduction technique [4]. This shows that 4 degrees of freedom are used to set the output label and three ciphertexts on the left-hand side.

In the half-gate garbling technique, random oracle queries take the form of $H(A_i)$ and $H(B_j)$ instead of $H(A_i, B_j)$, which results in a lower-rank garbling equation and, therefore, fewer ciphertexts.

In summary, constructing an improved garbled circuit involves finding appropriate linear equations of lower rank using the input labels and the global offset. To further reduce ciphertext size, Rosulek and Roy [15] introduced new types of random oracle queries, such as $H(A_i + B_j)$, alongside the existing $H(A_i)$ and $H(B_j)$ queries. Additionally, they split the output labels into two parts and considered eight (i.e., 2×4) linear equations, where each row represents the left or right half of the output labels based on the four possible input label combinations. This approach allowed to obtain smaller ciphertexts, but required an exhaustive search to finalize the garbling equation.

The recent approach by [1] builds on this idea of a linear algebraic representation of garbling equations. They extended the techniques from [15] to 3-input gates and demonstrated how to garble a gate of the form $g(u, v, w) := u(v + w)$. While they explained how to apply the linear algebraic approach to this gate, an explicit formula has not yet been provided.

More details can be found in the subsequent sections.

Our Techniques. Previous works, such as [15,1], viewed garbling schemes as systems of linear equations. The core idea of this paper is based on a novel observation by [2], which reformulates these linear systems in garbling schemes as algebraic equations. In this approach, the color bits x and y are treated as variables in $\mathbb{F}_2$, and the garbling equation is expressed as polynomials over $\mathbb{F}_{2^\kappa}$, with x and y as variables.

Let us revisit Yao's circuit as a concrete example. Using the algebraic framework of [2], the system of linear equations can be reformulated as follows:

$$\begin{aligned} &C + (x+1)(y+1)G_{0,0} + (x+1)yG_{0,1} + x(y+1)G_{1,0} + xyG_{1,1} \\ &= (x+1)(y+1)H_{0,0} + (x+1)yH_{0,1} + x(y+1)H_{1,0} + xyH_{1,1} + (x+\alpha)(y+\beta)\Delta, \end{aligned}$$

where $H_{i,j} := H(A_i, B_j)$. This shows that the garbling equation in Yao's circuit can be represented using quadratic polynomials, with coefficients on the right-hand side determined by the input labels and the global offset.

More generally, previous garbling schemes can be expressed in the following form:

$$C + g(x+\alpha, y+\beta)\Delta = \mathcal{F}(x, y),$$

where $\mathcal{F}$ is a function that takes the color bits x and y as inputs and g is a target function to be garbled.

This simple reformulation allows garbling schemes to be represented in a more compact and elegant form, making it easier to identify a suitable function $\mathcal{F}$ that yields a correct garbling scheme. A major challenge in the constructions of [15] and [1] was to find the *control matrices* required to complete the function $\mathcal{F}$ for a valid garbling scheme.

While the work of [2] primarily focuses on garbling 2-input AND gates, we observe that this algebraic perspective can be extended to garbling gadgets with an arbitrary number of inputs and degrees. This representation allows us to derive an *explicit formula* for the garbling equation in the construction by Ashur et al. Once this formula is established, it becomes evident that this construction leaks the permute bits.

As a side result, we derive an impossibility result, showing that garbling gates of degree greater than 2 is not feasible if the only allowed queries to the oracle are restricted to linear functions of the input labels. This result aligns with Theorem 3 from Ashur et al. [1]. For gates of degree 2, we provide sufficient conditions under which our attack can succeed. This also offers guidance on *how not to* design garbling schemes to reduce communication costs.

Limitations. We remark that our analysis primarily considers garbling schemes within the linear garbling model, as defined by [17]. In essence, this model permits only linear operations and queries to the random oracle. While the state-of-the-art construction [15] slightly deviates from the original definition of the linear garbling model, it still adheres to the principle of using only linear operations and random oracle queries, with the exception that wire labels are split into two parts, each obtained through linear operations and random oracle queries. In our analysis, we extend this by allowing wire labels to be split into multiple parts, though they are still derived exclusively through linear operations and random oracle queries. Additionally, we assume, as in previous approaches [15,1], that inputs to the random oracle are formed as linear combinations of the input labels. We exclude non-linear queries, as they appear to increase the number of ciphertexts due to the higher degree in the garbling equation.

It is also important to note that our focus does not extend to constructions that operate outside the linear garbling model. For example, we do not consider constructions based on one-hot garbling [8,9] or those relying on DCR assumptions [3].

Concurrent Works. Concurrently to our work, Fan, Lu, and Zhou [6] also described an attack on the scheme by Ashur, Hazay, and Satish [1]. In their work, they described how the permute bit α can be revealed using the linear-algebraic representation of garbling schemes when the evaluator's color bits are $(0, 0, 0)$. On the other hand, with our algebraic approach, we can provide more general results: we show that the evaluator can reveal not only the permute bit α, but also the value $\beta + \gamma$, regardless of the evaluator's given color bits.

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 [5]. In particular, as in [17], 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 [5].

3 Algebraic Understanding of Garbling Schemes

In [15], garbling schemes were interpreted as systems of linear equations. Building on this idea, they were able to devise an efficient garbling scheme that improved upon previous state-of-the-art constructions. In this section, we review the reformulation introduced by [2], which offers an algebraic perspective on existing garbling schemes.

3.1 Review on Existing Schemes

For now, we focus on a gate g with input wires a, b and output wire c. If g is the AND gate, we have $c = g(a, b) = a \cdot b$ for $a, b \in \mathbb{F}_2$. From now on we assume the point-and-permute techniques [4].

Notations.

- *(Permute bits)* The values α and β are secret permute bits that are only known by the garbler.
- *(Input labels)* $A_\alpha, B_\beta \in \{0,1\}^\kappa$ are wire labels corresponding to the **false** value on input wires a and b, respectively.
- *(Output label)* $C \in \{0,1\}^\kappa$ represents the output wire label corresponding to the **false** value on the output wire c.
- *(Color bits)* When A_x and B_y are held by the evaluator, we assume that the subscripts, x and y, referred to as color bits, are known by the evaluator.
- *(Free-XOR)* For each wire label W, it holds $W_0 + W_1 = \varDelta$, with the addition being performed over $\mathbb{F}_2$. Here $\varDelta \in \{0,1\}^\kappa$ is a global offset.
- *(Sliced labels)* Given an integer s, we represent the wire label $W = W^1 \| \cdots \| W^s$ as $\vec{W} = (W^1, \ldots, W^s)$, where each W^i is κ/s-bits. If $s = 1$, we simply write W instead of $\vec{W}$.

With the above notations, the wire labels A_x and B_y correspond to the logical values $u := x + \alpha$ and $v := y + \beta$, where the addition is computed over $\mathbb{F}_2$.

As a matter of mathematical conventions, we abstract strings in $\{0,1\}^\kappa$ as fields elements in $\mathbb{F}_{2^\kappa}$. However, the field structures that we are only interested in are additions and multiplications by $\mathbb{F}_2$ (no multiplications by full field elements are required).

Yao's Garbled Circuits. We describe the classical Yao's garbled circuits from the algebraic perspective. In Yao's circuit, the garbler generates the ciphertexts $G_{x,y}$ for each $(x, y) \in \mathbb{F}_2 \times \mathbb{F}_2$ in a way that the following equation holds:

$$C + (x + \alpha)(y + \beta)\mathit{\Delta} = G_{x,y} + H(A_x, B_y). \tag{1}$$

In other words, decryption of $G_{x,y}$ under the key $H(A_x, B_y)$ yields to the value $C + \mathit{\Delta}$, the output wire label corresponding to the logical value 1, if and only if $(x + \alpha)(y + \beta) = 1$.

Let us consider $G_{x,y}$ and $H_{x,y} := H(A_x, B_y)$ as functions in variables x, y with their values in $\mathbb{F}_{2^\kappa}$. Particularly, we only consider x, y that take values in $\mathbb{F}_2$, then we may write the functions as formal summations using group algebra:

$$\begin{aligned} \mathbb{T}_\kappa &:= \mathbb{F}_{2^\kappa}\left[\mathbb{F}_2[x,y]/(x^2+x, y^2+y)\right] \\ &= \left\{ \textstyle\sum_f a_f \cdot f \mid a_f \in \mathbb{F}_{2^\kappa}, f \in \mathbb{F}_2[x,y]/(x^2+x, y^2+y) \right\}. \end{aligned}$$

For instance, using Lagrange polynomials, we can write:

$$\begin{aligned} G_{x,y} &:= G_{0,0}(x+1)(y+1) + G_{0,1}(x+1)y + G_{1,0}x(y+1) + G_{1,1}xy \in \mathbb{T}_\kappa \\ H_{x,y} &:= H_{0,0}(x+1)(y+1) + H_{0,1}(x+1)y + H_{1,0}x(y+1) + H_{1,1}xy \in \mathbb{T}_\kappa. \end{aligned} \tag{2}$$

To generate the ciphertexts, the garbler chooses $\vec{G} := (G_{0,0}, G_{0,1}, G_{1,0}, G_{1,1})$ so that Equation (1) should hold for any choices of $(x, y) \in \mathbb{F}_2 \times \mathbb{F}_2$. By comparing the coefficients of $1, x, y$ and xy in the both sides, we obtain the following system of linear equations:

$$\begin{aligned} C + \alpha\beta\mathit{\Delta} &= G_{0,0} + H_{0,0} \\ \beta\mathit{\Delta} &= G_{0,0} + G_{1,0} + H_{0,0} + H_{1,0} \\ \alpha\mathit{\Delta} &= G_{0,0} + G_{0,1} + H_{0,0} + H_{0,1} \\ \mathit{\Delta} &= \textstyle\sum_{i,j \in \mathbb{F}_2} (G_{i,j} + H_{i,j}). \end{aligned} \tag{3}$$

The garbler produces the desired ciphertexts by solving this system of equations with respect to $\vec{G}$. One might easily check that it is equivalent to the system of linear equations described in [15].

For readers, it seems to be just a matter of wordplay. Nevertheless, it provides an intriguing intuition into the understanding of garbled schemes as we shall see below.

Row Reduction Technique. In Equation (3), all equations have the term $G_{0,0}$ in common. Therefore, simply canceling out the term will not affect on solving the system of linear equations. It allows us to set $G_{0,0} = 0 \in \mathbb{F}_{2^\kappa}$. It reduces the size of ciphertexts from 4κ to 3κ.

Half-Gate Garbling. Zahur, Rosulek and Evans [17] showed that the size of ciphertexts further reduces to 2κ. It is called the half-gate garbling technique. Their construction has two main differences: First, the ciphertexts $G_{x,y}$ are generated using only two ciphertexts G_0 and G_1 through the formula $G_{x,y} := xG_0 + yG_1$, rather than four independent ciphertexts. Second, the output labels are encrypted using the key $H(A_x) + H(B_y)$ instead of $H(A_x, B_y)$.

We provide a more general explanation of this technique from the algebraic point of view. In the technique, the garbling scheme can be represented as the following equation:

$$C+(x+\alpha)(y+\beta)\Delta = xG_0+yG_1+H(A_x)+H(B_y)+R_A(x,y)\cdot A_x+R_B(x,y)\cdot B_y, \quad (4)$$

where R_A, R_B are linear polynomials in $\mathbb{F}_2[x,y]$ to be determined later. Similar to Equation (2), we write the following terms as polynomials in $\mathbb{T}_\kappa$:

$$\begin{aligned} H(A_x) &= (x+1)H(A_0) + xH(A_1) \\ H(B_y) &= (y+1)H(B_0) + xH(B_1) \\ A_x &= A_0 + x\Delta \\ B_y &= B_0 + y\Delta. \end{aligned} \quad (5)$$

Let us set $R_A = y$ and $R_B = 0$ which is equivalent to the setting of [17]. As before, one substitutes Equation (5) into Equation (4) and compares the coefficients. We have

$$\begin{aligned} C + \alpha\beta\Delta &= H(A_0) + H(B_0) \\ \beta\Delta &= G_0 + H(A_0) + H(A_1) \\ \alpha\Delta &= G_1 + H(B_0) + H(B_1). \end{aligned} \quad (6)$$

The garbler determines the output label C and the ciphertexts (G_0, G_1) using the above equation.

In general, setting $R_A = a_0 + a_1x + a_2y$ and $R_B = b_0 + b_1x + b_2y$ such that $a_2 + b_1 = 1$ leads us to a valid garbled circuit. This can be seen as follows: The expression $R_A(A_0 + x\Delta) + R_B(B_0 + y\Delta)$ contains the term $xy\Delta$ if and only if $a_2 + b_1 = 1$. Since the quadratic term $xy\Delta$ gets cancelled out in Equation (4), the values of C and G_0, G_1 can be found by comparing the coefficients of $1, x$ and y.

From our algebraic viewpoint, it is interesting to observe that their modifications involving the use of $xG_0 + yG_1$ and $H(A_x) + H(B_y)$ instead of $G_{x,y}$ and $H(A_x, B_y)$ enabled the representation of the garbling equation without any quadratic terms. Consequently, this reduced the number of free variables that have to be determined.

RR21's Garbling Scheme. Rosulek and Roy [15] proposed a garbling scheme, dubbed as RR21 from now on, that further reduces the size of the ciphertexts from 2κ to $1.5\kappa + O(1)$. One of their primary ideas involves splitting the output wire label into two parts, say $C = (C^L \| C^R)$, where each half is of $\kappa/2$ bits. Then they discuss how to derive each half using three well-structured $\kappa/2$-bits ciphertexts G_0, G_1 and G_2, along with input labels. It should be noted that a straightforward application of the half-gate strategy would not yield any enhancements: The size of the garbled gates for each half would be of $2 \cdot (\kappa/2)$ bits.

To obtain a further reduction on the garbled gate size, the RR21 construction implicitly ensures that two of the four $\kappa/2$-bits ciphertexts will be identical. For instance, suppose that (G_0^L, G_1^L) and (G_0^R, G_1^R) are ciphertexts for the left and right halves of C, respectively. They enforce $G_1^L = G_0^R$ and demonstrate how to construct garbling schemes satisfying this condition. To accomplish this challenging task, they propose employing extra oracle queries on $A_x + B_y$ alongside A_x and B_y. Then the idea is to use the following combination of oracle queries to mask the each half of the output labels:

$$\begin{aligned} C^L + (x+\alpha)(y+\beta)\Delta^L &= xG_0^L + yG_1^L + H(A_x) + H(A_x + B_y) + \cdots, \\ C^R + (x+\alpha)(y+\beta)\Delta^R &= xG_0^R + yG_1^R + H(B_y) + H(A_x + B_y) + \cdots. \end{aligned} \tag{7}$$

Keeping in mind the above, one may interpret the RR21 construction from the algebraic viewpoint. Again, we can write $H(A_x + B_y)$ as follows:[2]

$$H(A_x + B_y) = (x+y+1)H(A_0 + B_0) + (x+y)H(A_0 + B_1).$$

Then we have

$$\begin{bmatrix} H(A_x) + H(A_x + B_y) \\ H(B_y) + H(A_x + B_y) \end{bmatrix} = \mathbf{M} \cdot \vec{H},$$

where

$$\mathbf{M} := \begin{bmatrix} x+1 & x & 0 & 0 & x+y+1 & x+y \\ 0 & 0 & y+1 & y & x+y+1 & x+y \end{bmatrix}$$

and $\vec{H} := (H(A_0), H(A_1), H(B_0), H(B_1), H(A_0 + B_0), H(A_0 + B_1))^\top$.

Interestingly, we observe that the y-coefficient of $H(A_x) + H(A_x + B_y)$ and the x-coefficient of $H(B_y) + H(A_x + B_y)$ are identical to each other. Let us enforce $G_1^L = G_0^R$ as desired. By properly rearranging and rewriting Equation (7), the RR21 construction can be restated as the following equation:

$$\mathbf{V} \begin{bmatrix} \vec{C} \\ \vec{G} \end{bmatrix} = \mathbf{M}\vec{H} + \mathbf{R}_A(\vec{A}_0 + x\vec{\Delta}) + \mathbf{R}_B(\vec{B}_0 + y\vec{\Delta}) + (x+\alpha)(y+\beta)\vec{\Delta}, \tag{8}$$

where

$$\mathbf{V} := \begin{bmatrix} 1 & 0 & x & 0 & x+y \\ 0 & 1 & 0 & y & x+y \end{bmatrix}, \vec{C} := \begin{bmatrix} C^L \\ C^R \end{bmatrix}, \vec{A}_0 := \begin{bmatrix} A_0^L \\ A_0^R \end{bmatrix}, \vec{B}_0 := \begin{bmatrix} B_0^L \\ B_0^R \end{bmatrix}, \vec{\Delta} := \begin{bmatrix} \Delta^L \\ \Delta^R \end{bmatrix},$$

and $\vec{G} := (G_0, G_1, G_2)^\top = (G_1^L + G_0^L, G_1^L + G_1^R, G_1^L)^\top$. Here, $\mathbf{R}_A$ and $\mathbf{R}_B$ are 2×2 matrices over $\mathbb{F}_2[x, y]$ to be determined later.

Observe that the y-coefficient of the upper and the x-coefficient of the lower in $\mathbf{V} \begin{bmatrix} \vec{C} \\ \vec{G} \end{bmatrix}$ coincides with each other. It is exactly equivalent to saying that $\mathbf{M}$ *and*

[2] With the free-XOR constraint, recall that $H(A_0 + B_0) = H(A_1 + B_1)$ and $H(A_0 + B_1) = H(A_1 + B_0)$.

$\mathbf{V}$ *have the same column space over* $\mathbb{F}_2$. Indeed, $\mathbf{V}$ is a column-reduced matrix of $\mathbf{M}$ where the operation is carried over $\mathbb{F}_2$.

Once $\mathbf{R}_A$ and $\mathbf{R}_B$ are determined (as we shall describe soon how to determine them), the garbler generates the output label $\vec{C}$ and the ciphertexts $\vec{G}$ analogously to prior constructions, ensuring that Equation (8) holds for all $x, y \in \mathbb{F}_2$.

Choosing Control Matrices. We now proceed to explain the procedure for determining the matrices $\mathbf{R}_A$ and $\mathbf{R}_B$. According to the linear-algebraic representation in [15], this step is equivalent to find out the control matrix. Although, according to their linear-algebraic representation, they managed to identify the control matrix solely through an exhaustive computer search, we can provide explicit formulas for $\mathbf{R}_A$ and $\mathbf{R}_B$ due to the algebraic perspective.

First of all, we note that $\mathbf{M}$ and $\mathbf{V}$ share the same column space. Therefore, it suffices to guarantee that the remaining term of Equation (8) also belongs to this common space, ensuring the existence of $\vec{C}$ and $\vec{G}$ satisfying Equation (8). More precisely, it is equivalent to the following:

1. The y-coefficient of the top and the x-coefficient of the bottom in $\mathbf{R}_A(\vec{A}_0 + x\vec{\Delta}) + \mathbf{R}_B(\vec{B}_0 + y\vec{\Delta}) + (x+\alpha)(y+\beta)\vec{\Delta}$ coincide with each other;
2. The xy-term in $\mathbf{R}_A(\vec{A}_0 + x\vec{\Delta}) + \mathbf{R}_B(\vec{B}_0 + y\vec{\Delta}) + (x+\alpha)(y+\beta)\vec{\Delta}$ vanishes.

To find $\mathbf{R}_A$ and $\mathbf{R}_B$ satisfying the aforementioned conditions, we write $\mathbf{R}_X = \mathbf{R}_{X,0} + \mathbf{R}_{X,1}x + \mathbf{R}_{X,2}y$ for $X \in \{A, B\}$, where each $\mathbf{R}_{X,i}$ is a 2×2 binary matrix.[3] Substituting this to Equation (8) and imposing that $x^2 = x$ and $y^2 = y$ yield

$$\begin{aligned}\mathbf{V}\begin{bmatrix}\vec{C}\\ \vec{G}\end{bmatrix} &= \mathbf{M}\vec{H} + \Big(\mathbf{R}_{A,1}\vec{A}_0 + \mathbf{R}_{B,1}\vec{B}_0 + \big(\mathbf{R}_{A,0} + \mathbf{R}_{A,1} + \beta\mathbf{I}\big)\vec{\Delta}\Big)\,x \\ &+ \Big(\mathbf{R}_{A,2}\vec{A}_0 + \mathbf{R}_{B,2}\vec{B}_0 + \big(\mathbf{R}_{B,0} + \mathbf{R}_{B,2} + \alpha\mathbf{I}\big)\vec{\Delta}\Big)\,y \\ &+ \Big(\big(\mathbf{R}_{A,2} + \mathbf{R}_{B,1} + \mathbf{I}\big)\vec{\Delta}\Big)\,xy + (constant),\end{aligned} \tag{9}$$

where $\mathbf{I}$ is the 2-dimensional identity matrix.

As the aforementioned conditions should satisfy for arbitrary choices of $\vec{A}_0, \vec{B}_0$, and $\vec{\Delta}$, the conditions translate to the following system of equations:

$$\begin{aligned}\mathbf{R}_{A,2} + \mathbf{R}_{B,1} + \mathbf{I}_s &= 0 \\ \begin{bmatrix}0 & 1\end{bmatrix}\mathbf{R}_{A,1} &= \begin{bmatrix}1 & 0\end{bmatrix}\mathbf{R}_{A,2} \\ \begin{bmatrix}0 & 1\end{bmatrix}\mathbf{R}_{B,1} &= \begin{bmatrix}1 & 0\end{bmatrix}\mathbf{R}_{B,2} \\ \begin{bmatrix}0 & 1\end{bmatrix}\big(\mathbf{R}_{A,0} + \mathbf{R}_{A,1} + \beta\mathbf{I}\big) &= \begin{bmatrix}1 & 0\end{bmatrix}\big(\mathbf{R}_{B,0} + \mathbf{R}_{B,2} + \alpha\mathbf{I}\big)\end{aligned} \tag{10}$$

[3] We observe that it is sufficient to consider the case where $\mathbf{R}_X$ is linear rather than of arbitrary degree. This is because when $\vec{H}$ consists solely of linear queries, $\mathbf{M}$ is composed only of linear polynomials. For $\mathbf{R}_A(\vec{A}_0 + x\vec{\Delta})$ to lie within the same column space as $\mathbf{M}$, we deduce that $\mathbf{R}_A\vec{A}_0 \in \mathrm{span}(\mathbf{M})$ for any $\vec{A}_0$. Therefore, $\mathbf{R}_A$ must also be linear.

Solving the above equations provides us the following formulas for $\mathbf{R}_A$ and $\mathbf{R}_B$:

$$\begin{aligned}
\mathbf{R}_{A,1} &= \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, & \mathbf{R}_{A,2} &= \begin{bmatrix} a_3 & a_4 \\ b_3 & b_4 \end{bmatrix}, & \mathbf{R}_{A,0} &= \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix} \\
\mathbf{R}_{B,1} &= \begin{bmatrix} a_3+1 & a_4 \\ b_3 & b_4+1 \end{bmatrix}, & \mathbf{R}_{B,2} &= \begin{bmatrix} b_3 & b_4+1 \\ e_3 & e_4 \end{bmatrix}, & \mathbf{R}_{B,0} &= \begin{bmatrix} f_1 & f_2 \\ f_3 & f_4 \end{bmatrix},
\end{aligned} \tag{11}$$

where $f_1 = a_3+b_3+c_3+\alpha$ and $f_2 = a_4+b_4+c_4+\beta+1$ and all the other unspecified entries are arbitrary binary elements. One might observe that $(\mathbf{R}_A, \mathbf{R}_B)$ is a 14-dimensional space.

To garble a gate, the garbler must select a pair of matrices $(\mathbf{R}_A, \mathbf{R}_B)$ from the set of 2^{14} possible choices. The garbler then needs to share this selection with the evaluator so she can utilize it in order to decrypt the encrypted gate. However, unlike the half-gate scheme, $(\mathbf{R}_A, \mathbf{R}_B)$ are dependent on the choice of the secret permute bits α and β. As a result, providing the matrices $(\mathbf{R}_A, \mathbf{R}_B)$ in clear to the evaluator would violate the privacy property of the garbling scheme. Rosulek and Roy addressed this challenge by incorporating the concept of *dicing*, initially proposed by Kempka, Kikuchi, and Suzuki in [10]. In a nutshell, the method involves encrypting the matrices $(\mathbf{R}_A, \mathbf{R}_B)$ and sending the resulting ciphertexts to the evaluator, ensuring that she can only obtain a decryption the ciphertexts on her active input labels. Precisely, if A_i and B_j are the evaluator's active input labels, then she will solely be capable of obtaining the value of $(\mathbf{R}_A(i,j), \mathbf{R}_B(i,j))$, rather than having access to the entire information on $(\mathbf{R}_A, \mathbf{R}_B)$. For further clarification, we have provided additional details regarding the dicing technique from the algebraic perspective in Appendix A.

4 Analysis on Sliced Garbling

4.1 A Genearlized Framework for Garbling Schemes

In this section, we provide a generalized framework for constructing garbled gates based on the observations made in Section 3. For simplicity of discussion, we primarily concentrate on a gate g with three fan-ins and a single fan-out throughout this section. Nonetheless, it is evident that similar arguments are still valid for any gate g with a higher fan-in.

To begin with, we first re-establish some essential notation. Let g be a gate with input wires a, b, c and output wire d.

Notations.

– *(Permute bits)* The values $\alpha, \beta, \gamma \in \{0,1\}$ are secret permute bits that are only known by the garbler.
– *(Input labels)* $A_\alpha, B_\beta, C_\gamma \in \{0,1\}^\kappa$ are wire labels corresponding to the **false** value on input wires a, b and c, respectively.
– *(Output label)* $D \in \{0,1\}^\kappa$ represents the output wire label corresponding to the **false** value on the output wire d.

– *(Color bits)* When A_x, B_y and C_z are held by the evaluator, we assume that the subscripts, x, y and z, referred to as color bits, are known by the evaluator.
– *(Free-XOR)* The value $\Delta \in \{0,1\}^\kappa$ is a global offset secretly chosen by the garbler. Then, $W_0 + W_1 = \Delta$ for $W \in \{A, B, C\}$. Also, $D + \Delta$ represents the output label corresponding to the **true** on the wire d.
– *(Sliced labels)* Given an integer s, we represent the wire label $W = W^1 \| \cdots \| W^s$ as $\vec{W} = (W^1, \ldots, W^s)$, where each W^i is κ/s-bits. If $s = 1$, we simply write W instead of $\vec{W}$.

With the above notations, the wire labels A_x, B_y and C_z correspond to the logical values $u := x + \alpha, v := y + \beta$ and $w := z + \gamma$, where the addition is carried out over $\mathbb{F}_2$.

In the following, we assume that only linear operations are allowed, apart from querying random oracles on input labels, during the construction of garbled circuits. Based on the prescribed observation, we specifically consider a garbling scheme that can be represented as an equation of the following form:

$$\vec{D} + g(x + \alpha, y + \beta, z + \gamma)\vec{\Delta} = \mathbf{W}\vec{G} + \mathbf{M}\vec{H} + \mathbf{R}_A\vec{A}_x + \mathbf{R}_B\vec{B}_y + \mathbf{R}_C\vec{C}_z. \quad (12)$$

Providing a concrete scheme involves the steps of specifying the matrices $\mathbf{W}$, $\mathbf{M}$, $\mathbf{R}_A$, $\mathbf{R}_B$ and $\mathbf{R}_C$ (which are over $\mathbb{F}_2[x, y, z]$) and $\vec{H}$ (which are over $\mathbb{F}_{2^{\kappa/s}}$), whose detailed descriptions will be presented subsequently.

Once they are established, to garble a 3-input gate g, the garbler generates the output label $\vec{D}$ and the ciphertexts $\vec{G}$ according to Equation (12). Upon receiving the ciphertexts $\vec{G}$, the evaluator computes the right-hand side of Equation (12) using her *active* input labels A_i, B_j, and C_k for some $(i, j, k) \in \mathbb{F}_2^3$. This allows the evaluator only revealing the output label corresponding to $g(i + \alpha, j + \beta, k + \gamma)$.[4]

Now, we proceed to explain on the idea of establishing Equation (12). It is analogous to the description we have seen from Section 3. It can be summarized as follows:

1. The vector $\vec{H}$ defined over $\mathbb{F}_{2^{\kappa/s}}$ consists of all possible responses of oracle queries that can be made during the construction of garbled circuits.
2. The matrix $\mathbf{M}$ is a matrix over $\mathbb{F}_2[x, y, z]$ with s rows. Each row of $\mathbf{M}$ is determined by which combinations of oracle queries are to be used for the corresponding slices.
3. The matrix $\mathbf{V} := [\mathbf{I}|\mathbf{W}]$ is chosen so that it has the same column space (over $\mathbb{F}_2$) with the matrix $\mathbf{M}$. Here, the matrix $\mathbf{I}$ is the s-dimensional identity matrix. Note that we implicitly assumed that the column-reduced matrix of $\mathbf{M}$ contains the identity matrix.
4. Each of the matrices $\mathbf{R}_X$ for $X \in \{A, B, C\}$ is chosen so that the vector $\mathbf{R}_A\vec{A}_x + \mathbf{R}_B\vec{B}_y + \mathbf{R}_C\vec{C}_z + g(x + \alpha, y + \beta, z + \gamma)\vec{\Delta}$ belongs to the same space spanned by the columns of $\mathbf{V}$.

[4] To simplify discussion, for now, let us presume that the evaluator obtains the values of $\mathbf{R}_A$, $\mathbf{R}_B$ and $\mathbf{R}_C$ precisely at (i, j, k) in a certain way. In other words, we implicitly assume the dicing technique is applied.

From the preceding discussion, we observe that primary concerns in garbling constructions revolve around (1) determining the matrix $\mathbf{M}$ and the vector $\vec{H}$, and (2) deriving the matrices $\mathbf{R}_X$ for each $X \in \{A, B, C\}$. Following the previous works, we refer the matrix $\mathbf{R}_X$ to the *control matrix.*

4.2 Garbled Circuits for 3-Input Gates

Recently, Ashur, Hazay and Satish [1] proposed a scheme that garbles a 3-input gate. They specifically claimed that a circuit of the form $g_{tri}(u, v, w) := u(v + w)$ can be garbled at a cost of $4/3\kappa + O(1)$ bits. As its corollary, they insisted that garbling an AND gate requires the same cost as garbling g_{tri} by fixing $w = 0$. This claim, if correct, would provide an asymptotic improvement compared to the state-of-the-art requiring $3/2\kappa + O(1)$ bits.

However, in this paper, we demonstrate that their proposed construction will leak permute bits, thereby jeopardizing the security guarantees offered by garbling schemes. Prior to describe our main results, we begin by reviewing their construction from our perspective provided in Section 4.1.

4.2.1 Review on Tri-gate Garbling Scheme

Choice of the matrix $\mathbf{M}$ *and the vector* $\vec{H}$*.* In their work [1], they suggested splitting wire labels into three slices, i.e. $s = 3$, to garble the gate g_{tri}. To encrypt each slices of the output label $\vec{D} = (D^1, D^2, D^3)$, they chose the following linear combinations of oracle queries:

$$\begin{aligned}
D^1 + g_{tri}(u, v, w)\Delta^1 &= H(A_x) + H(B_y) + H(A_x + B_y + C_z) + \cdots \\
D^2 + g_{tri}(u, v, w)\Delta^2 &= H(B_y) + H(C_z) + H(A_x + B_y + C_z) + \cdots \\
D^3 + g_{tri}(u, v, w)\Delta^3 &= H(A_x) + H(C_z) + H(A_x + B_y + C_z) + \cdots ,
\end{aligned}$$

where $u := x + \alpha, v := y + \beta$ and $w := z + \gamma$.

Analogously to the previous constructions, we represent them into polynomials as follows. For instance, under the free-XOR setting, we may write

$$H(A_x + B_y + C_z) = (x + y + z + 1)H(A_0 + B_0 + C_0) + (x + y + z)H(A_0 + B_0 + C_1).$$

Let us define

$$\vec{H} := \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \\ H(C_0) \\ H(C_1) \\ H(A_0 + B_0 + C_0) \\ H(A_0 + B_0 + C_1) \end{bmatrix}.$$

Then this determines the matrix $\mathbf{M}$ in Equation (12) as follows:

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

It can be easily verified that the following matrix $\mathbf{V}$ has the same column space as $\mathbf{M}$, where the span is carried out over $\mathbb{F}_{2^{\kappa/3}}$:

$$\mathbf{V} = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & x & y & 0 & x+y+z \\ 0 & 1 & 0 & 0 & y & z & x+y+z \\ 0 & 0 & 1 & x & 0 & z & x+y+z \end{array}\right] := [\mathbf{I} \mid \mathbf{W}]\,. \tag{13}$$

Here, $\mathbf{I}$ is the 3-dimensional identity matrix and $\mathbf{W}$ is the right (3×4)-matrix.

Choosing the control matrix $\mathbf{R}_X$. Having determined the matrix $\mathbf{M}$, the next step is to choose the control matrix $\mathbf{R}_X$. Based on the linear-algebraic representation by Rosulek and Roy [15], Ashur, Hazay and Satish [1] demonstrated how to find the control matrices. While they elucidated the methodology for finding these matrices, explicit formulas were not provided.

In the following, we provide explicit formulas for the control matrices based on our algebraic perspective. As desired, this will show that how their proposed construction leaks the secret permute bits, even though the dicing technique is properly applied.

We begin with scrutinizing the subspace spanned by the columns of $\mathbf{V} = [\mathbf{I} \mid \mathbf{W}]$. Let us consider

$$\mathit{span}(\mathbf{V}) := \left\{ \mathbf{V} \cdot \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} \middle| \vec{D} \in \mathbb{F}^3_{2^{\kappa/3}}, \vec{G} \in \mathbb{F}^4_{2^{\kappa/3}} \right\}.$$

We represent the matrix $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{W}_1 x + \mathbf{W}_2 y + \mathbf{W}_3 z,$$

where each $\mathbf{W}_i$ is a (3×4)-binary matrix derived from $\mathbf{W}$. Note that these matrices are formed by considering the coefficients of x, y, and z in the expansion of $\mathbf{W}$.

Let us consider a cokernel matrix $\mathbf{P} = [\mathbf{P}_1 \mid \mathbf{P}_2 \mid \mathbf{P}_3]$ of a binary matrix formed by $\begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \mathbf{W}_3 \end{bmatrix}$. In other words, we have the following:

$$\mathbf{P}_1\mathbf{W}_1 + \mathbf{P}_2\mathbf{W}_2 + \mathbf{P}_3\mathbf{W}_3 = 0,$$

where each $\mathbf{P}_i$ is a (5×3) binary matrix.

Given $\vec{\nu} \in \mathit{span}(\mathbf{V})$, we represent the vector $\vec{\nu}$ as $\vec{\nu} = \vec{\nu}_0 + \vec{\nu}_1 x + \vec{\nu}_2 y + \vec{\nu}_3 z$, where $\vec{\nu}_i = \mathbf{W}_i \vec{G}$ for some $\vec{G} \in \mathbb{F}^4_{2^{\kappa/3}}$. Thus we have that

$$\mathbf{P}_1\vec{\nu}_1 + \mathbf{P}_2\vec{\nu}_2 + \mathbf{P}_3\vec{\nu}_3 = 0 \text{ for all } \vec{\nu} \in \mathit{span}(\mathbf{V}).$$

We shall use this relation to find the control matrices $\mathbf{R}_X$. We define this relation, denoted by $\pi_{\mathbf{V}}$, as:

$$\pi_{\mathbf{V}}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) := \mathbf{P}_1\mathbf{X}_1 + \mathbf{P}_2\mathbf{X}_2 + \mathbf{P}_3\mathbf{X}_3,$$

where each $\mathbf{X}_i$ refers to a binary matrix with the same number of rows as the columns in $\mathbf{P}_i$.

In summary, the span by the columns of $\mathbf{V}$ is given by

$$\mathit{span}(\mathbf{V}) = \left\{ \vec{\nu} \mid \vec{\nu} = \vec{\nu}_0 + \vec{\nu}_1 x + \vec{\nu}_2 y + \vec{\nu}_3 z \text{ and } \pi_{\mathbf{V}}(\vec{\nu}_1, \vec{\nu}_2, \vec{\nu}_3) = 0 \text{ for } \vec{\nu}_i \in \mathbb{F}^3_{2^{\kappa/3}} \right\}.$$

Recall that each $\mathbf{R}_X$ has to be chosen so that

$$\begin{aligned} \rho &:= \mathbf{R}_A \vec{A}_x + \mathbf{R}_B \vec{B}_y + \mathbf{R}_C \vec{C}_z + g_{tri}(x+\alpha, y+\beta, z+\gamma)\vec{\varDelta} \\ &= \mathbf{R}_A \vec{A}_0 + \mathbf{R}_B \vec{B}_0 + \mathbf{R}_C \vec{C}_0 \\ &\quad + \big(x\mathbf{R}_A + y\mathbf{R}_B + z\mathbf{R}_C + (x+\alpha)(y+z+\beta+\gamma)\mathbf{I}\big)\vec{\varDelta} \in \mathit{span}(\mathbf{V}), \end{aligned} \tag{14}$$

for any choices of $\vec{A}_0, \vec{B}_0, \vec{C}_0$, and $\vec{\varDelta}$. Since $\rho := \rho_0 + \rho_1 x + \rho_2 y + \rho_3 z \in \mathit{span}(\mathbf{V})$, we also have $\pi_{\mathbf{V}}(\rho_1, \rho_2, \rho_3) = 0$. This condition must be satisfied for arbitrary choices of $\vec{A}_0, \vec{B}_0, \vec{C}_0$, and $\vec{\varDelta}$. Consequently, it is sufficient to analyze the equivalent relationships by focusing on the coefficients of $\vec{A}_0, \vec{B}_0, \vec{C}_0$, and $\vec{\varDelta}$ in Equation (14) independently.

For instance, in Equation (14), the vector $\mathbf{R}_A \vec{A}_0$ should belong to $\mathit{span}(\mathbf{V})$ for any $\vec{A}_0$. Let us express $\mathbf{R}_A = \mathbf{R}_{A,0} + \mathbf{R}_{A,1}x + \mathbf{R}_{A,2}y + \mathbf{R}_{A,3}z$, where each $\mathbf{R}_{A,i}$ is a 3×3 binary matrix. Then, the requirement that $\mathbf{R}_A \vec{A}_0 \in \mathit{span}(\mathbf{V})$ for all $\vec{A}_0$ is equivalent to the following condition:

$$\pi_{\mathbf{V}}\big(\mathbf{R}_{A,1}, \mathbf{R}_{A,2}, \mathbf{R}_{A,3}\big) = 0.$$

Similar reasoning applies to the matrices $\mathbf{R}_B$ and $\mathbf{R}_C$ as well. As a result, we have

$$\pi_{\mathbf{V}}\big(\mathbf{R}_{X,1}, \mathbf{R}_{X,2}, \mathbf{R}_{X,3}\big) = 0 \text{ for each } X \in \{A, B, C\}. \tag{15}$$

As we are interested in the case that x, y and z take the values in $\mathbb{F}_2$, we impose the condition that $x^2 = x$, $y^2 = y$, and $z^2 = z$. Then, the $\vec{\varDelta}$-term in Equation (14) can be rewritten as follows:

$$\begin{aligned} \rho_\Delta &:= x\mathbf{R}_A + y\mathbf{R}_B + z\mathbf{R}_C + (x+\alpha)(y+z+\beta+\gamma)\mathbf{I} \\ &= \big(\mathbf{R}_{A,0} + \mathbf{R}_{A,1} + (\beta+\gamma)\mathbf{I}\big)x + \big(\mathbf{R}_{B,0} + \mathbf{R}_{B,2} + \alpha\mathbf{I}\big)y + \big(\mathbf{R}_{C,0} + \mathbf{R}_{C,3} + \alpha\mathbf{I}\big)z + \\ &\quad \big(\mathbf{R}_{A,2} + \mathbf{R}_{B,1} + \mathbf{I}\big)xy + \big(\mathbf{R}_{A,3} + \mathbf{R}_{C,1} + \mathbf{I}\big)xz + \big(\mathbf{R}_{B,3} + \mathbf{R}_{C,2}\big)yz. \end{aligned}$$

The requirement that $\rho_\Delta \vec{\varDelta} \in \mathit{span}(\mathbf{V})$ for all $\vec{\varDelta}$ is equivalent to the following equations:

$$\begin{gathered} \pi_{\mathbf{V}}\big(\mathbf{R}_{A,0} + \mathbf{R}_{A,1} + (\beta+\gamma)\mathbf{I}, \mathbf{R}_{B,0} + \mathbf{R}_{B,2} + \alpha\mathbf{I}, \mathbf{R}_{C,0} + \mathbf{R}_{C,3} + \alpha\mathbf{I}\big) = 0 \\ \mathbf{R}_{A,2} + \mathbf{R}_{B,1} + \mathbf{I} = 0 \\ \mathbf{R}_{A,3} + \mathbf{R}_{C,1} + \mathbf{I} = 0 \\ \mathbf{R}_{B,3} + \mathbf{R}_{C,2} = 0 \end{gathered} \tag{16}$$

We can construct binary matrices $\mathbf{R}_{X,i}$ such that they fulfill the requirements imposed by Equation (15) and (16). Then it will provide explicit formulas for the set of all possible pairs of the control matrices. Although proving how this construction exposes the permutation bits does not require comprehensive explicit formulae, we nonetheless supply more details regarding the identification of $\mathbf{R}_{X,i}$ and their resulting explicit formulae in Appendix B for the sake of thoroughness.

4.2.2 Analysis on Tri-gate Garbling

In the rest of this section, we show that the garbling scheme suggested by [1] is insecure. To be precise, we show that the evaluator can reveal the values of α and $\beta + \gamma$. Consequently, she would be aware of the logical values associated with the input labels $\vec{A}_0$ and $\vec{B}_0 + \vec{C}_0$. Furthermore, if the evaluator has prior information about either β or γ, she can deduce all logical values related to her active input labels.

As we have explicit formulae for the control matrices $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$, as shown in Equation (22), it becomes straightforward to verify the aforementioned claims. For instance, if the evaluator's active input labels are $\vec{A}_0, \vec{B}_0$ and $\vec{C}_0$, then she will have the values of $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ evaluated at $(0, 0, 0)$. In other words, the evaluator is aware of the matrix values $(\mathbf{R}_{A,0}, \mathbf{R}_{B,0}, \mathbf{R}_{C,0})$. As one might observe from Equation (22), this directly implies the desired results: By adding the second row of $\mathbf{R}_{A,0}$ to its third row, the evaluator obtains the vector $(\beta + \gamma, 0, \beta + \gamma)$. Likewise, if the evaluator adds the first row of $\mathbf{R}_{B,0}$ with its second row, she will obtain the vector $(\alpha, \alpha, 0)$.

In the following, we also provide alternative mathematical arguments for the above claims without requiring explicit formulae for the control matrices. We begin with the following simple observations:

1. Assuming that $\pi_{\mathbf{V}}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) = 0$ and $\pi_{\mathbf{V}}(\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3) = 0$, it follows that

$$\pi_{\mathbf{V}}(\mathbf{X}_1 + \mathbf{Y}_1, \mathbf{X}_2 + \mathbf{Y}_2, \mathbf{X}_3 + \mathbf{Y}_3) = 0.$$

   Each of $\mathbf{X}_i$ and $\mathbf{Y}_i$ are binary matrices with identical dimensions and an equal number of rows as the columns in $\mathbf{P}_i$.
2. For the matrix $\mathbf{P} = [\mathbf{P}_1 \mid \mathbf{P}_2 \mid \mathbf{P}_3]$, as given in Equation (21), there exists non-zero vectors $\vec{k}_1, \vec{k}_2$ and $\vec{k}_3$ such that

$$\vec{k}_i^\top \mathbf{P}_{i-1} = \vec{k}_i^\top \mathbf{P}_{i+1} = 0 \text{ and } \vec{k}_i^\top \mathbf{P}_i \neq 0$$

   for every $i = 1, 2, 3$. Here, the subscript indices are computed modulo 3. For instance, one might choose $\vec{k}_1^\top = (1, 0, 0, 0, 0)$, $\vec{k}_1^\top = (0, 1, 0, 0, 0)$, and $\vec{k}_3^\top = (0, 0, 1, 0, 0)$.

Based on the above observations, we prove the following theorem:

Theorem 1. *Let* $\mathbf{V}$ *be as Equation* (13)*. For a fixed triple* $(\alpha, \beta, \gamma) \in \mathbb{F}_2^3$*, consider a set* $\mathcal{R}$ *of three matrices* $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ *such that the vector* ρ *defined in Equation* (14) *belongs to span*$(\mathbf{V})$ *for any choices of* $\vec{A}_0, \vec{B}_0, \vec{C}_0$*, and* $\vec{\Delta}$*. Take an element* $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ *from* $\mathcal{R}$*. As before, let us write* $\mathbf{R}_X = \mathbf{R}_{X,0} + \mathbf{R}_{X,1}x + \mathbf{R}_{X,2}y + \mathbf{R}_{X,3}z$*. Given the triple of matrices* $(\mathbf{R}_A(i,j,k), \mathbf{R}_B(i,j,k), \mathbf{R}_C(i,j,k))$ *for* $i, j, k \in \{0, 1\}$*, one can compute the values of* α *and* $\beta + \gamma$*.*

Proof. First, let us recall that $\rho \in \mathit{span}(\mathbf{V})$ implies that Equation (15) and (16). We prove the assertion case by case.

Case 1. $(i,j,k)=(0,0,0)$: Recall that we are working over a field of characteristic 2. By adding the first equation in Equation (16) with $\pi_{\mathbf{V}}\big(\mathbf{R}_{A,1},\mathbf{R}_{A,2},\mathbf{R}_{A,3}\big)=0$ (see Equation (15)), we obtain the following:

$$\pi_{\mathbf{V}}\big(\mathbf{R}_{A,0}+(\beta+\gamma)\mathbf{I},\mathbf{R}_{B,0}+\mathbf{R}_{B,2}+\mathbf{R}_{A,2}+\alpha\mathbf{I},\mathbf{R}_{C,0}+\mathbf{R}_{C,3}+\mathbf{R}_{A,3}+\alpha\mathbf{I}\big)=0. \quad (17)$$

Multiplying $\vec{k}_1^\top$ to both sides of Equation (17), we obtain:

$$\vec{k}_1^\top\mathbf{P}_1\big(\mathbf{R}_{A,0}+(\beta+\gamma)\mathbf{I}\big)=0 \Longrightarrow \vec{k}_1^\top\mathbf{P}_1\mathbf{R}_{A,0}=(\beta+\gamma)\vec{k}_1^\top\mathbf{P}_1.$$

Since $\mathbf{R}_A(0,0,0)=\mathbf{R}_{A,0}$ is given, calculating $\vec{k}_1^\top\mathbf{P}_1\mathbf{R}_{A,0}$ and examining whether it equals zero or not allows one to determine the bit of $\beta+\gamma$.

Similarly, adding the first equation in Equation (16) with $\pi_{\mathbf{V}}\big(\mathbf{R}_{A,1},\mathbf{R}_{A,2},\mathbf{R}_{A,3}\big)=0$ leads us to obtain the equation of the following form:

$$\pi_{\mathbf{V}}\big(\ *,\mathbf{R}_{B,0}+\alpha\mathbf{I},*\ \big)=0.$$

Multiplying $\vec{k}_2^\top$ to the both side of the equation, we obtain:

$$\vec{k}_2^\top\mathbf{P}_2\big(\mathbf{R}_{B,0}+\alpha\mathbf{I}\big)=0 \Longrightarrow \vec{k}_2^\top\mathbf{P}_2\mathbf{R}_{B,0}=\alpha\vec{k}_1^\top\mathbf{P}_1.$$

From this relation, one can deduce the bit of α.

Case 2. $(i,j,k)=(1,0,0)$: In this case $\mathbf{R}_A(1,0,0)=\mathbf{R}_{A,0}+\mathbf{R}_{A,1}$ is given. Multiplying $\vec{k}_1^\top$ to the first equation of Equation (16) provides us

$$\vec{k}_1^\top\mathbf{P}_1\big(\mathbf{R}_{A,0}+\mathbf{R}_{A,1}+(\beta+\gamma)\mathbf{I}\big)=0.$$

Thus, one can obtain the value of $\beta+\gamma$ by calculating $\vec{k}_1^\top\mathbf{P}_1\big(\mathbf{R}_{A,0}+\mathbf{R}_{A,1}\big)$.

To deduce the bit of α, we observe the followings: By adding $\pi_{\mathbf{V}}\big(\mathbf{R}_{A,1},\mathbf{R}_{A,2},\mathbf{R}_{A,3}\big)=0$ and $\pi_{\mathbf{V}}\big(\mathbf{R}_{B,1},\mathbf{R}_{B,2},\mathbf{R}_{B,3}\big)=0$ to the first equation of Equation (16), we obtain the equation of the form

$$\pi_{\mathbf{V}}\big(\ *,\mathbf{R}_{B,0}+\mathbf{R}_{A,2}+\alpha\mathbf{I},*\ \big)=0.$$

Since $\mathbf{R}_{A,2}=\mathbf{R}_{B,1}+\mathbf{I}$, we have $\pi_{\mathbf{V}}\big(\ *,\mathbf{R}_{B,0}+\mathbf{R}_{B,1}+(\alpha+1)\mathbf{I},*\ \big)=0$. Therefore, calculating the value of $\vec{k}_2^\top\mathbf{R}_B(1,0,0)$ provides us the value of α.

Case 3. $(i,j,k)=(0,1,0)$: For the value of α, we can just use the first relation in Equation (16). To obtain $\beta+\gamma$, use Equation (15) with $X=A$ and B and $\mathbf{R}_{B,1}=\mathbf{R}_{A,2}+\mathbf{I}$.

Case 4. $(i,j,k)=(0,0,1)$: For the value of α, it is sufficient with the first relation in Equation (16). To obtain $\beta+\gamma$, use Equation (15) with $X=A$ and C and $\mathbf{R}_{C,1}=\mathbf{R}_{A,3}+\mathbf{I}$.

For the other cases, we can proceed analogous arguments to obtain the desired results. We leave verifying them to readers. □

5 (Im)possibility of Higher Fan-In Gates Garbling

In this section, we build upon our earlier work and investigate the existence of secure garbling schemes for ℓ-input gates using s-sliced labels, where $\ell \geq 3$ and $s \geq 3$. Our previous research demonstrated that the construction presented in [1], which aims to garble tri-gates g_{tri}, leaks information about the permutation bits, rendering the scheme insecure. This construction relied on employing three-sliced labels to garble 3-input gates. The primary goal of this section is to address the following open question: Is there a secure garbling scheme for ℓ-input gates utilizing s-sliced labels, with $\ell \geq 3$ and $s \geq 3$? Unfortunately, our response is negative. Even considering $s \geq 3$ and $\ell \geq 3$ seems unlikely to yield a secure garbling scheme.

Let us begin with defining several notions. Throughout this section, we consider garbling an ℓ-input gate g with input wires $a_1, \ldots, a_\ell$ and output wire b.

Notations.

– *(Permute bits)* The values $\alpha_j \in \{0,1\}$ for $j \in \{1, \ldots, \ell\}$ are secret permute bits that are only known by the garbler.
– *(Input labels)* $A_{\alpha_j, j} \in \{0,1\}^\kappa$ are wire labels corresponding to the **false** value on the input wire a_j for each $j \in \{1, \ldots, \ell\}$.
– *(Output label)* $B \in \{0,1\}^\kappa$ represents the output wire label corresponding to the **false** value on the output wire b.
– *(Color bits)* When $A_{x_j, j}$ are held by the evaluator, we assume that the subscripts x_j, referred to as color bits, are known by the evaluator.
– *(Free-XOR)* The value $\Delta \in \{0,1\}^\kappa$ is a global offset secretly chosen by the garbler. Then, $A_{0,j} + A_{1,j} = \Delta$ for each $j \in \{1, \ldots, \ell\}$. Also, $B + \Delta$ represents the output label corresponding to the **true** on the wire b.
– *(Sliced labels)* Given an integer s, we represent the wire label $W = W^1 \| \cdots \| W^s$ as $\vec{W} = (W^1, \ldots, W^s)$, where each W^i is κ/s-bits.

Using the previously mentioned notation, we now define several matrices and vectors to establish a *garbling equation.*

– *(Linear queries)* We assume that queries to a random oracle are made on functions of the input labels. For each $j \in \{1, \ldots, \ell\}$, only one of $\vec{A}_{0,j}$ or $\vec{A}_{1,j}$ can be used as inputs to the random oracle. Particularly, we restrict our concern to the setting where the only possible queries allowed to the oracle are linear functions of the input labels. That is, we consider queries of the following form: $H\big(\sum_{j \in I} \vec{A}_{x_j, j}\big)$ for a subset $I \subset \{1, \ldots, \ell\}$. We call such queries as *linear queries.*
– *(Function representation of queries)* We continue to focus on the free-XOR setting. Given an oracle response of the form $H\big(\sum_{j \in I} \vec{A}_{x_j, j}\big)$, we represent it as a function in $\mathbb{F}_{2^{\kappa/s}}[x_1, \ldots, x_\ell]$:

$$\begin{aligned} H\big(\textstyle\sum_{j \in I} \vec{A}_{x_j, j}\big) = &\big(1 + \textstyle\sum_{j \in I} x_j\big) H\left(\textstyle\sum_{j \in I} \vec{A}_{0,j}\right) \\ &+ \big(\textstyle\sum_{j \in I} x_j\big) H\left(\vec{A}_{1,i_*} + \textstyle\sum_{j \in I \setminus \{i_*\}} \vec{A}_{0,j}\right), \end{aligned}$$

where i_* is the index such that $x_{i_*} = 1$. To understand why this representation holds, note that if there are even number of $j \in I$ such that $x_j = 1$, then we have $H\big(\sum_{j \in I} \vec{A}_{x_j,j}\big) = H\big(\sum_{j \in I} \vec{A}_{0,j}\big)$ due to the free-XOR condition. Similar arguments apply for the odd case.

- *(Query matrix)* Consider a vector $\vec{H}$ comprising $H\big(\sum_{j \in I} \vec{A}_{0,j}\big)$ and $H\big(\vec{A}_{1,i_*} + \sum_{j \in I \setminus \{i_*\}} \vec{A}_{0,j}\big)$ for all non-empty subset $I \subset \{1, \dots, \ell\}$. Based upon the aforementioned function representation of $h_I := H\big(\sum_{j \in I} \vec{A}_{x_j,j}\big)$, one can represent a linear sum of the form $\sum_I h_I$ as a dot product $\vec{M}^\top \cdot \vec{H}$, where $\vec{M}$ comprises polynomials $1 + \sum_{j \in I} x_j$ and $\sum_{j \in I} x_j$ for certain I's. In this manner, for s linear sums of $\sum_I h_I$'s, one can express $(h_1, \dots, h_s)^\top = \mathbf{M}\vec{H}$, where the k-th row of $\mathbf{M}$ corresponds to the vector $\vec{M}$ associated with h_j. We refer to the matrix $\mathbf{M}$ as a *query matrix* and the vector $\vec{H}$ as a *hash vector*.
- *(Control matrix)* For each $j \in \{1, \dots, \ell\}$, we introduce a j-th *control matrix*, denoted by $\mathbf{R}_j$, associated with the j-th input. The control matrix $\mathbf{R}_j$ is a $(s \times s)$-matrix with its entries in $\mathbb{F}_2[\alpha_1, \dots, \alpha_\ell, x_1, \dots, x_\ell]$. Since our main interest lies in the case when $\alpha_i \in \mathbb{F}_2$, to streamline notation without causing confusion, we treat its entries as elements in $\mathbb{F}_2[x_1, \dots, x_\ell]$, and their coefficients are parameterised by α_i's. As before, we also write

$$\mathbf{R}_j = \mathbf{R}_{j,0} + \mathbf{R}_{j,1} x_1 + \cdots + \mathbf{R}_{j,\ell} x_\ell,$$

  where each $\mathbf{R}_{j,k}$ is a binary matrix.[5]

Remark 1. Assume that arbitrary *non-linear* functions of the input labels are used to make queries to the random oracle. We observe that it reamins feasible to provide a function representation of the responses to these oracle queries. For instance, as we have seen from an example of Yao's garbling scheme, a response of the form $H(A_x, B_y)$ can be represented as a quadratic function. Nevertheless, we confine our attention to the setting of linear queries, consistent with prior studies [17,15,1]. Introducing non-linear queries may potentially increase communication costs since we must account for additional monomials resulting from higher-degree functions being considered. Notably, if our aim is to garble a degree-2 gate, it is unnecessary to consider non-linear queries of degree exceeding 2. Nonetheless, exploring the potential benefits of such non-linear queries in reducing communication cost when targeting higher-degree gates constitutes an intriguing subject for future investigation. We reserve this issue for further research. □

5.1 Garbling Equations

We are ready to formally define the notion of garbling equations. We will continue to utilize the previously introduced notation throughout this discussion.

[5] By a similar argument to Footnote 3, it suffices to consider only linear polynomials.

Definition 1 (Garbling Equation). *For each $i \in \{1, \dots, s\}$, let h_i be a linear sum of linear queries. For $j \in \{1, \dots, \ell\}$, let $\mathbf{R}_j$ be a j-th control matrix. The matrix $\mathbf{M}$ is the query matrix and the vector $\vec{H}$ is the hash vector associated with $(h_1, \dots, h_s)$ such that $(h_1, \dots, h_s) = \mathbf{M}\vec{H}$. Given the vector $(h_1, \dots, h_s)$ and the control matrix $\mathbf{R}_j$, we define a garbling equation $\mathcal{G}$ of a ℓ-input gate g by the following equation:*

$$\mathcal{G}_g : \mathbf{V}\begin{bmatrix}\vec{C}\\ \vec{G}\end{bmatrix} = \mathbf{M}\vec{H} + \sum_{1 \le j \le \ell} \mathbf{R}_j \vec{A}_{x_j, j} + g(x_1 + \alpha_1, \dots, x_\ell + \alpha_\ell)\vec{\Delta}.$$

Here, $\mathbf{V}$ is a matrix comprised of column basis of the $\mathbb{F}_2$-subspace generated by the columns of $\mathbf{M}$ and $[\vec{C}, \vec{G}]$ is a $(s + r)$-dimensional vector where $s + r$ is the column length of $\mathbf{V}$.

Definition 2. *For an ℓ-input gate g, let $\Pi = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be a garbling scheme associated with a garbling equation $\mathcal{G}_g$. If the garbling scheme Π is correct, then we say that Π is **correctly garbleable** with respect to the garbling equation $\mathcal{G}_g$. Moreover, if Π satisfies privacy property, then we call that Π is **privately garbleable**.*

We observe that if there exists a vector $[\vec{C}, \vec{G}]$ satisfying the garbling equation $\mathcal{G}_g$ holds for any input labels $\vec{A}_{x_j, j}$, a global offset Δ, and permute bits $\alpha_j \in \mathbb{F}_2$, then the corresponding garbling scheme Π is correct.

In what follows, we will expand upon the discussions presented earlier to encompass the garbling of arbitrary ℓ-input gates.

Definition 3. *Assume that $\mathbf{V}$ is of the form $\mathbf{V} = [\mathbf{I}_s \mid \mathbf{W}]$, where $\mathbf{I}_s$ is the s-dimensional identity matrix and $\mathbf{W}$ is a $(s \times r)$-matrix over $\mathbb{F}_2[x_1, \dots, x_\ell]$. As before, we write $\mathbf{W} = \mathbf{W}_1 x_1 + \dots + \mathbf{W}_\ell x_\ell$ for a $(s \times r)$-binary matrix. Let us consider a cokernel matrix $\mathbf{P}$ of the $(\ell s \times r)$-matrix formed by $\left[\mathbf{W}_1^\top \mid \cdots \mid \mathbf{W}_\ell^\top\right]^\top$. Abusing the notation, we denote the matrix $\mathbf{P}$ by coker$(\mathbf{W})$, i.e. coker$(\mathbf{W})$ is a binary matrix of the following form:*

$$\mathit{coker}(\mathbf{W}) := \mathbf{P} = \mathit{coker}\left(\begin{bmatrix}\mathbf{W}_1\\ \vdots\\ \mathbf{W}_\ell\end{bmatrix}\right).$$

Moreover, if the matrix $\left[\mathbf{W}_1^\top \mid \cdots \mid \mathbf{W}_\ell^\top\right]^\top$ is of rank r, then we can write coker$(\mathbf{W}) = \mathbf{P} = [\mathbf{P}_1 \mid \cdots \mid \mathbf{P}_\ell]$, where each $\mathbf{P}_j$ is a $(\ell s - r) \times s$ dimensional matrix.

For a finite field $\mathbb{F}$ of characteristic 2, we consider the $\mathbb{F}$-subspace spanned by the columns in $\mathbf{V}$ and denote it by *span*$(\mathbf{V})$:[6]

$$\mathit{span}(\mathbf{V}) = \left\{\mathbf{V}\begin{bmatrix}\vec{C}\\ \vec{G}\end{bmatrix} \middle| \vec{C} \in \mathbb{F}^s, \vec{G} \in \mathbb{F}^r\right\}.$$

[6] In the case that s-sliced labels used, we typically choose $\mathbb{F} = \mathbb{F}_{2^{\kappa/s}}$.

Similar to the previous sections, let us write $\vec{\nu} \in \mathit{span}(\mathbf{V})$ as $\vec{\nu} = \vec{\nu}_0 + \vec{\nu}_1 x_1 + \cdots + \vec{\nu}_\ell x_\ell$ for a $\mathbb{F}$-vector $\vec{\nu}_j$. Then we can check that

$$\vec{\nu} \in \mathit{span}(\mathbf{V}) \text{ if and only if } \mathbf{P}_1\vec{\nu}_1 + \cdots + \mathbf{P}_\ell \vec{\nu}_\ell = 0.$$

Again, we define the relation $\pi_{\mathbf{V}}$ by

$$\pi_{\mathbf{V}}(\mathbf{X}_1, \ldots, \mathbf{X}_\ell) := \mathbf{P}_1\mathbf{X}_1 + \cdots + \mathbf{P}_\ell \mathbf{X}_\ell,$$

where the dimension of each matrix $\mathbf{X}_j$ is properly defined.

5.2 Main Results

Next, we discuss on the conditions under which the garbling scheme Π is correct.

Lemma 1. *Assume that $\mathit{span}(\mathbf{M}) = \mathit{span}(\mathbf{V})$ in the garbling equation $\mathcal{G}_g$. If $\rho := \sum_{1 \le j \le \ell} \mathbf{R}_j \vec{A}_{x_j,j} + g(x_1 + \alpha_1, \ldots, x_\ell + \alpha_\ell)\vec{\Delta} \in \mathit{span}(\mathbf{V})$ for any $\vec{A}_{x_j,j}$ and $\vec{\Delta}$, then the garbling scheme Π is correct.*

Proof. It is obvious from the definition. Since $\mathit{span}(\mathbf{M}) = \mathit{span}(\mathbf{V})$, there exist $\vec{C}_M$ and $\vec{G}_M$ such that $\mathbf{M}\vec{H} = \mathbf{V}\big[\vec{C}_M^\top \mid \vec{G}_M^\top\big]^\top$. Similarly, as we have $\rho \in \mathit{span}(\mathbf{V})$, there exist $\vec{C}_\rho$ and $\vec{G}_\rho$ such that $\rho = \mathbf{V}\big[\vec{C}_\rho^\top \mid \vec{G}_\rho^\top\big]^\top$. Therefore, we have $\vec{C} = \vec{C}_M + \vec{C}_\rho$ and $\vec{G} = \vec{G}_M + \vec{G}_\rho$. □

In the following, let us write the ℓ-variate gate g as a polynomial of the following form:

$$g(x_1 + \alpha_1, \ldots, x_\ell + \alpha_\ell) = g^{(0)}(\alpha_1, \ldots, \alpha_\ell) + \sum_{d \ge 1} g^{(d)}_{i_1,\ldots,i_d}(\alpha_1, \ldots, \alpha_\ell) x_{i_1} \cdots x_{i_d}, \tag{18}$$

where each $g^{(d)}_{i_1,\ldots,i_d}$ is the coefficient of $x_{i_1} \cdots x_{i_d}$ in the expansion of g.

Lemma 2. *Let the notations as above. Assume that $\vec{H}$ only consists of linear queries. Then, we have $\rho \in \mathit{span}(\mathbf{V})$ for any $\vec{A}_{x_j,j}$ and $\vec{\Delta}$ if and only if (1) The degree of g is less than or equal to 2 and (2) the following equations hold:*

$$\begin{gathered}
\pi_{\mathbf{V}}\big(\mathbf{R}_{j,1}, \ldots, \mathbf{R}_{j,\ell}\big) = 0 \quad \textit{for each } j = 1, \ldots, \ell \\
\pi_{\mathbf{V}}\big(\mathbf{S}_1, \ldots, \mathbf{S}_\ell\big) = 0 \quad \textit{where } \mathbf{S}_j := \mathbf{R}_{j,0} + \mathbf{R}_{j,j} + g^{(1)}_j \mathbf{I}_s \\
\mathbf{R}_{j,k} + \mathbf{R}_{k,j} + g^{(2)}_{j,k}\mathbf{I}_s = 0 \quad \textit{for } 1 \le j \lneq k \le \ell,
\end{gathered} \tag{19}$$

where $\mathbf{I}_s$ is the s-dimensional identity matrix.

Proof. We follow a similar approach to the discussion presented in Equation (15) and (16). Let us write $\vec{A}_{x_j,j} = \vec{A}_{0,j} + x_j\vec{\Delta}$, for each j, and substitute them into the expression of ρ in Lemma 1. As before, since we are working with the case where the variable x_j takes the value in $\mathbb{F}_2$, we impose the condition that $x_j^2 = x_j$. Expanding the expression of ρ, we have the following:

$$\rho = \mathbf{R}_1\vec{A}_{0,1} + \cdots + \mathbf{R}_\ell\vec{A}_{0,\ell} + \rho_\Delta\vec{\Delta},$$

where

$$\rho_\Delta = \sum_{j=1}^{\ell}\big(\mathbf{R}_{j,0} + \mathbf{R}_{j,j}\big)x_j + \sum_{1\le j\ne k\le \ell}\big(\mathbf{R}_{j,k} + \mathbf{R}_{k,j}\big)x_jx_k + g(x_1+\alpha_1,\ldots,x_\ell+\alpha_\ell).$$

Since ρ should belong to *span*$(\mathbf{V})$ for any $\vec{A}_{0,j}$ and $\vec{\Delta}$, it should satisfy that $\mathbf{R}_j\vec{A}_{0,j} \in$ *span*$(\mathbf{V})$, for each j, and $\rho_\Delta\vec{\Delta} \in$ *span*$(\mathbf{V})$.

The condition that $\mathbf{R}_j\vec{A}_{0,j} \in$ *span*$(\mathbf{V})$ is equivalent to that

$$\pi_{\mathbf{V}}\big(\mathbf{R}_{j,1},\ldots,\mathbf{R}_{j,\ell}\big) = 0.$$

We notice that any elements in *span*$(\mathbf{V})$ contain only linear terms since we assumed that $\vec{H}$ contains only linear queries. Thus, any terms of degree ≥ 2 in ρ_Δ should vanish. This condition is equivalent to that the polynomial g is of at most degree 2 and the quadratic terms of ρ_Δ are zeros, i.e.

$$\mathbf{R}_{j,k} + \mathbf{R}_{k,j} + g^{(2)}_{j,k}\mathbf{I}_s = 0 \quad \text{for } 1 \le j \lneq k \le \ell.$$

Regarding the linear terms in ρ_Δ, it should satisfy the following:

$$\pi_{\mathbf{V}}\big(\mathbf{S}_1,\ldots,\mathbf{S}_\ell\big) = 0,$$

where each $\mathbf{S}_j = \mathbf{R}_{j,0} + \mathbf{R}_{j,j} + g^{(1)}_j\mathbf{I}_s$ is the coefficient of x_j in ρ_Δ. Therefore, we have proved our claims. □

The following theorem is a straightforward consequence of the preceding lemmata. It provides guidance on selecting the control matrices to ensure the correctness of a garbling scheme.

Theorem 2. *Assume that the vector $\vec{H}$ consists of only linear queries and the matrix $\mathbf{M}$ is of full rank in the garbling equation $\mathcal{G}_g$. Then the garbling scheme Π associated with the garbling equation $\mathcal{G}_g$ is correct, if and only if the followings hold:*

1. *The degree of the target gate g is less than or equal to 2;*
2. *The control matrix $\mathbf{R}_j$ satisfies Equation (19) for each j.*

Remark 2. From Theorem 2, we observe that if solely linear queries are permitted to the random oracle, then it becomes unfeasible to garble a gate of degree greater than or equal to 3 using the described method. Interestingly, this finding is equivalent to Corollary 4 in [1], which demonstrates the impossibility of garbling a higher fan-in gate of degree ≥ 3.

Indeed, although Theorem 2 implies that garbling high-degree gates using solely linear queries is infeasible, the prospect of constructing a garbling scheme for such gates through *non-linear queries* remains open. Exploring this avenue could yield fascinating results and warrants further investigation. □

As demonstrated in the example of the construction by Ashur et al. [1] in Section 4.2, choosing the control matrices in a manner that guarantees the correctness of a garbling scheme does not invariably imply that the scheme is also *private*. In other words, despite ensuring the correctness, the corresponding control matrices may inadvertently reveal some information about the permute bits. Consequently, we will explore the conditions under which information regarding the permute bits is compromised.

Precisely, we provide the following lemma and theorem:

Lemma 3. *For each j, suppose that there exists a non-zero vector $\vec{v}_j$ such that*

$$\vec{v}_j^\top \mathbf{P}_j \neq 0 \text{ and } \vec{v}_j^\top \mathbf{P}_i = 0 \text{ for all } i \neq j.$$

Then, given $\big(\mathbf{R}_1(i_1,\dots,i_\ell),\dots,\mathbf{R}_\ell(i_1,\dots,i_\ell)\big)$ for some $i_1,\dots,i_\ell \in \mathbb{F}_2$, one can compute the value $\tilde{g}_j(\alpha_1,\dots,\alpha_\ell)$ for each j, where $\tilde{g}_j$ is a function that is defined by the following:

$$\tilde{g}_j := g_j^{(1)} + \sum_{k \lneq j} i_k g_{k,j}^{(2)} + \sum_{j \lneq k} i_k g_{j,k}^{(2)}.$$

Proof. First, we consider the case of $j = 1$. From the relations in Equation (14), we have the following:

$$\begin{aligned}
&\pi(\mathbf{S}_1,\dots,\mathbf{S}_\ell) + (i_1+1)\pi(\mathbf{R}_{1,1},\dots,\mathbf{R}_{i,\ell}) + i_2\pi(\mathbf{R}_{2,1},\dots,\mathbf{R}_{2,\ell}) + \cdots + i_\ell\pi(\mathbf{R}_{\ell,1},\dots,\mathbf{R}_{\ell,\ell})\\
&= \pi\big(\mathbf{R}_{1,0} + i_1\mathbf{R}_{2,1} + \cdots + i_\ell\mathbf{R}_{\ell,1} + g_1^{(1)}\mathbf{I}_s,\ *\ ,\dots,\ *\ \big)\\
&= \pi\big(\widetilde{\mathbf{R}}_1 + \tilde{g}_1\mathbf{I}_s,\ *\ ,\cdots,\ *\ \big)\\
&= 0,
\end{aligned}$$

where $\widetilde{\mathbf{R}}_k := \mathbf{R}_k(i_1,\dots,i_\ell)$. In the second equality, we used the relation that $\mathbf{R}_{j,k} = \mathbf{R}_{k,j} + g_{j,k}^{(2)}\mathbf{I}_s$ from Equation (14).

Multiplying the vector $\vec{v}_1^\top$ to the both sides yields

$$\vec{v}_1^\top \mathbf{P}_1 \mathbf{R}_1(i_1,\dots,i_\ell) = \tilde{g}_1 \vec{v}_1^\top \mathbf{P}_1.$$

Thus, calculating the left-hand side, one can compute the value of $\tilde{g}_1$. Similar arguments hold for the other cases. □

The following theorem is an immediate consequence of the preceding lemma. It specifies the conditions under which the garbling scheme leaks information about the permute bits.

Theorem 3. *For each j, suppose that there exists a non-zero vector $\vec{v}_j$ such that*

$$\vec{v}_j^\top \mathbf{P}_j \neq 0 \text{ and } \vec{v}_j^\top \mathbf{P}_i = 0 \text{ for all } i \neq j.$$

Then the garbling scheme with the garbling equation $\mathcal{G}_g$ cannot be privately garbleable.

Proof. Recall that the evaluator is given $\big(\mathbf{R}_1(i_1,\dots,i_\ell),\dots,\mathbf{R}_\ell(i_1,\dots,i_\ell)\big)$ for some $i_1,\dots,i_\ell \in \mathbb{F}_2$ depending on her choices of the active input labels. By Lemma 3, she can compute the value of $\tilde{g}_j(\alpha_1,\dots,\alpha_\ell)$ for each $j = 1,\dots,\ell$. Thus, the garbling scheme leaks some information on the permute bits $\alpha_1,\dots,\alpha_\ell$. It violates the privacy property. □

5.3 Discussions

In this section, we discuss on several interesting implication of our results.

RR21 Construction. We observe that our proposed attack does not apply to the construction by Rosulek and Roy [15], implying that it remains secure against our presented attack. Recall that in their construction the matrix $\mathbf{W}$ is of the form $\mathbf{W} = \mathbf{W}_1 x + \mathbf{W}_2 y$, where:

$$\mathbf{W}_1 = \begin{bmatrix} 1\,0 & 1 \\ 0\,0 & 1 \end{bmatrix} \text{ and } \mathbf{W}_2 = \begin{bmatrix} 0\,0 & 1 \\ 0\,1 & 1 \end{bmatrix}.$$

Hence, the cokernel of $\mathbf{W}$ is $\mathit{coker}(\mathbf{W}) = \mathbf{P} = [\mathbf{P}_1 \mid \mathbf{P}_2] = [0\ 1 \mid 1\ 0]$. Since the matrix $\mathbf{P}_k$ consists of only one row, its left kernel is trivial. In other words, there exists no non-zero vector $\vec{v}$ such that $\vec{v}^\top \mathbf{P}_k = 0$, which precludes our proposed attack.

Indeed, as observed in the explicit formula presented in Section 3, the values $\mathbf{R}_A(i,j)$ and $\mathbf{R}_B(i,j)$ do not reveal any information on the permute bits α and β. Due to the enough degrees of freedom available in the choice of the control matrices, the permute bits are effectively masked by these random values, hindering the evaluator from deducing the permute bits.

When Our Attack Works. Recalling that the matrix $\mathbf{P}_i$ depends directly on the matrix $\mathbf{V} = [\mathbf{I} \mid \mathbf{W}]$, we will now present a simple criterion for determining whether our attack can be applied simply by observing the matrix $\mathbf{W}$. Recall that $\mathbf{W} = \mathbf{W}_1 x_1 + \cdots + \mathbf{W}_\ell x_\ell$, where each $\mathbf{W}_j$ is a $(s \times r)$-binary matrix. We assume that $s \le r$; we will address later why this assumption seems to hold whenever the matrix $\mathbf{V}$ includes the identity matrix $\mathbf{I}$.

In what follows, we argue that if the $\mathbf{W}_j$ is not of a full rank for some j, then our attack is applicable. Without loss of generality, assume that the rank of $\mathbf{W}_1$ is less than s. By this assumption, the rows of $\mathbf{W}_1$ are linearly dependent. Therefore, there exists a non-zero vector $\vec{p}_1$ such that $\vec{p}_1^\top \mathbf{W}_1 = 0$. Since $\big[\vec{p}_1^\top \mid 0 \mid \cdots \mid 0\big]$ is an element of the cokernel of $\big[\mathbf{W}_1^\top \mid \cdots \mid \mathbf{W}_\ell^\top\big]^\top$ and the matrix $\mathbf{P}$ is a basis matrix of the cokernel, there exists a non-zero vector $\vec{v}_1$ such that $\vec{v}_1 \mathbf{P} = [\vec{v}_1 \mathbf{P}_1 \mid \cdots \mid \vec{v}_1 \mathbf{P}_\ell] = \big[\vec{p}_1^\top \mid 0 \mid \cdots \mid 0\big]$. Thus, the vector $\vec{v}_1$ satisfies the condition in Theorem 3.

For instance, as discussed in Section 4, in the construction by Ashur et al. [1], the (3×4)-matrix $\mathbf{W}_j$ has rank 2. Hence, their construction is vulnerable to our attack.

Necessary Conditions to Succeed Our Attack. Unfortunately, the condition in Theorem 3 is not a necessary condition for our attack to succeed. We will now examine the following intriguing case study. Let us consider a three-sliced garbling scheme, as described in the construction presented by Ashur et al., i.e. we set $s = 3$. We maintain the same notation used in Section 4.2. In our example, let us take into account the following linear combinations of linear queries:

$$\begin{aligned}D^1 + g_{tri}(u,v,w)\varDelta^1 &= H(B_y + C_z) + H(A_x + B_y + C_z) + \cdots\\ D^2 + g_{tri}(u,v,w)\varDelta^2 &= H(A_x + C_z) + H(A_x + B_y + C_z) + \cdots\\ D^3 + g_{tri}(u,v,w)\varDelta^3 &= H(A_x + B_y) + H(A_x + B_y + C_z) + \cdots .\end{aligned}$$

In this case, we have the matrix $\mathbf{V} = [\mathbf{I} \mid \mathbf{W}]$ where

$$\mathbf{W} = \begin{bmatrix} y+z & 0 & 0 & x+y+z \\ 0 & x+z & 0 & x+y+z \\ 0 & 0 & x+y & x+y+z \end{bmatrix} = \mathbf{W}_1 x + \mathbf{W}_2 y + \mathbf{W}_3 z.$$

One might check that each matrix $\mathbf{W}_j$ has full rank. Moreover, we could not find a vector $\vec{v}_j$ satisfying the condition in Theorem 3. However, if we find the control matrices fullfilling Equation (19), we observe that the corresponding garbling scheme still leaks some information on the permute bits. Precisely, when the evaluator is given the control matrices at $(x, y, z) = (0, 0, 0)$, i.e. the active input labels are A_0, B_0, C_0, the evaluator can deduce the value of $\beta + \gamma$ by comparing the values $\mathbf{R}_{1,0} = \mathbf{R}_A(0, 0, 0)$ and $\mathbf{R}_{2,0} = \mathbf{R}_B(0, 0, 0)$.

Consequently, it has been observed that the rank condition on $\mathbf{W}_j$ is insufficient to construct a secure garbling scheme. It remains as an open problem to explore further when a secure garbling scheme can be constructed.

References

1. T. Ashur, C. Hazay, and R. Satish. On the feasibility of sliced garbling. Cryptology ePrint Archive, Paper 2024/389, 2024. https://eprint.iacr.org/2024/389.
2. C. Baek and T. Kim. Can we beat three halves lower bound?: (im)possibility of reducing communication cost for garbled circuits. Cryptology ePrint Archive, Paper 2024/803, 2024. https://eprint.iacr.org/2024/803.
3. M. Ball, H. Li, H. Lin, and T. Liu. New ways to garble arithmetic circuits. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2023.
4. 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.
5. 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.

6. 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.
7. 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.
8. D. Heath and V. Kolesnikov. One hot garbling. In Y. Kim, J. Kim, G. Vigna, and E. Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 574–593. ACM, 2021.
9. D. Heath, V. Kolesnikov, and L. K. L. Ng. Garbled circuit lookup tables with logarithmic number of ciphertexts. In M. Joye and G. Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 185–215. Springer, 2024.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.

16. 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.
17. 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 from the algebraic perspective. For the sake of readability, we mainly describe the technique with the example of RR21's construction.

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 $\mathbf{R} = [\overrightarrow{r_1}, \dots, \overrightarrow{r_4}]$, where $\overrightarrow{r_k}$ is the k-th column of $\mathbf{R}$. The garbler 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).

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

$$\mathbf{V}\overrightarrow{z_k} = \mathbf{M}\overrightarrow{S}^{con} + \overrightarrow{r_k}. \tag{20}$$

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 evaluator 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 $\overrightarrow{\tilde{r}_k}$ is the value of $\overrightarrow{r_k}$ evaluated at $(x, y) = (i, j)$.

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 (20) 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 $\pi_{\mathbf{V}}$ in Section 5. Recall that $\overrightarrow{r_k}$ is the column of $\mathbf{R}$. We observe that $\mathbf{R}$, thus each of its columns, satisfies the relation $\pi_{\mathbf{V}}$ 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 the RR21 construction. 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 $\mathbf{R}$.

One might observe that the number of additional ciphertexts required to encrypt the control matrices can be further reduced in RR21's construction. Let us consider the control matrices given by

$$\left[\mathbf{R}_A \mid \mathbf{R}_B\right] = \left[\begin{array}{cc|cc} r_1 & r_2 & r_2 + x & r_1 + r_2 + y \\ r_2 & r_1 + r_2 & r_1 + r_2 & r_1 + x \end{array}\right]$$

where $r_1(x, y) := \alpha x + (\beta + 1)y + c$ and $r_2(x, y) := (\beta + 1)x + (\alpha + \beta + 1)y + e$ are the polynomials in $\mathbb{F}_2[x, y]$, and the bits c and e are randomly chosen. It can be readily verified that the above control matrices yield a correct garbling scheme for RR21's construction. Thus, it is enough to send only the encryption of $(r_1, r_2)^\top$, instead of sending entire encryptions of all columns. Therefore, it reduces the number of ciphertexts garbling the control bits.

B How to Choose Control Matrices

Given the matrix $\mathbf{V}$ as defined in Equation (13), for any vector $\vec{\nu} = \vec{\nu}_0 + \vec{\nu}_1 x + \vec{\nu}_2 y + \vec{\nu}_3 z \in span(\mathbf{V})$, we obtain $\pi_{\mathbf{V}}(\vec{\nu}_1, \vec{\nu}_2, \vec{\nu}_3) = \mathbf{P}_1\vec{\nu}_1 + \mathbf{P}_2\vec{\nu}_2 + \mathbf{P}_3\vec{\nu}_3 = 0$, where

$$\mathbf{P} = [\mathbf{P}_1 \mid \mathbf{P}_2 \mid \mathbf{P}_3] = \left[\begin{array}{ccc|ccc|ccc} 1&0&1&0&0&0&0&0&0\\ 0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&1\\ 0&1&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&1&0&0 \end{array}\right]. \tag{21}$$

By computing the control matrices satisfying Equation (15) and (16), we can provide their explicit formula as follows.

B.1 Formulas for the Control Matrices

We provide explicit formulas for the control matrices.

$$
\begin{aligned}
\mathbf{R}_A &= \begin{bmatrix} a_0 & b_0 & c_0 \\ a_1 & b_1 & c_1 \\ a_0+\beta+\gamma & b_0 & c_0+\beta+\gamma \end{bmatrix} + \begin{bmatrix} a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \\ a_3 & b_3 & c_3 \end{bmatrix} x + \begin{bmatrix} a_4+1 & b_4 & c_4+1 \\ a_4+1 & b_4 & c_4+1 \\ a_4 & b_4 & c_4 \end{bmatrix} y + \begin{bmatrix} a_4 & b_4 & c_4 \\ a_4+1 & b_4 & c_4+1 \\ a_4+1 & b_4 & c_4+1 \end{bmatrix} z \\
\mathbf{R}_B &= \begin{bmatrix} d_0 & e_0 & f_0 \\ d_0+\alpha & e_0+\alpha & f_0 \\ a_1+1 & b_1+\beta+\gamma+1 & c_1+\alpha+1 \end{bmatrix} + \begin{bmatrix} a_4 & b_4 & c_4+1 \\ a_4+1 & b_4+1 & c_4+1 \\ a_4 & b_4 & c_4+1 \end{bmatrix} x + \begin{bmatrix} d_5 & e_5 & f_5 \\ d_5 & e_5 & f_5 \\ a_4+1 & b_4+1 & c_4+1 \end{bmatrix} y + \begin{bmatrix} a_4+1 & b_4+1 & c_4+1 \\ a_4+1 & b_4+1 & c_4+1 \\ a_4+1 & b_4+1 & c_4+1 \end{bmatrix} z \qquad (22) \\
\mathbf{R}_C &= \begin{bmatrix} a_1+\alpha+1 & b_1+\beta+\gamma+1 & c_1+1 \\ g_1 & h_1 & i_1 \\ g_1 & h_1+\alpha & i_1+\alpha \end{bmatrix} + \begin{bmatrix} a_4+1 & b_4 & c_4 \\ a_4+1 & b_4+1 & c_4+1 \\ a_4+1 & b_4 & c_4 \end{bmatrix} x + \begin{bmatrix} a_4+1 & b_4+1 & c_4+1 \\ a_4+1 & b_4+1 & c_4+1 \\ a_4+1 & b_4+1 & c_4+1 \end{bmatrix} y + \begin{bmatrix} a_4+1 & b_4+1 & c_4+1 \\ g_6 & h_6 & i_6 \\ g_6 & h_6 & i_6 \end{bmatrix} z,
\end{aligned}
$$

where all of the entries are binary elements. We observe that the set of the pairs of $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ is isomorphic to 24-dimensional subspace.