On the Feasibility of Sliced Garbling

Tomer Ashur[1], Carmit Hazay[2], and Rahul Satish[3*]

[1] 3MI Labs, Leuven, Belgium tomer@3milabs.tech
[2] Bar-Ilan University, Ramat Gan, Israel carmit.hazay@biu.ac.il
[3] IT University of Copenhagen, Denmark rahs@itu.dk

Abstract. Garbling schemes are one of the most fundamental objects in cryptography and have been studied extensively due to their broad applicability. The state-of-the-art is a construction in which XOR gates are free and AND gates require $3\kappa/2 + \mathcal{O}(1)$ bits per gate, due to Rosulek and Roy (CRYPTO'21). An important technique in their garbling is slicing, which partitions the labels into two equal-length slices. In this paper, we explore the feasibility of the slicing technique for garbling schemes beyond the results introduced by Rosulek and Roy, demonstrating both its potential and its limitations.

In particular, we extend this technique to demonstrate the garbling of certain higher fan-in gadgets, and then use this to show that it is possible to garble 2-input AND gates at a cost of $4\kappa/3 + \mathcal{O}(1)$ bits. We then give a separation result showing that sliced garbling cannot be used to garble higher fan-in gadgets of degree ≥ 3 when restricted to making queries that are linear functions of the input labels to the random oracle. We further demonstrate the usefulness of our techniques in the context of oblivious garbling, a newly introduced concept for capturing circuit hiding from the garbler. The complexity of our construction is superior to that of universal circuits, and grows linearly with circuit size.

Note: A bug in Theorem 1 was brought to our attention by Lei Fan, Zenghao Lu, and Hong-Sheng Zhou. The attack on the scheme that reveals the permute bits when garbled using the scheme shown in Figure 1. In particular, we have observed that there are linear dependencies in the system of equations corresponding to the scheme that can be exploited. We are currently exploring whether a workaround can be found. The sections containing the separation result and oblivious garbling are not affected by this bug.

1 Introduction

Garbling schemes have been a fundamental object in cryptography since their introduction by Yao [40] in 1986. In addition to being one of the major paradigms in achieving constant round secure computation [8, 15, 23, 28, 30, 40], it also has found numerous applications across various areas of cryptography. Computing on encrypted data [11, 36], verifiable computing [5, 17], parallel cryptography [3, 4], software protection [9, 18, 20], obfuscation of code [2, 12], functional encryption [19, 35], and key-dependent message security [1, 7], among others, are some of the areas where garbling schemes have found applications.

Most garbling techniques are presented in the gate-by-gate paradigm, which involves garbling each gate separately. Working in this paradigm requires encoding the truth table row-wise, which involves encrypting each row separately. For that, each wire is associated with two labels (bit-strings), representing the wire's semantic value as true or false. Then, the appropriate output wire labels are encrypted using the corresponding input wire labels based on the gate's truth table. Therefore, the communication complexity of the scheme is determined by the number of such ciphertexts per gate. Garbling schemes employ efficient symmetric-key primitives, making them highly practical. Due to their broad applicability, they have been extensively studied with the aim of reducing their concrete costs where the bulk of the effort has concentrated on reducing the communication complexity; see [21, 26, 27, 30, 32, 41] for a partial list. In [27], Kolesnikov and Schneider introduced the free-XOR technique, which removed communication for all XOR gates in a circuit. Due to

[*] The work was done when the author was a student at Bar-Ilan University, Israel.

this, a standard approach in designing a garbling scheme is considering circuits with 2-input XOR and AND gates.

The best communication complexity today is achieved by the three-halves garbling scheme [34] that requires communicating $\frac{3}{2}\kappa$ bits for every AND gate, and no communication is needed for XOR gates, for a given security parameter κ. Three-halves improves upon the half-gates garbling scheme [41] by introducing a technique called *slicing*. While the garbling mechanism within all garbling schemes till then involved working with the label as a whole, with the slicing technique, Rosulek and Roy suggested splitting each label into two halves and deriving each slice for the output label separately. Continuing the line initiated in [41], they showed that schemes like half-gates could be represented as a system of linear equations and modified this abstraction to garble individual slices separately rather than the entire label. To achieve this, they used the *dicing* technique, introduced by Kempka, Kikuchi and Suzuki in [24]. Namely, to ensure that the corresponding linear system of equations has a solution, the dicing technique adds a random linear combination of the input labels to each equation. In Section 3, we provide a detailed overview of the linear algebraic representation of garbling schemes as presented by Rosulek and Roy in [34].

Along with the half-gates garbling scheme in [41], Zahur, Rosulek and Evans also showed a matching lower bound of 2κ bits to garble an AND gate in a model they define as linear garbling. In a nutshell, the model restricts all operations in the garbling scheme to either making oracle calls or bitwise-XOR of strings, where the oracle captures the cryptographic object used for encryption. They then demonstrated how the linear garbling model captures most of the practical garbling techniques known at that time. While the lower bound of 2κ bits provided in [41] gives a fair idea of the communication complexity of garbling an AND gate, it is limited to a single two-input AND gate.

Following this, using techniques not captured by the linear garbling model, various works [6, 24, 38] showed how to construct better garbling schemes for a single AND gate but could not extend these to improve communication for garbling arbitrary circuits. As discussed earlier, this was overcome by the three-halves garbling scheme using the *Slicing* and *Dicing* techniques. These techniques, too, are not captured by the linear garbling model and hence provide a way to circumvent the bound.

While the linear garbling model does not capture the slicing and dicing techniques, the resulting garbling scheme still has linearity with respect to the type of operations performed. To be precise, the only permissible operation in the linear garbling model is addition over the field $\mathbb{GF}(2^{\kappa})$. In the three-halves garbling scheme, the only operation performed is addition over the field $\mathbb{GF}(2^{\frac{\kappa}{2}})$, as a result of working with slices instead of whole labels. As discussed earlier, through the dicing technique, a random linear combination of labels unknown to the evaluator is introduced as a part of the system of equations to ensure they are solvable. It, however, does not alter the kind of mathematical operation the scheme uses and hence still maintains linearity. In this work, we continue exploring the boundaries of garbled gate compression. Our goal is to understand the potential of the three-halves slicing technique, and explore the feasibility of garbling gadgets with higher fan-in using the same. In particular, *Is it possible to increase the number of slices or the number of inputs to take full advantage of the slicing technique?*

In this paper, we answer the above question in the affirmative. We show that it is feasible to garble certain gadgets of higher fan-in by increasing the number of slices, specifically for the 3-input case. We also show how these gadgets can be used in turn to garble AND gates with $\frac{4\kappa}{3} + \mathcal{O}(1)$ bits of communication while maintaining the free-XOR property. Thus, we provide an asymptotic improvement in the communication complexity. We state and prove the security of our garbling scheme under (a modified version of) the TCCR assumption [22] that captures label slicing. However, we show that these methods are not universally applicable to garbling all higher fan-in gadgets. Specifically, we establish that this approach is unsuitable for directly garbling a 3-input AND gate and other higher degree gadgets.

Additionally, we show that these extension techniques are useful in the context of a new notion, that we define as *Oblivious Garbling*. We believe this cryptographic object is of independent interest, as it captures instances where a garbler needs to remains oblivious to the actual gates used within the circuit. This notion becomes particularly pertinent in situations where one seeks to outsource the garbling task while maintaining secrecy over the circuit, possibly due to the presence of sensitive states or information.

1.1 Overview and Our Contributions

This paper improves over the state-of-the-art for garbling techniques on the two fronts. Firstly, we show it's feasible to extend the slicing technique further to achieve a garbling scheme for Boolean circuits with better communication complexity with respect to the security parameter. Then, we show the infeasibility of extending the slicing technique to garble cubic and higher degree gadgets under a reasonable set of assumptions. In addition to this, we use these techniques to design a garbling scheme that can hide the functionality of the circuit from the garbler. While previous schemes have addressed gate hiding, typically defined with respect to the evaluator, none have explicitly catered to scenarios where the *garbler* remains unaware of the circuit's structure. Thus, we formally define "oblivious garbling" as a means to encapsulate this requirement, and present a garbling scheme to achieve this notion. We give an overview of our contributions:

1. The feasibility of sliced garbling: We show it is possible extend the slicing technique from three halves to garble 3-input gadgets, while achieving better communication with respect to the security parameter. The three-halves garbling scheme [34] achieves its efficiency through two optimizations: (a) the half-gates optimization, which restricts the oracle queries to linear functions on the input labels. Non-linear queries like $H(A_i, B_j)$ can only be used for the row in the truth table corresponding to the input tuple (i, j), whereas queries like $H(A_i), H(B_j)$ and $H(A_i \oplus B_j)$ can be reused for two rows in each truth table. The ability to reuse oracle responses across multiple equations in a linear system creates linear dependencies that allow for the compression of the ciphertexts to be communicated (see [34] for a more detailed explanation regarding this), and (b) slicing, which divides the output labels into two halves and uses a distinct combination of linear queries to derive each slice separately.

A natural question is whether slicing can be extended further. For example, if we slice a label into three parts for 2-input gates, we will need to derive each part separately. However, with just one oracle $H(.)$, the only possible linear queries are $H(A)$, $H(B)$, and $H(A+B)$. This makes it impossible to find a viable set of combinations of these three queries that can garble three different slices independently and correctly. The only way to do this would be to introduce another oracle $H'(.)$, which would increase the communication cost instead of decreasing it because we will have to send ciphertexts corresponding to two oracles instead of just one.

We consider a different route by providing a framework to logically extend the garbling equation presented in [34] to garble a 3-input gadget $\mathcal{G}_{\text{tri}}(x, y, z) = x \wedge (y \oplus z)$. In Section 4.1, we provide a detailed overview of this generalization and how to garble this gadget. We then extend this approach in Section 4.2 to garble arbitrary circuits with XOR gates and $\mathcal{G}_{\text{tri}}$ gadgets. Finally, we show how such a gadget can be reduced to a 2-input AND gate while preserving the communication complexity, thus proving Theorem 1.

Any such generalization of the slicing technique to a higher fan-in setting requires simultaneously addressing the following concerns: (1) finding the right combination of oracle queries, (2) existence of linear combinations of label slices that ensure the corresponding system is solvable, and (3) concise representation of such linear combinations, all while maintaining the correctness and privacy of the garbling scheme and optimizing for the garbled gate size.

In addressing (1), we require that each combination of oracle queries yields a distinct masking value, and one such combination is used for each slice. While the three-halves scheme addressed (2) and (3) by performing an exhaustive search, such an approach is infeasible in the higher input setting, where the number of constraints in the linear combination grows double exponentially in the fan-in size n. We solve (1) and (2) completely, and (3) partially. Due to the increased dimension, a three-half approach is not readily available to us, so we leave the search for a better representation via linear algebra techniques to future work.

Finally, we describe a garbling scheme for Boolean circuits with XOR gates and AND gates, where XOR gates need no communication, and each AND gate has a garbled gate of size approximately $\frac{4\kappa}{3}$ bits. More precisely, we prove the following theorem:

Theorem (Informal) 1 *There exists a garbling scheme for arbitrary circuits comprised of 2-input XOR and AND gates, such that the size of the garbled circuit is $n \cdot \left(\frac{4\kappa}{3} + \mathcal{O}(1)\right) + \kappa$ bits, where n is the number of AND gates and κ is the security parameter.*

Note that the three-halves garbling scheme [34], which is the current state-of-the-art in communication for garbling such a circuit, would incur a cost of $n \cdot \left(\frac{3\kappa}{2} + \mathcal{O}(1)\right)$ bits. Our scheme achieves better communication complexity per AND gate with respect to κ, by making two extra calls to the oracle. Table 1 summarises the landscape of garbling schemes for circuits with XOR and AND gates:

Table 1. The landscape of garbling for circuits with XOR and AND gates, up to constants, where κ is the security parameter, and the numbers mentioned is per gate

Technique	Size (in terms of κ)		Calls to H				assump.
			Garbler		Evaluator		
	XOR	AND	XOR	AND	XOR	AND	
Classical [40]	8	8	4	4	2.5	2.5	PRF
Point-and-Permute [8]	4	4	4	4	1	1	PRF
Free-XOR [27]	0	3	0	4	0	1	CCR
GRR3 [30]+ Free-XOR	0	3	0	4	0	1	CCR
GRR2 [32]	2	2	4	4	1	1	PRF
Half-Gates [41]	0	2	0	4	0	2	CCRND
Three-Halves [34]	0	3/2	0	6	0	3	RTCCR
Our Scheme	**0**	**4/3**	**0**	**8**	**0**	**4**	3-TCCR

Security Assumption: Garbling schemes can be instantiated under different security assumptions, as shown in Table 1. While schemes that do not employ the free-XOR optimization can be instantiated using pseudo-random functions, this optimization requires a stronger assumption called Circular Correlation Robustness (CCR), formalized in [13]. In this paper, we prove our scheme under the Tweakable CCR assumption, which was introduced in [22], that instantiates the oracle with a tweakable block cipher. We note that the three-halves garbling scheme has been proven secure under the Randomized TCCR (RTCCR) assumption, which is implied by TCCR. Our scheme can also be proven secure under the RTCCR assumption by slight modifications to the proof shown in Appendix 4.2.

2. Limitations of linear queries for gadgets: As discussed earlier, we restricted the garbling scheme to allow only linear queries for the reasons suggested by Rosulek and Roy in [34], i.e., queries on linear combinations of input labels. When formalizing this notion, we observe that the functions garbled by the garbler are at most quadratic in the 2-input case, whereas for a 3-input AND gate, the corresponding Boolean function contains a cubic term. We show that when restricted to making only linear queries, one cannot use the garbling equation to garble a 3-input AND gate. We later extend this result to show that this applies to any higher fan-in gadget, with degree greater than 3. The degree of a gadget refers to the multiplicative depth of the gadget when we represent it in terms of just AND and XOR gates. We prove the following theorem:

Theorem (Informal) 2 *Let g be a Boolean gadget of degree $d \geq 3$. Then, no n-sliced extension of the garbling scheme in [34] can garble g, provided the queries to the oracle are limited to be a linear function of the input labels.*

To prove Theorem 2, we first show that every gadget of degree d, introduces a multi-linear term of total degree d in the system of linear equations. Then, we proceed to show that when restricted to linear queries, for all extensions of [34], the corresponding system of linear equations is solvable only if the total degree is at most two. As the gadget has a degree higher than 2, the corresponding system of equations remains unsolvable, hence proving the claim.

In particular, we notice that the motivation behind using linear queries for 2-input gates, as suggested by Rosulek and Roy in [34], no longer remains valid for degrees higher than 2. The reason behind avoiding non-linear queries was to be able to reuse oracle responses on queries for encrypting multiple rows of the corresponding truth table. For instance, for a 3-input AND gate, the query $H(A_0, B_0)$ can be used for two rows, namely $(a, b, c) = (0, 0, 0)$ or $(0, 0, 1)$. This no longer holds for gates of higher fan-in. Therefore, we conclude that linear queries are insufficient for garbling gates with a high degree in this framework. We leave it as future work to explore the consequences of using non-linear queries.

3. Oblivious Garbling schemes: As we consider the extensions of the slicing technique to garble higher input gadgets, specifically the $\mathcal{G}_{\text{tri}}(x, y, z) = x \wedge (y \oplus z)$ gadget introduced earlier, we notice that this gadget exhibits a unique property: it can be programmed to function as either an AND or an XOR gate by setting specific inputs. For example, setting z=0 turns it into an AND, while setting x=1 creates an XOR. This versatility allows it to be considered a "universal" gadget. This programmability opens up exciting possibilities, especially in the context of garbled circuits. Traditionally, garbling a circuit involves encrypting its truth table, which requires knowledge of the logic implemented by each gate. Consequently, most existing schemes require the garbler to have complete knowledge of the circuit structure.

This programmability is interesting because it can be used to hide the gate that the gadget implements, if one could hide the inputs to the gate. This could be particularly useful in instances when dealing with circuits embedded with secrets, like those used in laconic function evaluation [14,33]. It also applies to scenarios where garbling is outsourced to cloud computing or specialized hardware accelerators, while keeping the circuit itself hidden. Traditionally, garbling a circuit involves encrypting its truth table, which requires knowing the logic implemented by each gate. For this reason, most schemes so far require the garbler to know the circuit structure. In this paper, we show how to use our garbling scheme for $\mathcal{G}_{\text{tri}}$ gadgets to achieve a scheme that can hide the implemented circuit from the garbler, even if the topology of the circuit is revealed. To achieve this, we use the fact that the $\mathcal{G}_{\text{tri}}$ gadget can be reduced to either an XOR or an AND gate based on fixing certain values to the wire x or z respectively. We then show how to use the soldering techniques to connect independently garbled $\mathcal{G}_{\text{tri}}$ gadgets to realise an oblivious garbling scheme for any arbitrary boolean circuit. In particular, we prove the following theorem:

Theorem (Informal) 3 *There exists an oblivious garbling scheme for arbitrary circuits comprised of 2-input AND and XOR gates, such that the size of the garbled gate is $\mathcal{O}(n)$ bits, where n is the size of the circuit.*

While it appears that the same can be achieved using universal gates, such as NAND gates, the functionality it implements is revealed when the garbler knows the topology. Our scheme manages to hide the circuit from the garbler even while leaking the topology. Another possible approach to oblivious garbling is to use universal circuits [25, 29, 37, 42]. A universal circuit, UC, refers to a circuit that can be programmed to simulate any Boolean circuit C up to a given size n. That is, a UC takes as input program bits p_C (which encode C) in addition to an input x, and produces as output $UC(x, p_C) = C(x)$. While this approach works, it was shown in [39] that to simulate arbitrary circuits of size n, one would need the size of UC to be $\Omega(n \log n)$. Our scheme represents a significant advance over this, since it incurs computational and communication complexity linear in the circuit size $|f|$. Moreover, our scheme ensures that each gate of the circuit is garbled independently, making it highly parallelizable. To achieve this, we replace all the gates with the $\mathcal{G}_{\text{tri}}$ gadget and show how can we let the evaluator program each gadget during evaluation to privately evaluate the circuit. We discuss this in detail in Section 6.

2 Preliminaries

2.1 Notations

For the rest of the paper, we use $\vec{X}$ to denote any array $[X_1, \cdots, X_n]$. When a vector $\vec{X}$ is used inside a matrix, we mean to include all its elements as a part of the column it is used in. For example, for $\vec{X} = [X_1, X_2]$ and $\vec{Y} = [Y_1, Y_2]$, and a matrix A:

$$A \begin{bmatrix} \vec{X} \\ \vec{Y} \end{bmatrix} = A \begin{bmatrix} X_1 \\ X_2 \\ Y_1 \\ Y_2 \end{bmatrix}$$

For any matrix M, we denote by colspace(M), the vector space generated by the columns of the matrix M. By I_n, we denote an identity matrix of dimensions $n \times n$, and by $0_{m \times n}$, we denote a zero matrix of dimensions $m \times n$. For discussing 2-input gates, we model the gate g as having a, b as input wires and c as the output wire. For 3-input gates similarly, we have a, b, c as input wires and d as the output wire. We designate two labels for a wire x, a 0-label X_0, and a 1-label X_1. So, for wire a, we have labels $\{A_0, A_1\}$. We stick to the free-XOR setting of [27] and denote by Δ the global free-XOR offset. For schemes that use slicing, we denote by X_0^i, the i'th slice of label X_0. We denote sliced wire labels as a vector for concise representations in equations. For schemes that require k-slices of the wire labels, we represent wire label $A_0 = A_0^1 || \cdots || A_0^k$ as $\vec{A_0} = [A_0^1, \cdots, A_0^k]$. As we use Δ, the global free-XOR offset, we will also represent that as a sliced vector $\vec{\Delta} = [\Delta^1, \cdots, \Delta^k]$.

We next define computationally indistinguishablity.

Definition 1 *Let $\{X_n\}, \{Y_n\}$ be sequences of distributions with X_n, Y_n ranging over $\{0,1\}^{\ell(n)}$ for some $\ell(n) = n^{O(1)}$. Then, $\{X_n\}$ and $\{Y_n\}$ are computationally indistinguishable (notation: $X_n \overset{c}{\approx} Y_n$) if for every polynomial-time A and polynomially-bounded ϵ, and sufficiently large n*

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leq \epsilon(n)$$

2.2 Definition for Garbling Schemes

We use and state the definition in [34], a slightly modified definition of garbling schemes to that of [10]. We rewrite it here for the sake of completeness.

Definition 2 *A garbling scheme consists of four algorithms:*

1. $\mathsf{Garble}(1^\kappa, f) \rightarrow (F, e, d)$ *which takes in as input the security parameter κ, and the functionality f in the form of a circuit and returns the garbled circuit F, encoding information e for the wire labels and the decoding information d for the output wire labels.*
2. $X := \mathsf{Encode}(e, x)$ *which takes in as input the encoding information e, input x and gives the Garbled input X.*
3. $Y := \mathsf{Eval}(F, X)$ *which takes in as input the garbled circuit F, the garbled input X, and returns the garbled evaluation of f on x, as Y.*
4. $y := \mathsf{Decode}(d, Y)$ *which takes in as input the decoding information d, garbled output Y and gives the output y.*

We want the following properties from a garbling scheme:

Correctness: *For any circuit f and input x, while $(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$, we have $f(x) = \mathsf{Decode}(d, \mathsf{Eval}(\mathsf{Encode}(e, x)))$ except for negligible probability.*

Privacy: *There is a simulator Sim such that for any circuit f and input x, the following distributions are indistinguishable:*

```
$(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$
$X := \mathsf{Encode}(e, x)$
$return\ (F, X, d)$
```

```
$(F, X, d) \leftarrow \mathsf{Sim}(1^\kappa, \Phi(f), f(x))$
$return\ (F, X, d)$
```

Obliviousness: *There is a simulator* Sim *such that for any circuit f and input x, the following distributions are indistinguishable:*

```
$(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$
$X := \mathsf{Encode}(e, x)$
$return\ (F, X)$
```

```
$(F, X) \leftarrow \mathsf{Sim}(1^\kappa, \Phi(f))$
$return\ (F, X)$
```

Authenticity: *For any circuit f and input x, no PPT adversary* Adv *can make the following distribution output* TRUE *except with negligible probability:*

```
$(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$
$X := \mathsf{Encode}(e, x)$
$Y \leftarrow \mathsf{Adv}(F, d, X)$
$return\ \mathsf{Decode}(d, Y) \notin \{f(x), \perp\}$
```

2.3 Tweakable Circular Correlation Robust Hash Functions

We use the TCCR assumption as stated in [22] by making modifications to the input parameters and incorporating the fact that the hash output is $\kappa/3$ bits long, and we can now XOR the three-slices of Δ, namely Δ_1,Δ_2 and Δ_3 to the hash output.

Definition 3 *3-Sliced Tweakable Circular Correlation Robustness (3-TCCR): Let $\mathcal{H} = \{H : \{0,1\}^\kappa \times \mathcal{T} \to \{0,1\}^{\kappa/3}\}$ be a set of functions for a defined set of tweaks $\mathcal{T}$, and let $\mathcal{R}$ be a distribution on $\{0,1\}^\kappa$. Let $F_{m,n}$ be the set of all functions mapping m bit strings to n bit strings. For a uniformly sampled $\Delta = (\Delta^1||\Delta^2||\Delta^3)$ from $\mathcal{R}$ such that $|\Delta^i| = \frac{\kappa}{3}$ and $i \in \mathcal{T}$, let*

$$\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}(x, i, (b_1, b_2, b_3)) := H(x \oplus \Delta, i) \oplus b_1 \cdot \Delta^1 \oplus b_2 \cdot \Delta^2 \oplus b_3 \cdot \Delta^3$$

For a distinguisher D, define:

$$Adv_{H,\mathcal{R}}^{3-\mathsf{tccr}}(D) := |\Pr_{\Delta \leftarrow \mathcal{R}}[D^{\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}(.)} = 1] - \Pr_{f \leftarrow \mathsf{F}_{2\kappa+3,\frac{\kappa}{3}}}[D^{f(.)} = 1]|$$

where we require that D never queries $(x, i, L = (b_1, b_2, b_3))$ for any x such that it can trivially find Δ. We say that H is $(t, q, \rho, \epsilon)-$ 3-tweakable circular correlation robust if for all D running in time at most t, and making at most q queries to $\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}(.)$, and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $Adv_{H,\mathcal{R}}^{3-\mathsf{tccr}} \leq \epsilon$.

3 Linear Garbling as a System of Equations Revisited

Zahur, Rosulek and Evans in [41] introduced the linear garbling model. The model encompassed most of the practical garbling constructions known at that time. They also proved a lower bound on the concrete communication complexity of a garbled gate for a 2-input AND gate. We now discuss how some existing schemes in the linear garbling model can be seen as a system of linear equations, as shown in [34]. We only discuss garbling AND gates as XOR is free.

Notations: For this section, we consider 2-input AND gates with input wires a, b and output wire c. Each wire x has two wire labels, a 0-label X_0 and a 1-label X_1. To comply with the free-XOR framework [27], we also have a global $\varDelta$ such that $X_1 = X_0 \oplus \varDelta$ for all wire labels. Therefore we can always consider a pair of wire labels to be denoted $\{X_0, X_0 \oplus \varDelta\}$. We denote by G_{ij} the ciphertext for the row $2i + j$ in the permuted table. Therefore, the garbled gate is represented as $G = [G_{00}, G_{01}, G_{10}, G_{11}]$. We also assume oracle access of the garbler and evaluator to a hash function H satisfying some security assumption. We also assume the point and permute optimization suggested by Beaver, Micali and Rogaway in [8].

Yao's Garbling Scheme: We now describe Yao's Garbling Scheme, used along with the point and permute technique. The garbler encrypts the labels using a key and generates four ciphertexts $G_{00}, G_{01}, G_{10}, G_{11}$, corresponding to each row of the permuted truth table. A key is derived by hashing the input labels corresponding to that particular row. This gives the following four equations, assuming permute bits $p_a = 1, p_b = 0$ (the second row encrypts $C \oplus \varDelta$):

$$\begin{aligned}
G_{00} &= H(A_0, B_0) \oplus C \\
G_{01} &= H(A_0, B_1) \oplus (C \oplus \varDelta) \\
G_{10} &= H(A_1, B_0) \oplus C \\
G_{11} &= H(A_1, B_1) \oplus C
\end{aligned}$$

GRR3 Row Reduction: Randomly sampling A, B, C and a fixed global offset $\varDelta$ fixes the garbled gate G above for the sampled values. Hence its size is also fixed to 4κ bits. However, as observed by Naor, Pinkas and Sumner in [30], we need a uniformly sampled C. Since the output of H is already pseudorandom, we can set $C = H(A_0, B_0)$, which would imply $G_{00} = 0$, hence reducing the size of the garbled gate to 3κ. The set of equations then reduces to:

$$\begin{aligned}
0 &= H(A_0, B_0) \oplus C \\
G_{01} &= H(A_0, B_1) \oplus (C \oplus \varDelta) \\
G_{10} &= H(A_1, B_0) \oplus C \\
G_{11} &= H(A_1, B_1) \oplus C
\end{aligned}$$

Half-Gates: Zahur, Rosulek and Evans showed in [41] how to further reduce the size of a garbled gate to 2κ bits. This is done by making one of the ciphertexts depend on the other two ciphertexts in the garbled gate and the active input label A that the evaluator holds. They achieve this by replacing the term $H(A, B)$ in the previous schemes with $H(A) \oplus H(B)$, which results in the following equations:

$$\begin{aligned}
0 &= H(A_0) \oplus H(B_0) \oplus C \\
G_{01} &= H(A_0) \oplus H(B_1) \oplus (C \oplus \varDelta) \oplus A_0 \\
G_{10} &= H(A_1) \oplus H(B_0) \oplus C \\
G_{11} &= H(A_1) \oplus H(B_1) \oplus C \oplus (A_0 \oplus \varDelta)
\end{aligned}$$

By adding the first three equations, we can see that G_{11} is just the sum of G_{10} and G_{01}. We now restructure the half-gates equation by shifting everything the garbler can't control to one side of the equation. The following are the values that the garbler cannot control:

1. *Hash outputs*: The hash outputs are pseudorandom, so they cannot be controlled by the garbler.
2. *The value of $\varDelta$*: This value of $\varDelta$ is fixed for the entire circuit and hence cannot change for every gate. Though it is possible for the garbler to control the value of $\varDelta$ for one input gate, it cannot be true for all gates.
3. *Input wire labels for a gate*: Recall that we stick to gate-by-gate garbling. Also, the GRR3 optimization dictates that the output label for each gate needs to be derived as a function of the hash outputs of input labels. So, the garbler has control over the input labels only for the input gates. Beyond this, all other labels are a function of the hash outputs, hence not something the garbler can control.

We move these values to the right side of the half-gates equation. On the left are the elements the garbler has to compute: the output label C and the garbled gate $G = \{G_1, G_2\}$.

$$\begin{aligned} C &= H(A_0) \oplus H(B_0) \\ C \oplus G_1 &= H(A_0) \oplus H(B_1) \oplus \Delta \oplus A_0 \\ C \oplus G_2 &= H(A_1) \oplus H(B_0) \\ C \oplus G_1 \oplus G_2 &= H(A_1) \oplus H(B_1) \oplus (A_0 \oplus \Delta) \end{aligned}$$

The above set of equations can be viewed as a system of equations, $\mathbf{A}x = b$, with its solution $x = [C, G_1, G_2]$, matrix A being the matrix on the left, and everything on the right computing to the vector b. For half-gates, the following system holds:

$$\begin{bmatrix}1&0&0\\1&1&0\\1&0&1\\1&1&1\end{bmatrix}\begin{bmatrix}C\\G_1\\G_2\end{bmatrix} = \begin{bmatrix}1&0&1&0\\1&0&0&1\\0&1&1&0\\0&1&0&1\end{bmatrix}\begin{bmatrix}H(A_0)\\H(A_1)\\H(B_0)\\H(B_1)\end{bmatrix} \oplus \begin{bmatrix}0&0&0\\1&0&0\\0&0&0\\1&0&1\end{bmatrix}\begin{bmatrix}A_0\\B_0\\\Delta\end{bmatrix} \oplus \begin{bmatrix}0\\1\\0\\0\end{bmatrix}\Delta \tag{1}$$

Denoting the matrices by variables V, M, R, and T in Equation (1), we have:

$$V\begin{bmatrix}C\\G_1\\G_2\end{bmatrix} = M\begin{bmatrix}H(A_0)\\H(A_1)\\H(B_0)\\H(B_1)\end{bmatrix} \oplus R\begin{bmatrix}A_0\\B_0\\\Delta\end{bmatrix} \oplus T\Delta \tag{2}$$

Here T represents the output column of the permuted truth table, and R represents the combinations of output labels added to create a system of equations that have a solution. Multiplying the left inverse of the matrix V to both sides gives the solution to this system, and the garbler can then send $G = [G_1, G_2]$ to the evaluator.

Note that we can rewrite the garbling equations as a system of linear equations, even for Yao's Garbling scheme or the GRR3 row reduction technique. Rosulek and Roy made this observation in [34], showing that most practical garbling schemes could be seen as a system of linear equations, where the solution to this system is the garbled gate.

Three-Halves: Rosulek and Roy [34] showed how one could improve the communication complexity beyond the half-gates scheme using a novel technique called **slicing**. They suggested that one can slice the output labels into two halves of size $\frac{\kappa}{2}$ bits each and then garble each slice separately. Using the half-gates scheme, garbling each slice separately would generate a garbled gate of size $2 \times \frac{\kappa}{2} = \kappa$ bits for each output slice. For instance, to garble the left slice, one would generate the garbled gate, $G_L = [G_1^L, G_2^L]$; and $G_R = [G_1^R, G_2^R]$ for the right slice. This scheme will still require 2κ bits of communication as there are two slices. However, if the two garbled gates could share a ciphertext, say by having $G_2^L = G_2^R$, then the size of the garbled gate would be $\frac{3\kappa}{2}$ bits, as the shared ciphertext would only have to be communicated once.

They then showed how one could garble each slice such that they indeed share a ciphertext. They firstly notice that along with the oracle responses $H(A_0), H(A_1), H(B_0)$, and $H(B_1)$ that are used in half-gates, one could also query the oracle on $A_0 \oplus B_0$ and $A_0 \oplus B_1$. Next, they used these additional oracle queries to garble the left and the right slices separately while ensuring they share one ciphertext. The three-halves garbling scheme uses the following combination of the oracle queries to mask each output slice:

$$\begin{aligned} C_k^L &\hookleftarrow H(A_i) \oplus H(A_i \oplus B_j) \\ C_k^R &\hookleftarrow H(B_j) \oplus H(A_i \oplus B_j) \end{aligned}$$

where A_i and B_j are the input labels and C_k^L, C_k^R are the left and right slices of the output label, representing the inputs (i, j) and the output k respectively.

Recall that for the half-gates scheme, we can rewrite the encryption of labels corresponding to each row as a system of equations after choosing the combination of oracle queries as $H(A_i) \oplus H(B_j)$ as the masking values for each row corresponding to inputs (i, j), where A_i, B_j are the input labels. Consequently, in this case, where there are two output slices, we can derive the following two systems of equations, one for each slice:

For the left slice C_0^L we have:

$$\begin{bmatrix}1&0&0\\1&0&1\\1&1&1\\1&1&0\end{bmatrix}\begin{bmatrix}C_0^L\\G_1^L\\G_2^L\end{bmatrix}=\begin{bmatrix}1&0&1&0\\1&0&0&1\\0&1&0&1\\0&1&1&0\end{bmatrix}\begin{bmatrix}H(A_0)\\H(A_1)\\H(A_0\oplus B_0)\\H(A_0\oplus B_1)\end{bmatrix}\oplus R_1\begin{bmatrix}A_0^L\\A_0^R\\B_0^L\\B_0^R\\\Delta_L\\\Delta_R\end{bmatrix}\oplus T\Delta_L \tag{3}$$

and for the right slice C_0^R, we have:

$$\begin{bmatrix}1&0&0\\1&1&1\\1&0&1\\1&1&0\end{bmatrix}\begin{bmatrix}C_0^R\\G_1^R\\G_2^R\end{bmatrix}=\begin{bmatrix}1&0&1&0\\0&1&0&1\\1&0&0&1\\0&1&1&0\end{bmatrix}\begin{bmatrix}H(B_0)\\H(B_1)\\H(A_0\oplus B_0)\\H(A_0\oplus B_1)\end{bmatrix}\oplus R_2\begin{bmatrix}A_0^L\\A_0^R\\B_0^L\\B_0^R\\\Delta_L\\\Delta_R\end{bmatrix}\oplus T\Delta_R \tag{4}$$

Let us denote in the above system of equations with variables V_1, M_1, V_2, M_2 like we did for the half-gates equation. Thus, we have the following system of equations to solve for:

$$V_1\begin{bmatrix}C_0^L\\G_1^L\\G_2^L\end{bmatrix}=M_1\begin{bmatrix}H(A_0)\\H(A_1)\\H(A_0\oplus B_0)\\H(A_0\oplus B_1)\end{bmatrix}\oplus R_1\begin{bmatrix}A_0^L\\A_0^R\\B_0^L\\B_0^R\\\Delta_L\\\Delta_R\end{bmatrix}\oplus T\Delta_L \tag{5}$$

$$V_2\begin{bmatrix}C_0^R\\G_1^R\\G_2^R\end{bmatrix}=M_2\begin{bmatrix}H(B_0)\\H(B_1)\\H(A_0\oplus B_0)\\H(A_0\oplus B_1)\end{bmatrix}\oplus R_2\begin{bmatrix}A_0^L\\A_0^R\\B_0^L\\B_0^R\\\Delta_L\\\Delta_R\end{bmatrix}\oplus T\Delta_R \tag{6}$$

As one can see, for Equation (5), the garbled gate $G_L = [G_1^L, G_2^L]$ is a part of its solution. Similarly, the garbled gate for the second slice $G_R = [G_1^R, G_2^R]$ is a part of the solution for Equation (6). We now discuss how to find the solution for both these systems of equations to compute both the garbled gates. Due to the symmetric nature of these systems, we demonstrate it only for the left slice. All techniques, however, extend to the right slice, too.

Let us consider the system represented by Equation (5). Consider a left inverse of the matrix V_1, denoted by V_1^{-1}. Note that V_1 is a full-rank matrix; otherwise, we could reduce the number of ciphertexts required to compute the equations on the right-hand side of the system. Therefore, at least one left inverse for V_1 exists. On multiplying V_1^{-1} on both sides of Equation (5), we get the following solution:

$$\begin{bmatrix}C_0^L\\G_1^L\\G_2^L\end{bmatrix}=V_1^{-1}M_1\begin{bmatrix}H(A_0)\\H(A_1)\\H(A_0\oplus B_0)\\H(A_0\oplus B_1)\end{bmatrix}\oplus V_1^{-1}\left(R_1\begin{bmatrix}A_0^L\\A_0^R\\B_0^L\\B_0^R\\\Delta_L\\\Delta_R\end{bmatrix}\oplus T\Delta_L\right) \tag{7}$$

In Equation (7), the solution vector can be divided into two components. The first component corresponds to the terms derived from the oracle responses, represented by $V_1^{-1}M_1$. The second component corresponds to the slices of the input labels and Δ, represented by $V_1^{-1}R_1$ and $V_1^{-1}T$. Based on this understanding, we can create two subsystems of equations whose solutions will sum up to the solution of Equation (5). Let's consider these two subsystems:

Subsystem 1 corresponds to the matrix M_1 operating on the vector of oracle responses, with a solution vector $[C_{0,H}^L, G_{1,H}^L, G_{2,H}^L]$:

$$\begin{bmatrix} C_{0,H}^L \\ G_{1,H}^L \\ G_{2,H}^L \end{bmatrix} = V_1^{-1} M_1 \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix} \tag{8}$$

Subsystem 2 corresponds to only the matrices R_1, T operating on the slices of input labels and Δ, with a solution vector $[C_{0,X}^L, G_{1,X}^L, G_{2,X}^L]$:

$$\begin{bmatrix} C_{0,X}^L \\ G_{1,X}^L \\ G_{2,X}^L \end{bmatrix} = V_1^{-1} \left(R_1 \begin{bmatrix} A_0^L \\ A_0^R \\ B_0^L \\ B_0^R \\ \Delta_L \\ \Delta_R \end{bmatrix} \oplus T\Delta_L \right) \tag{9}$$

As stated earlier, the sum of solutions obtained in Equations (8) and (9) is a solution to (5). For Equation (8), by substituting the values of the matrices V_1 and M_1 from (3), we have a solution of the following form:

$$\begin{aligned} C_{0,H}^L &= H(A_0) \oplus H(A_0 \oplus B_0) \\ G_{1,H}^L &= H(A_0) \oplus H(A_1) \\ G_{2,H}^L &= H(A_0 \oplus B_0) \oplus H(A_0 \oplus B_1) \end{aligned}$$

Similarly, the system of equations corresponding to the right slice in Equation (6) will also split into the following two subsystems: Subsystem 1 for the right slice corresponds to the matrix M_2 operating on the vector of oracle responses, with a solution vector $[C_{0,H}^R, G_{1,H}^R, G_{2,H}^R]$:

$$\begin{bmatrix} C_{0,H}^R \\ G_{1,H}^R \\ G_{2,H}^R \end{bmatrix} = V_2^{-1} M_2 \begin{bmatrix} H(B_0) \\ H(B_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix} \tag{10}$$

Subsystem 2 for the right slice corresponds to only the matrices R_2, T operating on the slices of input labels and Δ, with a solution vector of $[C_{0,X}^R, G_{1,X}^R, G_{2,X}^R]$:

$$\begin{bmatrix} C_{0,X}^R \\ G_{1,X}^R \\ G_{2,X}^R \end{bmatrix} = V_2^{-1} \left(R_2 \begin{bmatrix} A_0^L \\ A_0^R \\ B_0^L \\ B_0^R \\ \Delta_L \\ \Delta_R \end{bmatrix} \oplus T\Delta_R \right) \tag{11}$$

Extending the splitting technique used for the left slice, we get the following solution for the subsystem in Equation (10):

$$\begin{aligned} C_{0,H}^R &= H(B_0) \oplus H(A_0 \oplus B_0) \\ G_{1,H}^R &= H(B_0) \oplus H(B_1) \\ G_{2,H}^R &= H(A_0 \oplus B_0) \oplus H(A_0 \oplus B_1) \end{aligned}$$

The above solutions show that $G^L_{2,H} = G^R_{2,H}$. It is evident that the first subsystems of both slices share a ciphertext. If a shared ciphertext exists between the second subsystems of both slices, we will have a shared ciphertext overall. However, for the second subsystem, as opposed to half-gates, no matrices R_1, R_2 satisfy this set of equations for all values of T. Nevertheless, given T, there exist matrices R_1 and R_2 that satisfy the above equation system. One cannot send them in plain, as they contain information about T, which is the permuted truth table, and hence can reveal the permute bits. In the three-halves scheme [34], Rosulek and Roy solve this problem through the idea of **dicing**, introduced by Kempka, Kikuchi and Suzuki in [24]. They find R_1, R_2 that work for a chosen T, compute encryptions of these matrices, and send the ciphertexts to the evaluator such that it can only decrypt the input labels added to the active row. This is called dicing. In three-halves [34], Rosulek and Roy showed the existence of matrices R_1, R_2, for any given T, using a brute-force search from the space of all possible matrices satisfying some given conditions. They combine the systems in Equations (5)-(6) to a single system of equations:

$$V \begin{bmatrix} C_L \\ C_R \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = M \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix} \oplus R \begin{bmatrix} A_L \\ A_R \\ B_L \\ B_R \\ \Delta_L \\ \Delta_R \end{bmatrix} \oplus T \begin{bmatrix} \Delta_L \\ \Delta_R \end{bmatrix}$$

Which can be rewritten as:

$$V \begin{bmatrix} \vec{C_0} \\ \vec{G} \end{bmatrix} = M\vec{H} \oplus R \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{\Delta} \end{bmatrix} \oplus T\vec{\Delta} \tag{12}$$

As R in Equation (12) was sampled for a given T, it contains information about the permute bits and cannot be sent in plain to the evaluator. However, the evaluator only needs the values of the matrix R, corresponding to the rows of the active inputs. As we already know, the garbling equation is a tool at our disposal to reveal information to the evaluator only for the active rows. Rosulek and Roy showed how to encrypt these control matrices using the garbling equation. The matrix R has four columns in this case, and they showed how to encrypt each column with five additional bits of ciphertext. Hence, with an additional ciphertext of 20 bits, one could encrypt the chosen control matrix. We discuss more about how to use this method later in Section 4 when we reuse this technique in our scheme. Rosulek and Roy also showed a series of optimizations to bring this cost down to 5 bits, hence setting the cost of the garbled gate to $3\frac{\kappa}{2} + 5$ bits.

4 Our Garbling Scheme

In this section, we present a garbling scheme that can garble any circuit comprised of 2-input AND and XOR gates with communication complexity $n \cdot \left(4\frac{\kappa}{3} + 63\right) + \kappa$ bits, where n is the number of AND gates in the circuit. To achieve this, we first discuss how to garble a specific 3-input gadget

$$\mathcal{G}_{\text{tri}} = x \wedge (y \oplus z)$$

with the size of the garbled gate for the gadget being $4\frac{\kappa}{3} + 63$ bits. Later, we will show how to extend this technique to garble circuits that are comprised of 2-input XOR gates and $\mathcal{G}_{\text{tri}}$ gadgets. Finally, we will show how to construct a garbling scheme for a circuit with 2-input XOR gates and 2-input AND gates from a garbling scheme for a circuit with 2-input XOR gates and $\mathcal{G}_{\text{tri}}$ gadgets while preserving the communication complexity of the scheme. This will allow us to garble any circuit with 2-input XOR and AND gates using our garbling scheme. All the schemes we present are proven secure, assuming 3TCCR assumption as stated in Definition 3, while supporting the free-XOR optimization.

4.1 Garbling Scheme for $\mathcal{G}_{\text{tri}}$ Gadgets

In this section, we will discuss how to extend the three-halves garbling scheme to garble the 3-input gadget $\mathcal{G}_{\text{tri}}$. Recall that the previous section discussed how to garble a 2-input AND gate using the three-halves scheme and how the scheme corresponds to the system represented in Equation (12). We extend this equation for directly garbling the 3-input gadget $\mathcal{G}_{\text{tri}}$. For this, we have to account for an additional input label, as the gadget has three inputs.

Let A_0, B_0, C_0 be the 0-labels of the input wires a, b and c; D_0, the 0-label for the output wire d, and $\mathit{\Delta}$ be the global free-XOR offset. Consequently, we have the following system:

$$V \begin{bmatrix} \vec{D_0} \\ \vec{G} \end{bmatrix} = M\vec{H} \oplus R \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{C_0} \\ \vec{\mathit{\Delta}} \end{bmatrix} \oplus T\vec{\mathit{\Delta}} \tag{13}$$

The system of linear equations, represented by Equations (13), provides an abstract representation of a sliced garbling scheme for a 3-input gadget. To translate this abstract representation into a concrete scheme, it is necessary to determine the specific values for the involved matrices and vectors. These values play a vital role for two main reasons. Firstly, they determine the feasibility of a scheme by ensuring that the system is solvable. Secondly, through careful selection, these values contribute to optimizing the size of the garbled gate. In the following, we will delve into the key parameters that determine the values of these matrices and vectors.

1. The number of slices: If each wire label is of size κ bits and we divide it into s equal slices, then by definition, each slice would be of size $\frac{\kappa}{s}$. It is an essential parameter because it also decides the size of each ciphertext in the garbled gate. We define a ciphertext to be a slice's encryption, and hence would also be of size $\frac{\kappa}{s}$. In our scheme, we make three slices to the wire labels, and hence each component of $\vec{G}$, would be of size $\frac{\kappa}{3}$. However, we leave it as an open problem to determine if this is the optimal number of slices that can be made in this setting.

2. Oracle Queries: Another essential parameter that affects the efficiency of a garbling scheme is the oracle queries made by the garbler and the evaluator. In sliced-garbling schemes, the garbler has to use some combination of these queries to mask the slices of the output label. For determining this combination, it is imperative to consider the following two constraints:

(a) Each slice needs to have a distinct combination of oracle queries masking it to ensure that the slice is derived independently of the others.

(b) For a given slice, the chosen combination of oracle queries must yield distinct masking values for every pair of inputs. This is to avoid situations where having access to a particular masking value reveals the corresponding slice for multiple inputs.

Keeping the above constraints in mind, our scheme uses the following combinations:

$$\begin{aligned} D_l^1 &\hookleftarrow H(A_i) \oplus H(B_j) \oplus H(A_i \oplus B_j \oplus C_k) \\ D_l^2 &\hookleftarrow H(B_j) \oplus H(C_k) \oplus H(A_i \oplus B_j \oplus C_k) \\ D_l^3 &\hookleftarrow H(A_i) \oplus H(C_k) \oplus H(A_i \oplus B_j \oplus C_k) \end{aligned}$$

where A_i, B_j, and C_k are the input labels and D_l is the output label, representing the inputs (i, j, k) and the output l respectively.

Notice that the above combination already sets the values of the vector $\vec{H}$ and the matrices M and V, in Equation (13). The vector $\vec{H}$ should contain all possible queries that can be made to the oracle for the scheme. Hence,

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

The matrix M, as seen in three-halves, represents which combination of queries from $\vec{H}$ are chosen for inputs (i, j, k) to derive the corresponding output slice. The above combination, hence, completely determines M. Also recall from the three-halves scheme in [34] that setting M also sets V, as both of them share the same column space and $\mathsf{Dim}(V) = \mathsf{Rank}(M)$. Therefore, we can choose any V that satisfies both these conditions. We list the matrices V and M that we use for this scheme in Appendix A.

3. The gadget being garbled: Recall that in the three-halves garbling scheme [34] when encrypting a 1-slice, the corresponding slice of $\varDelta$ is included in the equation, while the 0-slice does not include $\varDelta$. The decision of whether to include $\varDelta$ depends on the specific gadget being used and the chosen permute bits for a given row. These two factors collectively determine the values in the matrix T. The main purpose of matrix T is to determine whether a slice of $\varDelta$ should be added as a term in a particular equation within the system. We now will see how to compute the matrix T in our case. Recall that T is a matrix determined by the output column of the permuted truth table. Let p_a, p_b and p_c be the three chosen permute bits for wires a, b and c, of the 3-input gadget $\mathcal{G}_{\text{tri}}$. The output of the gadget, for the given permute bits, is

$$T_{\mathsf{out}}(a, b, c) = (a \oplus p_a) \wedge ((b \oplus p_b) \oplus (c \oplus p_c))$$

The matrix T can be computed in the following way, given a set of permute bits for each input wire:

$$T = \begin{bmatrix} T_{0,0,0} & T_{0,0,1} & T_{0,1,0} & T_{0,1,1} & T_{1,0,0} & T_{1,0,1} & T_{1,1,0} & T_{1,1,1} \end{bmatrix}^{\intercal} \tag{14}$$

where we have that $\forall (i, j, k) \in \{0, 1\}$, the sub-matrix $T_{i,j,k} = I_3$ if $T_{\mathsf{out}}(i, j, k) = 1$; $0_{3\times3}$ otherwise. For example, if $p_a = p_b = p_c = 0$, then T would be the following matrix for $\mathcal{G}_{\text{tri}}$:

$$T = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_3 & I_3 & 0_{3\times3} \end{bmatrix}^{\intercal}$$

At this point, we know how to compute the matrices V, M, and T and sample vectors $\vec{A_0}, \vec{B_0}, \vec{C_0}$ and $\vec{\varDelta}$. It remains is to show how to choose R such that Equation (13) represents a garbling scheme for $\mathcal{G}_{\text{tri}}$.

Choosing Control Matrix *R*: We will now discuss choosing the matrix R to garble the gadget $\mathcal{G}_{\text{tri}}$. Similar to the three-halves garbling scheme, we observe that we do not have a single control matrix R that satisfies the equation for all values of T. So, we will have to reuse their dicing technique, where we find a matrix R that works, given T, and encrypt it before sending it to the evaluator. We split this discussion into two parts. We first focus on the constraints for choosing such a matrix R. We then show how to choose it, given T, to satisfy the constraints. This approach is similar to that discussed in [34] for the three-halves garbling scheme. In our case, however, we cannot do a brute-force search for the matrices that satisfy these constraints, like in three-halves. So, we also show a novel algorithm to optimize the time complexity to search for such a matrix R. In the following, we will discuss the constraints determining the size and structure of R.

1. Size of R: It's important to note that in Equation (13), each of the vectors $\vec{A_0}, \vec{B_0}, \vec{C_0}, \vec{\varDelta}$ has dimension 3, meaning that R operates on a vector of size 12. The truth table for the 3-input gadget $\mathcal{G}_{\text{tri}}$ consists of eight rows, and since each of the three slices needs to be encrypted separately, each row introduces three equations. Hence, the system involves 24 equations, and R is a size 24×12 matrix.

2. Structure of R: In addition to considering the size of matrix R, there is another important constraint to consider. The matrix R operates on the slices of the input labels. This means that R determines whether a particular slice of a wire label is included in an equation. Remember that each equation in the system describes an encryption of a particular output slice, corresponding to some inputs (i, j, k). The triplet of active labels associated with these inputs are (A_i, B_j, C_k). Therefore, it cannot be that R chooses to add an inactive label in any equation corresponding to these inputs (i, j, k), as the evaluator would not be able to access it. So, we must ensure that R only uses combinations of slices that the evaluator can reproduce for a given input. Let us try to use this intuition to induce an explicit structure on R, similar to how it was done for the three-halves garbling scheme in [34].

For the sake of illustration, let us start with a garbling scheme without slicing for 2-input gates. Recall that we have derived the following equation for this setting when we discussed the half-gates scheme.

$$V \begin{bmatrix} C \\ G_1 \\ G_2 \end{bmatrix} = M \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix} \oplus R \begin{bmatrix} A_0 \\ B_0 \\ \varDelta \end{bmatrix} \oplus T\varDelta \tag{15}$$

Here, the matrix R is of size 4×3. Let us explicitly rewrite R in terms of its components in the following way:

$$R \begin{bmatrix} A_0 \\ B_0 \\ \varDelta \end{bmatrix} = \begin{bmatrix} R_{0,0,A_0} & R_{0,0,B_0} & R_{0,0,\varDelta} \\ R_{0,1,A_0} & R_{0,1,B_0} & R_{0,1,\varDelta} \\ R_{1,0,A_0} & R_{1,0,B_0} & R_{1,0,\varDelta} \\ R_{1,1,A_0} & R_{1,1,B_0} & R_{1,1,\varDelta} \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ \varDelta \end{bmatrix} \tag{16}$$

where $R_{i,j,X} \in \{0, 1\}$ denotes the part of R corresponding to inputs (i, j) operating on X. We now show that the column corresponding to $\varDelta$ in the matrix R is deducible from other columns of R. For example, consider the inputs $(0, 1)$. For these inputs, $\{A_0, B_1 = B_0 \oplus \varDelta\}$ is the set of active labels. To include the label A_0 to the equation corresponding to these inputs, we set $R_{0,1,A_0} = 1$. If we do not wish to include the label A_0, then we set $R_{0,1,A_0} = 0$. Note that including or excluding A_0 does not affect the value of $R_{0,1,\varDelta}$ as it corresponds to $\varDelta$, and the label A_0 does not include $\varDelta$ term. However, this is not the case for label B_1. If we want to include B_1, then we will have to set both $R_{0,1,B_0} = 1$ and $R_{0,1,\varDelta} = 1$. If we choose to exclude B_1, then we set $R_{0,1,B_0} = 0, R_{0,1,\varDelta} = 0$. So, in both cases, we have that $R_{0,1,\varDelta} = R_{0,1,B_0}$. On deriving this for equations corresponding to all inputs, one can realise the structure we sought for matrix R in Equation (16) as follows:

$$R \begin{bmatrix} A_0 \\ B_0 \\ \varDelta \end{bmatrix} = \begin{bmatrix} R_{0,0,A_0} & R_{0,0,B_0} & 0 \\ R_{0,1,A_0} & R_{0,1,B_0} & R_{0,1,B_0} \\ R_{1,0,A_0} & R_{1,0,B_0} & R_{1,0,A_0} \\ R_{1,1,A_0} & R_{1,1,B_0} & R_{1,1,A_0} \oplus R_{1,1,B_0} \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ \varDelta \end{bmatrix} \tag{17}$$

The XOR term in the last row of R is because both A_1 and B_1 contribute a $\varDelta$ term to the equation. Note that this logic can now be extended to sliced garbling schemes by imposing the structure for every slice. This involves viewing the term $R_{i,j,X}$ acting on single element X as sub-matrices $R_{i,j,X}$ acting on a vector $\vec{X}$. Here, each row of $R_{i,j,X}$ corresponds to a slice of the output label. So, for a scheme like three-halves, the structure of R would look as follows:

$$R \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{\varDelta} \end{bmatrix} = \begin{bmatrix} R_{0,0,A_0} & R_{0,0,B_0} & 0 \\ R_{0,1,A_0} & R_{0,1,B_0} & R_{0,1,B_0} \\ R_{1,0,A_0} & R_{1,0,B_0} & R_{1,0,A_0} \\ R_{1,1,A_0} & R_{1,1,B_0} & R_{1,1,A_0} \oplus R_{1,1,B_0} \end{bmatrix} \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{\varDelta} \end{bmatrix} \tag{18}$$

where $R_{i,j,X}$ is a sub-matrix corresponding to inputs (i, j) acting on vector $\vec{X}$.

Having established the structure for R for 2-input gadgets in a sliced garbling scheme, it is straightforward to extend it to the 3-input case by just accounting for additional input. The following is the structure of R

in the 3-input case:

$$R = \begin{bmatrix} R_{0,0,0,A_0} & R_{0,0,0,B_0} & R_{0,0,0,C_0} & 0 \\ R_{0,0,1,A_0} & R_{0,0,1,B_0} & R_{0,0,1,C_0} & R_{0,0,1,C_0} \\ R_{0,1,0,A_0} & R_{0,1,0,B_0} & R_{0,1,0,C_0} & R_{0,1,0,B_0} \\ R_{0,1,1,A_0} & R_{0,1,1,B_0} & R_{0,1,1,C_0} & R_{0,1,1,B_0} \oplus R_{0,1,1,C_0} \\ R_{1,0,0,A_0} & R_{1,0,0,B_0} & R_{1,0,0,C_0} & R_{1,0,0,A_0} \\ R_{1,0,1,A_0} & R_{1,0,1,B_0} & R_{1,0,1,C_0} & R_{1,0,1,A_0} \oplus R_{1,0,1,C_0} \\ R_{1,1,0,A_0} & R_{1,1,0,B_0} & R_{1,1,0,C_0} & R_{1,1,0,A_0} \oplus R_{1,1,0,B_0} \\ R_{1,1,1,A_0} & R_{1,1,1,B_0} & R_{1,1,1,C_0} & R_{1,1,1,A_0} \oplus R_{1,1,1,B_0} \oplus R_{1,1,1,C_0} \end{bmatrix} \tag{19}$$

where the subscript (i, j, k, X) for R denotes a sub-matrix of R corresponding to inputs (i, j, k) in the truth table, operating on the vector $\vec{X}$. For our scheme, as we have 3 slices for each wire label, each $R_{i,j,k,X}$ is a 3×3 matrix.

Lastly, there is one more constraint on matrix R that arises from the system represented in Equation (13). Consider a matrix K as a basis for the co-kernel of matrix M. If we multiply K on both sides of Equation (13), the term $KM = 0$, by the definition of co-kernel and similarly, the term $KV = 0$ as V has the same column space of M. So we have,

$$0 = K \left(R \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix} \oplus T\vec{\Delta} \right) \tag{20}$$

Consider a matrix $T' = \begin{bmatrix} 0_{24\times3} & 0_{24\times3} & 0_{24\times3} & T \end{bmatrix}$. Then, we can rewrite Equation (20) as follows:

$$0 = K \left(R \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix} \oplus T' \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix} \right) = K(R \oplus T') \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix}$$

Notice that T operating on $\vec{\Delta}$ is the same as T' operating on $\begin{bmatrix} \vec{A}_0 & \vec{B}_0 & \vec{C}_0 & \vec{\Delta} \end{bmatrix}^\intercal$. It is because T' is a matrix with three zero-matrices appended to the left of T. For this reason, from hereon, we stick to T', which T completely determines. Then, from the above equation, we want R such that:

$$K(R \oplus T') = 0 \implies KR = KT' \tag{21}$$

Sampling a control matrix *R*: As we have now established we want to find a control matrix R of size 24×12, whose structure resembles that shown in Equation (19) and such that it satisfies Equation (21), for a given T'. We now show that finding the set of matrices R that satisfy Equation (21), given T', is equivalent to finding a general solution to a non-homogeneous system of linear equations. Firstly, we observe that $KT' \neq 0$ for any given T' by computing KT' for all possible T'. As KT' is a fixed non-zero matrix, the system is a non-homogeneous system of linear equations. The list of possible R matrices would hence be the general solution to this system.

To find the set of possible control matrices R for a given T' that satisfies the equation $KR = KT'$, we can employ the standard two-step approach based on solving a non-homogeneous system of equations $AX = B$. Firstly, we find the general solution to the homogeneous system of equations $AX = 0$. This provides a list of solutions that satisfy the equation without the affine term. Secondly, we find a specific solution to the non-homogeneous system $AX = B$. By adding this specific solution to each solution in the list of the general solution to the homogeneous system, we obtain the general solution to the non-homogeneous system. This process generates an exhaustive list of possible control matrices R. The garbler can then select one matrix from this list.

General Solution for the homogeneous system KR = 0: To obtain the general solution for the homogeneous system $KR = 0$, let's revisit our previous discussion. Since K represents a basis matrix for the co-kernel of M, the general solution for this system would be all possible matrices R that share the same column space as M.

However, we encounter a slight complication. We desire a matrix R that satisfies the structure described in Equation (19), but not all matrices sharing the same column space as M satisfy this condition. Let us denote by $R_{\$}$ a matrix that satisfies $KR_{\$} = 0$, and adheres to the structure described in Equation (19). By determining all possible matrices $R_{\$}$, we can find a comprehensive set of matrices that fulfill the given requirements.

Let the general solution, the list of all possible solutions, be denoted as a set of matrices, $\mathcal{S}_{R_{\$}}$. We will now show how to find all these solutions. We know that any $R_{\$}$ should follow the exact structure of R as shown in Equation (19) and is a 24×12 matrix. Let $\mathsf{C}_1, \cdots, \mathsf{C}_{12}$ be its 12 columns. Then, because of this structure, we see that for a given input of the gadget, the values in column C_{10} are determined by the values of $\mathsf{C}_1, \mathsf{C}_4$ and C_7. This is because C_{10} is the column operating on the value Δ^1 in Equation (13). Δ_1 is the first slice of Δ, and its inclusion only depends on the inclusion of the first slice of each input label(s). The columns corresponding to these slices are $\mathsf{C}_1, \mathsf{C}_4$, and C_7. Similarly, C_{11} depends only on $\mathsf{C}_2, \mathsf{C}_5, \mathsf{C}_8$; and C_{12} depends only on $\mathsf{C}_3, \mathsf{C}_6, \mathsf{C}_9$, respectively. Moreover, we observe that this dependency is symmetric. So it is enough to find a span for the matrix $R_{\$,1} = [\mathsf{C}_1, \mathsf{C}_4, \mathsf{C}_7, \mathsf{C}_{10}]$ and reuse it for the other two column sets to construct the entire matrix $R_{\$}$.

Furthermore, since we aim for $KR_{\$} = 0$, the columns of $R_{\$}$ can be derived using a span of the basis of the column space of M. We first compute this basis, which has dimension 7 (i.e., same as that of M). We then consider all possible 2^7 linear combinations of this basis for each column C_1, C_4, and C_7 (noting that C_{10} is fixed). These combinations should form a valid matrix $[\mathsf{C}_1, \mathsf{C}_4, \mathsf{C}_7, \mathsf{C}_{10}]$, adhering to the requisite structure of $R_{\$}$ and satisfying $KR_{\$,1} = 0$. Each resulting matrix is added to the set $\mathcal{S}_{R_{\$}}$.

Following this process, we obtain the complete set $\mathcal{S}_{R_{\$}}$ of dimension 2^8. To generate the entire matrix $R_{\$}$, we randomly select three matrices from $\mathcal{S}_{R_{\$}}$. These matrices are as follows:

$$\begin{aligned} R_{\$,1} &= [\mathsf{C}_1, \mathsf{C}_4, \mathsf{C}_7, \mathsf{C}_{10}] \\ R_{\$,2} &= [\mathsf{C}_2, \mathsf{C}_5, \mathsf{C}_8, \mathsf{C}_{11}] \\ R_{\$,3} &= [\mathsf{C}_3, \mathsf{C}_6, \mathsf{C}_9, \mathsf{C}_{12}]. \end{aligned}$$

Hence, with augmenting and rearranging some columns, we have the matrix

$$R_{\$} = [\mathsf{C}_1, \cdots, \mathsf{C}_{12}]$$

Finding a Specific Solution for the homogeneous system KR = KT': We will now discuss the approach to search for a matrix R that fulfills Equation (21). To achieve this, we followed a similar process, where we sought an $R_{\$}$ matrix satisfying the equation $KR_{\$} = 0$. However, now, our objective is to find a single matrix R that satisfies the equation $KR = KT'$. It is important to note that in this case, instead of identifying a comprehensive set of matrices, we aim to find just one matrix that meets the requirements. After conducting the search, we have determined that such matrices do indeed exist for this particular gadget for any value of T'.

After finding a general solution for the equation $KR_{\$} = 0$ and a specific solution for $KR = KT'$ using the method described above, it is possible to find a general solution for $KR = KT'$ for some T'. Let $\mathcal{S}_{R_{\$}}$ be the list of all possible solutions to the equation $KR_{\$} = 0$, and let $R_{T'}$ be a matrix that satisfies $KR_{T'} = KT'$. The list of all possible solutions for $KR = KT'$ can be expressed as:

$$\mathcal{S}_{T'} = \{R_{\$} \oplus R_{T'} | R_{\$} \in \mathcal{S}_{R_{\$}}\}.$$

Encrypting the Control Matrix $\boldsymbol{R}$ Any control matrix R sampled as described above contains information about T, and hence the permute bits. Therefore, we need to encrypt it before sending it to the evaluator so that it can only decrypt the rows of R corresponding to the active inputs (x, y, z). We already noticed that the garbling equation implements such an encryption mechanism. Namely, Each row in the garbling equation encrypts a slice of an output label D_i corresponding to the active inputs (x, y, z) by masking them with hash outputs. We can use these to column-wise encrypt R as well. It is important to note that only the first nine columns of R are needed to be communicated, as the last three columns can be derived from them. Furthermore, for the first nine columns $\mathsf{C}_1, \cdots, \mathsf{C}_9$, it holds that $K\mathsf{C}_i = 0$, indicating that they are within the column space of M. The column space of M has dimension 7. As we have already seen for our garbling scheme, this means that there is a 7-bit vector $\vec{r_i}$ such that

$$V\vec{r_i} = M\vec{H} \oplus \mathsf{C}_i$$

where $\vec{H}$ denotes just the additional one bit of the hashes used to encrypt each C_i for all $i \in \{1, \cdots, 9\}$. We include the vector $\vec{r} = [\vec{r_1}, \cdots, \vec{r_9}]$ as a constant size ciphertext in addition to the ciphertext for the garbling equation $\vec{G}$. We also now modify the hash function H to map κ bits strings to $\frac{\kappa}{3} + 9$ bit strings instead of them mapping to $\frac{\kappa}{3}$ bit strings.

On the potential compression of R: We now describe a potential optimization to reduce the size of the ciphertext for the control matrix R using the same technique as described by Rosulek and Roy in [34]. We currently use the set $S = \{\mathsf{C}_1, \cdots, \mathsf{C}_9\}$ to represent R, and encrypt each component of S individually. If one can find a smaller span for S, say $S' = \{\mathsf{S}_1, \cdots \mathsf{S}_d\}$ where $d < 9$, then we could instead encrypt S', which would cost $7d$ bits. In [34], Rosulek and Roy showed that there exists S' for their case, with $d = 1$, which reduced the constant ciphertext size from 20 bits to 5 bits. If such a d exists for our case, then the constant ciphertext in our case reduces optimistically to 7 bits, bringing the size of the garbled gate for each gadget to $4\frac{\kappa}{3} + 7$ bits. We believe that such a span exists, and leave it for future work.

Decoding information: To authenticate output labels, and determine if they are a 0-label or 1-label, the garbler provides the hashes of these labels as decoding information. The hash H used to garble is a function mapping κ bit strings to $\frac{\kappa}{3}$ bit strings, and would not be sufficient to provide κ bits of authenticity. Therefore, we use another hash function H' with output length κ bits. A possible instantiation for H' is:

$$H'(E, k) = [H(E, t(k))]_{\frac{\kappa}{3}} \parallel [H(E, t(k) + 1)]_{\frac{\kappa}{3}} \parallel [H(E, t(k) + 2)]_{\frac{\kappa}{3}}$$

where $t(k)$ is some appropriate tweak that does not collide with any other hash call in the scheme, and the notation $[X]_{\frac{\kappa}{3}}$ refers to the first $\frac{\kappa}{3}$ bits of the string X.

4.2 Garbling a Circuit with XOR Gates and the $\mathcal{G}_{\mathbf{tri}}$ Gadget

We formalise the techniques mentioned in Section 4.1 in Figure 1, to describe a garbling scheme for a circuit comprised of XOR gates and $\mathcal{G}_{\text{tri}}$ gadgets.

```
procedure Init($\kappa$):
    $H := \mathcal{H}(\kappa)$;
procedure Garble($1^\kappa, f$):
    $\Delta :=_\$ \{0,1\}^{\kappa-1} || 1$;
    for $i \in$ Inputs($f$) do
        $W_i^0 :=_\$ \{0,1\}^\kappa$;
        $W_i^1 := W_i^0 \oplus \Delta$;
        $e_i := W_i^0$;
    for $i \notin$ Inputs($f$){in topo.order} do
        if $i \in$ XORGates($f$) then
            $\{a,b\} :=$ GateInputs($f, i$);
            $W_i^0 := W_a^0 \oplus W_b^0$;
            $W_i^1 := W_i^0 \oplus \Delta$;
            continue;
        $\{a,b,c\} :=$ GateInputs($f, i$);
        $(W_i^0, \vec{G}_i, \vec{z}_i) :=$ Gb3GADGET($W_a^0, W_b^0, W_c^0$);
        $W_i^1 := W_i^0 \oplus \Delta$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
    for $i \in$ Outputs($f$), $j \in \{0,1\}$ do
        $D_i^j := H'(W_i^0 \oplus \mathrm{lsb}(W_i)\Delta, i)$;
    return $(\hat{F}, \hat{e}, \hat{D})$;
procedure Gb3GADGET($W_a^0, W_b^0, W_c^0$):
    $p_a, p_b, p_c := \mathrm{lsb}(W_a^0), \mathrm{lsb}(W_b^0), \mathrm{lsb}(W_c^0)$;
    $T' :=$ 3GADGET($p_a, p_b, p_c$);
    $R \leftarrow_\$ S_T$, $\vec{r} := R_{[1:9]}$;
    $\vec{z} \parallel \begin{bmatrix} W_d^0 \\ \vec{G} \end{bmatrix} := V^{-1}(\vec{r} \parallel (R \oplus T') \begin{bmatrix} W_a^0 \\ W_b^0 \\ W_c^0 \\ \Delta \end{bmatrix} \oplus M\vec{H})$;
    $W := W_d^0$;
    return $(W, \vec{G}, \vec{z})$

procedure Encode($\hat{e}, \hat{x}$):
    for $e_i \in \hat{e}$ do
        $X_i := e_i \oplus x_i \Delta$;
    return $\hat{X}$;
procedure Decode($\hat{D}, \hat{Y}$):
    for $D_i \in \hat{D}$ do
        if $\exists j \in \{0,1\}$ s.t. $D_i^j = H'(Y_i, i)$ : then
            $y_i := j \oplus \mathrm{lsb}(Y_i)$;
    return $\hat{Y}$;
procedure Eval($\hat{F}, \hat{X}$):
    for $i \in$ Inputs($\hat{F}$) do
        $W_i := X_i$;
    for $i \notin$ Inputs($\hat{F}$){in topo.order} do
        if $i \in$ XORGates($\hat{F}$) then
            $\{a,b\} :=$ GateInputs($\hat{F}, i$);
            $W_i := W_a \oplus W_b$;
            $E_i := W_i$;
            continue;
        $\{a,b,c\} :=$ GateInputs($\hat{F}, i$);
        $E_i :=$ Eval3GADGET($F_i, W_a, W_b, W_c$);
    Return $\hat{E}$;
procedure Eval3GADGET($F = (\vec{G}, \vec{z}), W_a, W_b, W_c$):
    $s_a, s_b, s_c := \mathrm{lsb}(W_a), \mathrm{lsb}(W_b), \mathrm{lsb}(W_c)$;
    $\vec{r}_{s_a,s_b,s_c} \parallel X_{s_a,s_b,s_c} := V_{s_a,s_b,s_c}\left(\vec{z} || \begin{bmatrix} 0 \\ \vec{G} \end{bmatrix}\right) \oplus M_e \vec{H}_e$;
    $E := X_{s_a,s_b,s_c} \oplus \vec{r}_{s_a,s_b,s_c} \begin{bmatrix} \vec{W}_a \\ \vec{W}_b \\ \vec{W}_c \end{bmatrix}$;
    Return $E$;
```

Figure 1: Garbling scheme

We next prove the following theorem.

Theorem 1 *Assuming the 3-TCCR assumption (Definition 3), the garbling scheme described in Figure 1 meets Definition 2 for any circuit comprised of 2-input XOR gates and $\mathcal{G}_{\mathrm{tri}}$ gadgets, where each $\mathcal{G}_{\mathrm{tri}}$ gadget has a garbled gate of size $4\frac{\kappa}{3} + \mathcal{O}(1)$ bits and XOR gates are free, where κ is the security parameter.*

Proof: We retain correctness from the three-halves garbling scheme, as we logically extend the algebraic equation to be able to garble any 3-input $\mathcal{G}_{\mathrm{tri}}$ gadget. For privacy, we have to simulate the tuple $(\vec{G}, \vec{X}, \vec{D})$ for all relevant gates, such that they are indistinguishable from the real execution of the garbling scheme. We have standard encoding and decoding procedures, and the 3-TCCR oracle in Definition 3 lets us extract the oracle response for the inactive labels to be able to simulate the garbled gate for each GTri gadget.

More precisely, we define a simulator that does the following in a topological order: (1) the simulator samples the active labels for the input wires uniformly at random; (2) for each XOR gate, it sets the active output label to be the XOR of the active input labels, and (3) for the $\mathcal{G}_{\mathrm{tri}}$ gadgets, it samples at random, the corresponding ciphertext and control bits, and uses the evaluation algorithm to get the active output label. We then show through a series of hybrids, starting from the real execution to the above described simulation, that the views of these hybrids are indistinguishable. We state all the hybrids and the indistinguishability arguments in the full proof below. As stated in Definition 2, we need to prove four properties the garbling scheme needs to satisfy.

Correctness: We show that for every XOR gate or $\mathcal{G}_{\mathrm{tri}}$ gadget, the correct output label is extracted. The correctness for XOR gates follows from the free-XOR optimization. What remains to be shown is the correctness of the $\mathcal{G}_{\mathrm{tri}}$ gadget. Let (x_a, x_b, x_c) be the inputs on the wires (a, b, c), (p_a, p_b, p_c) be the permute bits for those wires for a $\mathcal{G}_{\text{tri}}$ gadget, and $W_a^{x_a}, W_b^{x_b}$ and $W_c^{x_c}$ be the active input labels. The evaluator computes Eval3Gadget with these wire labels, and the garbled gate $\{\vec{G}, \vec{z}\}$ as its input, as described in Figure 1. The output label it receives is

$$E_k := X_{s_a,s_b,s_c} \oplus R_{s_a,s_b,s_c} \begin{bmatrix} \vec{W}_a \\ \vec{W}_b \\ \vec{W}_c \end{bmatrix}$$

where $s_a = p_a \oplus x_a, s_b = p_b \oplus x_b, s_c = p_c \oplus x_c$ are the select bits for this gadget the evaluator obtained. Also, X_{s_a,s_b,s_c} denotes the three-rows of

$$\vec{X} = \vec{W_d^0} \oplus R \begin{bmatrix} \vec{W_a^0} \\ \vec{W_b^0} \\ \vec{W_c^0} \\ \vec{\Delta} \end{bmatrix} \oplus T\vec{\Delta}$$

that corresponds to the case where inputs are x_a, x_b, x_c. The structure of R in Equation (19) ensures that

$$R \begin{bmatrix} \vec{W_a^0} \\ \vec{W_b^0} \\ \vec{W_c^0} \\ \vec{\Delta} \end{bmatrix} = R_{s_a,s_b,s_c} \begin{bmatrix} \vec{W}_a \\ \vec{W}_b \\ \vec{W}_c \\ \vec{\Delta} \end{bmatrix}$$

for the rows corresponding to the inputs x_a, x_b, x_c. Therefore, from the above equation and the definition of the encode algorithm, we have

$$\begin{aligned} E_k &= X_{s_a,s_b,s_c} \oplus R_{s_a,s_b,s_c} \begin{bmatrix} \vec{W}_a \\ \vec{W}_b \\ \vec{W}_c \end{bmatrix} \\ &= \vec{W_d^0} \oplus T_{s_a,s_b,s_c}\vec{\Delta} \\ &= W_d^0 \oplus (\mathcal{G}_{\text{tri}}(x_a \oplus p_a, x_b \oplus p_b, x_c \oplus p_c) \oplus p_d)\Delta. \end{aligned}$$

which by definition is the output label corresponding to the inputs x_a, x_b, x_c. Hence, it is true that the evaluator always derives the active output label. Having derived all the output labels, the decoding information ensures that $H'(W_k \oplus (x_k \oplus p_k)\Delta, k) = H'(E_k, k)$, given that $D_k^{x_k} \neq D_k^{1\oplus x_k}$. This ensures the evaluator always decodes the output label to the correct value except with negligible probability. Therefore, correctness holds.

Privacy: For proof of privacy, we need to write a simulator Sim that can generate an indistinguishable view of the following output of the Garble, Encode algorithms:

$$\begin{aligned} (\hat{F}_s, \hat{X}_s, \hat{D}_s) &\leftarrow \mathsf{Sim}(1^\kappa, f, \hat{y} = f(\hat{x})) \\ (\hat{F}_r, \hat{X}_r, \hat{D}_r) &\leftarrow \mathsf{Garble}(1^\kappa, f), \mathsf{Encode}(\hat{e}, \hat{x}) \end{aligned}$$

We have the following simulator algorithm:

By the definition of privacy for garbling schemes as in Definition 2, we need the above outputs of Sim and Garble, Encode to be indistinguishable for any circuit f and input $\hat{x}$. Let us fix some $f, \hat{x}$. We now describe the following sequence of hybrids that will help us show indistinguishability between Sim and Garble, Encode views as described above, on these fixed $f, \hat{x}$:

```
procedure $\mathsf{Sim}(1^\kappa, f, \hat{y}) = \mathsf{Hybrid}_6(1^\kappa, f, \hat{y})$:
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i \leftarrow_\$ \{0,1\}^\kappa$;
        $X_i := W_i$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        if $i \in \mathrm{XORGates}(f)$ then
            $\{a, b\} := \mathsf{GateInputs}(f, i)$;
            $W_i := W_a \oplus W_b$;
            $X_i := W_i$;
            continue;
        $\{a, b, c\} := \mathsf{GateInputs}(f, i)$;
        $G_i \parallel z_i \leftarrow_\$ \{0,1\}^{\frac{4\kappa}{3}+63}$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
        $W_i := \mathsf{Eval3GADGET}(F_i, W_a, W_b, W_c)$;
        $X_i := W_i$;
    for $i \in \mathsf{Outputs}(f), j \in \{0,1\}$ do
        $D_i^{y_i} := H'(W_i, i)$;
        $D_i^{1-y_i} \leftarrow_\$ \{0,1\}^\kappa$;
    return $(\hat{F}, \hat{X}, \hat{D})$;
```
Figure 2: Simulator

Hybrid 1: We switch from the garbler's perspective to the evaluator's perspective when we garble. That is, instead of tracking wire labels for wire A as $(W_a^0, W_a^0 \oplus \varDelta)$, we track them as $(W_a^{x_a}, W_a^{x_a} \oplus \varDelta)$. Notice that W_a^0 here can be either $W_a^{x_a}$ or $W_a^{x_a} \oplus \varDelta$. We also change it this way for all other wire labels. These changes also introduce other variable name changes throughout the algorithm. Another important change is that the algorithm Hybrid_1 takes as input $\hat{x}$, which corresponds to the value the active labels represent. Following is a list of all changes:

1. Replace W_a^0 with the active label $W_a^{x_a}$, and the inactive label as $W_a^{x_a} \oplus \varDelta$ for wires A, B, C, D
2. We know that the control matrix R has been written keeping in mind the zero labels. So we do a basis change for R, and replace the term

$$R \times \begin{bmatrix} W_a^0 \\ W_b^0 \\ W_c^0 \\ \varDelta \end{bmatrix} \rightarrow R' \times \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \\ \varDelta \end{bmatrix}$$

3. We have $\vec{H}$ also have these A_0 terms. For this, we split the matrix into two halves, one corresponding to the active labels and the other to the inactive labels:

$$\vec{H}_0 := \begin{bmatrix} H(W_a^{x_a}) \\ H(W_b^{x_b}) \\ H(W_c^{x_c}) \\ H(W_a^{x_a} \oplus W_b^{x_b} \oplus W_c^{x_c}) \end{bmatrix}, \vec{H}_\varDelta := \begin{bmatrix} H(W_a^{x_a} \oplus \varDelta) \\ H(W_b^{x_b} \oplus \varDelta) \\ H(W_c^{x_c} \oplus \varDelta) \\ H(W_a^{x_a} \oplus W_b^{x_b} \oplus W_c^{x_c} \oplus \varDelta) \end{bmatrix}$$

   Then, we will have to also split up M and rewrite the term $M\vec{H}$ as $M_0\vec{H}_0 \oplus M_\varDelta\vec{H}_\varDelta$.

4. The matrix V^{-1} has seven rows, where the first three rows correspond to slices of the output label and the last 4 to the corresponding ciphertexts (We stress that this is easy to check keeping in mind that the first three columns of V corresponding to the output label slices and the last four columns correspond

to the ciphertexts). We divide V^{-1} based on this, denoting them as $V^{-1}_{\text{label}}, V^{-1}_{\text{gate}}$. Instead of multiplying by V^{-1}, we just multiply with V^{-1}_{gate} to solve for the gate-ciphertexts only. We then evaluate those gate ciphertexts with A and B to learn the active output label E_c. This approach still has the same result using the fact that the scheme is correct. We also partition the control bit ciphertexts $\vec{z}$ to $\vec{z} = [(\vec{z})_{\text{top}}, (\vec{z})_{\text{bot}}]$, use V^{-1}_{gate} to compute $(\vec{z})_{\text{bot}}$, and then use the evaluator's computation to solve for $(\vec{z})_{\text{top}}$.

5. We now focus on the term
$$V^{-1}_{\text{gate}} M \vec{H} = V^{-1}_{\text{gate}}(M_0 \vec{H}_0 \oplus M_\Delta \vec{H}_\Delta)$$
We see that
$$V^{-1}M = \left[\begin{array}{cc|cc|cc|cc} 1&0&1&0&0&0&1&0 \\ 0&0&1&0&1&0&1&0 \\ 1&0&0&0&1&0&1&0 \\ 1&1&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&1&1 \end{array}\right]$$
Originally, in Garble, we know that odd columns of M corresponded to queries of $W_a^0, W_b^0, W_c^0, W_a^0 \oplus W_b^0 \oplus W_c^0$ and even columns of M corresponded to their respective Δ versions. As described in 3, M was split into M_0 and M_Δ. It is easy to see that M_Δ has exactly one of rows $\{1,2\}$,exactly one of rows $\{3,4\}$,exactly one of rows $\{5,6\}$ of M and exactly one of rows $\{7,8\}$. Irrespective of which of these rows it has, $V^{-1}_{\text{gate}} M_\Delta$(defined in 4) will always be the 4×4 identity matrix for any active label(irrespective of the input and permute bits). The same applies to $V^{-1}_{\text{gate}} M_0$ too. Therefore, we can write
$$V^{-1}_{\text{gate}} M \vec{H} = \vec{H}_0 \oplus \vec{H}_\Delta$$

6. Finally, for the sake of readability and convenience, we now rewrite the term:
$$\vec{z}_{\mathsf{bot}} \parallel \vec{G} := V^{-1}_{\text{gate}}\left(\left(\vec{r} \| (R' \oplus \left[0 \cdots | t\right]) \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \\ \Delta \end{bmatrix} \right) \oplus M_0 \vec{H}_0 \oplus M_\Delta \vec{H}_\Delta \right)$$
as the following:
$$\vec{z}_{\mathsf{bot}} \parallel \vec{G} := L_a(W_a^{x_a}) \oplus L_b(W_b^{x_b}) \oplus L_c(W_c^{x_c}) \oplus L_\Delta(\Delta) \oplus \vec{H}_0 \oplus \vec{H}_\Delta$$
where L_a, L_b, L_c, L_Δ are functions which denotes which linear combination of the slices of labels $W_a^{x_a}, W_b^{x_b}, W_c^{x_c}$ are to be used for garbling, something which is derived from the original equation, say through a procedure DeriveL.

Figure 3 formally describes Hybrid_1 to generate the changes mentioned above. We now can discuss indistinguishability between the joint distribution of the output of $\mathsf{Garble}(1^\kappa, f)$ and the output of $\mathsf{Hybrid}_1(1^\kappa, f, x)$. We argue the following for each change listed above:

– For 1, we see that it is a change in the variable name from W_a^0 to $W_a^{x_a}$ where x_a is now the label for the active value x_a. Hence, they are identically distributed.

– For 2, the change we do is a change of basis of R from $[A_0, B_0, C_0, \Delta]$ to $[A, B, C, \Delta]$ which results in R'. This does not affect the distribution by definition of basis change.

– For 3, 4, 5 and 6, we stress that it is again just a reordering and splitting of already existing matrices, and this does not affect the distribution of outputs too.

As these changes do not change any value in the outputs of Garble, and are just semantic changes in the way we write the equations, we have that the outputs of Hybrid_1 are identical to that of Garble.

```
procedure $\mathsf{Hybrid}_1(1^\kappa, f, \hat{x})$:
    $\Delta \leftarrow_\$ \{0,1\}^{\kappa-1}||1$;
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i^{x_i} \leftarrow_\$ \{0,1\}^\kappa$;
        $W_i^{1-x_i} := W_i^{x_i} \oplus \Delta$;
        $X_i := W_i^{x_i}$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        if $i \in \mathrm{XORGates}(f)$ then
            $\{a, b\} := \mathsf{GateInputs}(f, i)$;
            $W_i^{x_i} := W_a^{x_a} \oplus W_b^{x_b}$;
            $W_i^{1-x_i} := W_i^{x_i} \oplus \Delta$;
            $x_i := x_a \oplus x_b \oplus x_c$;
            $X_i := W_i^{x_i}$;
            continue;
        $\{a, b, c\} := \mathsf{GateInputs}(f, i)$;
        $x_i := x_a \wedge x_b \wedge x_c$;
        $(W_i^{x_i}, \vec{G}_i, \vec{z}_i) := \mathsf{Hyb3GADGET}_1(W_a^{x_a}, W_b^{x_b}, W_c^{x_c})$;
        $W_i^{1-x_i} := W_i^{x_i} \oplus \Delta$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
        $X_i := W_i^{x_i}$;
    for $i \in \mathsf{Outputs}(f)$ do
        $D_i^{x_i} := H'(W_i^{x_i}, i)$;
        $D_i^{1-x_i} := H'(W_i^{x_i} \oplus \Delta, i)$
    return $(\hat{F}, \hat{X}, \hat{D})$;
procedure $\mathsf{Hyb3GADGET}_1(W_a^{x_a}, W_b^{x_b}, W_c^{x_c})$:
    $s_a := \mathrm{lsb}(W_a^{x_a})$, $p_a := x_a \oplus s_a$;
    $s_b := \mathrm{lsb}(W_b^{x_b})$, $p_b := x_b \oplus s_b$;
    $s_c := \mathrm{lsb}(W_c^{x_c})$, $p_c := x_c \oplus s_c$;
    $T' \leftarrow \mathsf{3GADGET}(p_a, p_b, p_c)$, $R \leftarrow_\$ S_{T'}$;
    $\vec{L}_a, \vec{L}_b, \vec{L}_c, \vec{L}_\Delta, \vec{r} := \mathsf{DeriveL}(R, s_a, s_b, s_c)$;
    $\vec{z}_{\mathsf{bot}} \parallel \vec{G} := \vec{L}_a(W_a^{x_a}) \oplus \vec{L}_b(W_b^{x_b}) \oplus \vec{L}_c(W_c^{x_c}) \oplus \vec{L}_\Delta(\Delta) \oplus \vec{H}_0 \oplus \vec{H}_\Delta$;
    $W := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{G} \end{bmatrix} \oplus M_e \mathrm{msb}_{\frac{\kappa}{3}}(\vec{H}_0) \oplus R_{s_a,s_b,s_c} \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \end{bmatrix}$;
    $\vec{z}_{\mathsf{top}} := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{z}_{\mathsf{bot}} \end{bmatrix} \oplus M_e \mathrm{lsb}_{63}(\vec{H}_0) \oplus \vec{r}_{s_a,s_b,s_c}$;
    return $(W, \vec{G}, \vec{z})$
procedure $\mathsf{DeriveL}(R, s_a, s_b, s_c)$:
    $R' = \mathsf{ChangeBasis}(R, s_a, s_b, s_c)$, $\vec{r} := R'_{[1:9]}$;
    $\vec{L}_a, \vec{L}_b, \vec{L}_c, \vec{L}_\Delta := V_{\mathrm{gate}}^{-1} \left( \vec{r} || (R' \oplus T') \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \\ \Delta \end{bmatrix} \right)$;
    return $(\vec{L}_a, \vec{L}_b, \vec{L}_c, \vec{L}_\Delta, \vec{r})$;
```

Figure 3: Hybrid 1

Hybrid 2: We now do the following changes to Hybrid_1 to obtain Hybrid_2. We give Hybrid_2 access to an oracle $\mathcal{O}_\Delta^{3-\mathsf{tccr}}$ and remove all terms corresponding to Δ. For the labels, the Δ term was in the inactive wire labels, something to which the evaluator does not have access. For the terms having Δ that are inputs to the hash, we use the oracle $\mathcal{O}_{\Delta}^{\mathsf{tccr}}$. We formally describe the array $\vec{\mathcal{O}}$ as the array of values corresponding to oracle calls on $W_a^{x_a}, W_b^{x_b}, W_c^{x_c}$ respectively as needed.

We formally describe the hybrid 2 in Figure 4 to generate the changes mentioned above. The outputs of Hybrid_1 and $\mathsf{Hybrid}_2^{\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}}$ is identical, as all we did was replace each term of the form $H(X \oplus \Delta, i) \oplus b_1\Delta^1 \oplus b_2\Delta^2 \oplus b_3\Delta^3$ with the term $\mathcal{O}_{\Delta}^{\mathsf{tccr}}(X, i, \vec{L} = \{b_1, b_2, b_3\})$. They are the same by the definition of $\mathcal{O}_{\Delta}^{\mathsf{tccr}}$.

```
procedure $\mathsf{Hybrid}_2^{\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}}(1^\kappa, f, \hat{x})$:
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i^{x_i} \leftarrow_\$ \{0,1\}^\kappa$;
        $X_i := W_i^{x_i}$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        $\{a, b, c\} := \mathsf{GateInputs}(f, i)$;
        if $i \in \mathrm{XORGates}(f)$ then
            $W_i^{x_i} := W_a^{x_a} \oplus W_b^{x_b} \oplus W_c^{x_c}$;
            $x_i := x_a \oplus x_b \oplus x_c$;
            $X_i := W_i^{x_i}$;
            continue;
        $x_i := x_a \wedge x_b \wedge x_c$;
        $(W_i^{x_i}, \vec{G}_i, \vec{z}_i) := \mathsf{Hyb3GADGET}_2^{\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}}(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
        $X_i := W_i^{x_i}$;
    for $i \in \mathsf{Outputs}(f)$ do
        $D_i^{x_i} := H'(W_i^{x_i}, i)$;
        $D_i^{1-x_i} := \mathcal{O}(W_i^{x_i}, i)$
    return $(\hat{F}, \hat{X}, \hat{D})$;
procedure $\mathsf{Hyb3GADGET}_2^{\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}}(W_a^{x_a}, W_b^{x_b}, i)$:
    $s_a := \mathrm{lsb}(W_a^{x_a})$, $p_a := x_a \oplus s_a$;
    $s_b := \mathrm{lsb}(W_b^{x_b})$, $p_b := x_b \oplus s_b$;
    $s_c := \mathrm{lsb}(W_c^{x_c})$, $p_c := x_c \oplus s_c$;
    $T' \leftarrow \mathsf{3GADGET}(p_a, p_b, p_c)$, $R \leftarrow_\$ S_{T'}$;
    $\vec{L}_a, \vec{L}_b, \vec{L}_c, \vec{L}_\Delta, \vec{r} := \mathsf{DeriveL}(R, s_a, s_b, s_c)$;
    $\vec{z}_{\mathsf{bot}} \parallel \vec{G} := \vec{L}_a(W_a^{x_a}) \oplus \vec{L}_b(W_b^{x_b}) \oplus \vec{L}_c(W_c^{x_c}) \oplus \vec{H}_0 \oplus \vec{\mathcal{O}}(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$;
    $W := V_{s_a,s_b,s_c}\begin{bmatrix}0\\ \vec{G}\end{bmatrix} \oplus M_e \mathrm{msb}_{\frac{\kappa}{3}}(\vec{H}_0) \oplus R_{s_a,s_b,s_c}\begin{bmatrix}W_a^{x_a}\\ W_b^{x_b}\\ W_c^{x_c}\end{bmatrix}$;
    $\vec{z}_{\mathsf{top}} := V_{s_a,s_b,s_c}\begin{bmatrix}0\\ \vec{z}_{\mathsf{bot}}\end{bmatrix} \oplus M_e \mathrm{lsb}_{63}(\vec{H}_0) \oplus \vec{r}_{s_a,s_b,s_c}$;
    return $(W, \vec{G}, \vec{z})$
```

Figure 4: Hybrid 2

Hybrid 3: We now do the following changes to Hybrid_2 to obtain Hybrid_3. We give Hybrid_3 access to an oracle Rand instead of $\mathcal{O}_{\Delta}^{\mathsf{tccr}}$. We keep everything else the same. We formally describe the array $\vec{\mathcal{O}}$ as the array of values corresponding to oracle calls on $W_a^{x_a}, W_b^{x_b}, W_c^{x_c}$ respectively as needed. We formally describe the hybrid 3 in Figure 5 to generate the changes mentioned above.

To show the indistinguishability of these hybrids, we show a reduction to the definition of $3-\mathsf{TCCR}$ for $\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}$.

```
procedure $\mathsf{Hybrid}_3^{\mathsf{Rand}}(1^\kappa, f, \hat{x})$:
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i^{x_i} \leftarrow_\$ \{0,1\}^\kappa$;
        $X_i := W_i^{x_i}$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        $\{a, b, c\} := \mathsf{GateInputs}(f, i)$;
        if $i \in \mathrm{XORGates}(f)$ then
            $W_i^{x_i} := W_a^{x_a} \oplus W_b^{x_b} \oplus W_c^{x_c}$;
            $x_i := x_a \oplus x_b \oplus x_c$;
            $X_i := W_i^{x_i}$;
            continue;
        $x_i := x_a \wedge x_b \wedge x_c$;
        $(W_i^{x_i}, \vec{G}_i, \vec{z}_i) := \mathsf{Hyb3GADGET}_3^{\mathsf{Rand}}(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
        $X_i := W_i^{x_i}$;
    for $i \in \mathsf{Outputs}(f)$ do
        $D_i^{x_i} := H'(W_i^{x_i}, i)$;
        $D_i^{1-x_i} := \mathcal{O}(W_i^{x_i}, i)$
    return $(\hat{F}, \hat{X}, \hat{D})$;
procedure $\mathsf{Hyb3GADGET}_3^{\mathsf{Rand}}(W_a^{x_a}, W_b^{x_b}, i)$:
    $s_a := \mathrm{lsb}(W_a^{x_a})$, $p_a := x_a \oplus s_a$;
    $s_b := \mathrm{lsb}(W_b^{x_b})$, $p_b := x_b \oplus s_b$;
    $s_c := \mathrm{lsb}(W_c^{x_c})$, $p_c := x_c \oplus s_c$;
    $T' \leftarrow \mathsf{3GADGET}(p_a, p_b, p_c)$, $R \leftarrow_\$ S_{T'}$;
    $\vec{L}_a, \vec{L}_b, \vec{L}_c, \vec{L}_\Delta, \vec{r} := \mathsf{DeriveL}(R, s_a, s_b, s_c)$;
    $\vec{z}_{\mathsf{bot}} \parallel \vec{G} := \vec{L}_a(W_a^{x_a}) \oplus \vec{L}_b(W_b^{x_b}) \oplus \vec{L}_c(W_c^{x_c}) \oplus \vec{H}_0 \oplus \vec{\mathcal{O}}(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$;
    $W := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{G} \end{bmatrix} \oplus M_e \mathrm{msb}_{\frac{\kappa}{3}}(\vec{H}_0) \oplus R_{s_a,s_b,s_c} \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \end{bmatrix}$;
    $\vec{z}_{\mathsf{top}} := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{z}_{\mathsf{bot}} \end{bmatrix} \oplus M_e \mathrm{lsb}_{63}(\vec{H}_0) \oplus \vec{r}_{s_a,s_b,s_c}$;
    return $(W, \vec{G}, \vec{z})$
```

Figure 5: Hybrid 3

Lemma 1 *Assuming the 3-TCCR assumption in Definition 3, then:*

$$\mathsf{Hybrid}_2^{\mathcal{O}_\Delta^{\mathsf{tccr}}}(1^\kappa, f, \hat{x}) \overset{\mathsf{c}}{\approx} \mathsf{Hybrid}_3^{\mathsf{Rand}}(1^\kappa, f, \hat{x})$$

Proof: For the sake of contradiction, let us assume that there exists a PPT distinguisher $\mathcal{D}_{\mathsf{Hybrid}}$ which can distinguish between Hybrid_2 and Hybrid_3 for the given $f, \hat{x}$ with a non-negligible probability ϵ. Then, we show that there exists a PPT distinguisher $\mathcal{D}_{\mathsf{TCCR}}$ that can distinguish between the outputs of a $3-TCCR$ oracle and a Rand oracle with non-negligible advantage ϵ', by making a black box call to $\mathcal{D}_{\mathsf{Hybrid}}$. The distinguisher $\mathcal{D}_{\mathsf{TCCR}}^{\mathcal{O}(.)}$ with an oracle access to $\mathcal{O}(.)$ can be described as follows:

– For every input gate, $\mathcal{D}_{\mathsf{TCCR}}$ samples strings $W_i^{x_i}$ for all $i \in \mathsf{Inputs}(f)$, and sets them to be the active input labels for the garbling scheme.

– To derive the rest of the active labels, it does the following:

- If the gate i is an XOR gate with input wires a, b and active inputs x_a, x_b, it sets the active output label W_i to be the XOR of the two input labels, ie.. $W_i^{x_i} = W_a^{x_a} \oplus W_b^{x_b}$ and active output to be $x_i = x_a \oplus x_b$.
- If the gate i is a $\mathcal{G}_{\text{tri}}$ gadget, with input wires a, b, c and active inputs x_a, x_b, x_c, then it sets the active output to be $x_i = x_a \wedge (x_b \oplus x_c)$, and computes the active output label $W_i^{x_i}$ and the garbled gate $F_i = \vec{G}_i, \vec{z}_i$ to be $\mathsf{Hyb3Gadget}^{\mathcal{O}}(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$ where $\mathcal{O}$ is the oracle $\mathcal{D}_{\mathsf{TCCR}}$ has access to in the 3-TCCR distinguishing game.

– For every output gate i, it computes the decoding $D_i^{x_i}$ by computing H on $W_i^{x_i}$, and computes $D_i^{1-x_i}$ by calling $\mathcal{O}(.)$ on $W_i^{1-x_i}$.

– After generating, it sets $(\hat{F} = \{F_i\}_{i \in f \setminus \mathsf{Inputs}(f)}, \hat{X} = \{X_i\}_{i \in f}, \hat{D} = \{D_i\}_{i \in \mathsf{Outputs}(f)})$ in the way described above.

– $\mathcal{D}_{\mathsf{TCCR}}$ invokes the distinguisher $\mathcal{D}_{\mathsf{Hybrid}}$ on the inputs $(\hat{F}, \hat{X}, \hat{D})$.

– If $\mathcal{D}_{\mathsf{Hybrid}}$ outputs 0 (meaning Hybrid_2), then $\mathcal{D}_{\mathsf{TCCR}}$ outputs 0 (meaning $\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}(.)$). If $\mathcal{D}_{\mathsf{Hybrid}}$ outputs 1 (meaning Hybrid_3), then $\mathcal{D}_{\mathsf{TCCR}}$ outputs 1 (meaning $\mathsf{Rand}(.)$)

We see that $\mathcal{D}_{\mathsf{TCCR}}$ is a PPT algorithm, as it just emulates the garbling scheme on randomly sampled active labels, and makes only polynomial many calls to the oracle $\mathcal{O}(.)$. We now show that the distinguisher $\mathcal{D}_{\mathsf{TCCR}}$ has the distinguishing advantage $\epsilon' \geq \epsilon$ in the $3 - TCCR$ game as that of $\mathcal{D}_{\mathsf{Hybrid}}$ in the hybrid indistinguishability game. If the oracle in the $3 - TCCR$ game was $\mathcal{O}_{\Delta}^{3-\mathsf{tccr}}(.)$, then the output of $\mathcal{D}_{\mathsf{TCCR}}$ would look exactly as in Hybrid_2, and if the oracle in the $3 - TCCR$ game was $\mathsf{Rand}(.)$, then the output of $\mathcal{D}_{\mathsf{TCCR}}$ would look exactly as the output in Hybrid_3. Therefore, $\mathcal{D}_{\mathsf{TCCR}}$ wins the $3 - TCCR$ game if $\mathcal{D}_{\mathsf{Hybrid}}$ wins the hybrid indistinguishability game. Therefore we have that

$$\mathrm{Adv}_{H,\mathcal{R}}^{3-\mathsf{tccr}}(\mathcal{D}_{\mathsf{TCCR}}) \geq \epsilon(\kappa).$$

By assumption ϵ was a non-negligble function. However, as H was 3-tweakable circular correlation robust, this contradicts the $3 - TCCR$ assumption. □

Hybrid 4: In this hybrid, we aim to replace the value $z_{\mathsf{bot}} \parallel G$ and $D_i^{1-x_i}$ in $\mathsf{Hybrid}_3^{\mathsf{Rand}}$ with random values. It is straightforward for the term $D_i^{1-x_i}$. For $z_{\mathsf{bot}} \parallel G$, we notice that it is of the form [linear combination of labels] $\oplus$ Rand. Therefore, the Rand output acts like a one-time pad, enabling us to replace the entire string with a random value. By the above arguments, it is clear that $\mathsf{Hybrid}_3^{\mathsf{Rand}} \overset{c}{\approx} \mathsf{Hybrid}_4$. We formally describe the hybrid 4 in Figure 6 to generate the changes mentioned above.

Hybrid 5: Here we aim to replace the values of $R, \vec{r}$ with random values. We know that $\mathcal{R}$ was specifically chosen such that marginal view R_{s_a,s_b,s_c} was uniform for all t and all s_a, s_b, s_c. So one can uniformly sample $\vec{r}_{s_a,s_b,s_c}$, use $\mathsf{DecodeR}$ on this chosen $\vec{r}_{s_a,s_b,s_c}$ to reconstruct the marginal view R_{s_a,s_b,s_c}. We also stress that this is the only thing needed, not the entire R, per the evaluator's view. So, we make these changes, and as a result of these changes, the $(\vec{z})_{\text{top}}$ term also becomes uniform due to $\vec{r}_{s_a,s_b,s_c}$ acting as a one time pad on it. We also notice that the inputs x_a, x_b, x_c were only needed for the values (p_a, p_b, s_c), which was only needed for $\mathsf{SampleR}$. In this step, as we eliminated the need for $\mathsf{SampleR}$, we can change the inputs to the hybrid from $\hat{x}$ to $\hat{y}$, where now it needs the output only for decoding. We formally describe the hybrid 5 in Figure 7 to generate the changes mentioned above. It is straightforward to show that Hybrid_5 is indistinguishable from the output of Hybrid_5 as the marginal view was chosen uniformly at random, and these two outputs will hence be identically distributed.

Simulator: The last step of the hybrid is to observe that the last three lines of $\mathsf{Hyb3GADGET}_4$ depict the $\mathsf{Eval3GADGET}$ algorithm that the evaluator uses to evaluate AND gates. Therefore, we can replace this part as necessary, which gives us Hybrid_6, which is nothing but the Sim algorithm described in Figure 2. These

```
procedure $\mathsf{Hybrid}_4(1^\kappa, f, \hat{x})$:
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i^{x_i} \leftarrow_\$ \{0,1\}^\kappa$;
        $X_i := W_i^{x_i}$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        $\{a, b, c\} := \mathsf{GateInputs}(f, i)$;
        if $i \in \mathrm{XORGates}(f)$ then
            $W_i^{x_i} := W_a^{x_a} \oplus W_b^{x_b} \oplus W_c^{x_c}$;
            $x_i := x_a \oplus x_b \oplus x_c$;
            $X_i := W_i^{x_i}$;
            continue;
        $x_i := x_a \wedge x_b \wedge x_c$;
        $(W_i^{x_i}, \vec{G}_i, \vec{z}_i) := \mathsf{Hyb3GADGET}_4(W_a^{x_a}, W_b^{x_b}, W_c^{x_c})$;
        $F_i := (\vec{G}_i, \vec{z}_i),\ X_i := W_i^{x_i}$;
    for $i \in \mathsf{Outputs}(f)$ do
        $D_i^{x_i} := H'(W_i^{x_i}, i)$;
        $D_i^{1-x_i} \leftarrow_\$ \{0,1\}^\kappa$;
    return $(\hat{F}, \hat{X}, \hat{D})$;
procedure $\mathsf{Hyb3GADGET}_4(W_a^{x_a}, W_b^{x_b}, W_c^{x_c}, i)$:
    $s_a := \mathrm{lsb}(W_a^{x_a}),\ p_a := x_a \oplus s_a$;
    $s_b := \mathrm{lsb}(W_b^{x_b}),\ p_b := x_b \oplus s_b$;
    $s_c := \mathrm{lsb}(W_c^{x_c}),\ p_c := x_c \oplus s_c$;
    $T' \leftarrow \mathsf{3GADGET}(p_a, p_b, p_c),\ R \leftarrow_\$ S_{T'},\ \vec{r} := R_{[1:9]}$ ;
    $\vec{z}_{\mathsf{bot}} \parallel \vec{G} \leftarrow_\$ \{0,1\}^{\frac{4\kappa}{3}+42}$;
    $W := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{G} \end{bmatrix} \oplus M_e \mathrm{msb}_{\frac{\kappa}{3}}(\vec{H}_0) \oplus R_{s_a,s_b,s_c} \begin{bmatrix} W_a^{x_a} \\ W_b^{x_b} \\ W_c^{x_c} \end{bmatrix}$;
    $\vec{z}_{\mathsf{top}} := V_{s_a,s_b,s_c} \begin{bmatrix} 0 \\ \vec{z}_{\mathsf{bot}} \end{bmatrix} \oplus M_e \mathrm{lsb}_{63}(\vec{H}_0) \oplus \vec{r}_{s_a,s_b,s_c}$;
    return $(W, \vec{G}, \vec{z})$
```

Figure 6: Hybrid 4

views are identical as replacing three lines with a function that does the exact computation. Hence, we have shown through a sequence of hybrids that for any given circuit f and inputs $\hat{x}$, we have that:

$$(\hat{F}_s, \hat{X}_s, \hat{D}_s) \overset{c}{\approx} (\hat{F}_r, \hat{X}_r, \hat{D}_r)$$

Obliviousness: The only difference between achieving obliviousness and privacy is that the simulator is not given $y = f(x)$ and is not expected to output the decoding information. Our simulator can be easily modified for this case, by only returning the fake garbled gates without the decoding information. This requires the simulator to not use y for creating the fake garbling, which is the case anyway.

Authenticity: For authenticity, we need to show that any PPT adversary $\mathcal{A}$ that receives as input $(\hat{F}, \hat{X}, \hat{D})$ computed as in the real execution, cannot produce an inactive output label. That is, the adversary succeeds if it can get Decode to output $1 \oplus x_k$ for any output gate k with active value x_k. Decode will output $1 \oplus x_k$ iff it receives Y' such that $Y' = D_k^{1 \oplus x_k} = H'(Y_k, k)$. In the simulator for privacy (cf. Figure 2), we show that this is indistinguishable from a freshly sampled random κ bit string, and hence the probability that $\mathcal{A}$ finds this is $\frac{1}{2^\kappa}$

This concludes that the garbling scheme in Figure 1 satisfies Definition 2. □

```
procedure $\mathsf{Hybrid}_5(1^\kappa, f, \hat{y})$:
    for $i \in \mathsf{Inputs}(f)$ do
        $W_i \leftarrow_\$ \{0,1\}^\kappa$;
        $X_i := W_i$;
    for $i \notin \mathsf{Inputs}(f)\{in\ topo.order\}$ do
        $\{a,b,c\} := \mathsf{GateInputs}(f,i)$;
        if $i \in \mathrm{XORGates}(f)$ then
            $W_i := W_a \oplus W_b \oplus W_c$;
            $X_i := W_i$;
            continue;
        $(W_i, \vec{G}_i, \vec{z}_i) := \mathsf{Hyb3GADGET}_5(W_a, W_b, W_c)$;
        $F_i := (\vec{G}_i, \vec{z}_i)$, $X_i := W_i$;
    for $i \in \mathsf{Outputs}(f)$ do
        $D_i^{y_i} := H'(W_i, i)$;
        $D_i^{1-y_i} \leftarrow_\$ \{0,1\}^\kappa$;
    return $(\hat{F}, \hat{X}, \hat{D})$;
procedure $\mathsf{Hyb3GADGET}_5(W_a, W_b, W_c, i)$:
    $s_a := \mathrm{lsb}(W_a)$, $s_b := \mathrm{lsb}(W_b)$, $s_c := \mathrm{lsb}(W_c)$;
    $\vec{z} \leftarrow_\$ \{0,1\}^{63}$, $\vec{r}_{s_a,s_b,s_c} \leftarrow_\$ \{0,1\}^{27}$, $\vec{G} \leftarrow_\$ \{0,1\}^{\frac{4\kappa}{3}}$;
    $R_{s_a,s_b,s_c} := \mathsf{DecodeR}(\vec{r}_{s_a,s_b,s_c}, s_a, s_b, s_c)$;
    $W := V_{s_a,s_b,s_c}\begin{bmatrix}0\\ \vec{G}\end{bmatrix} \oplus M_e \mathrm{msb}_{\frac{\kappa}{3}}(\vec{H}_0) \oplus R_{s_a,s_b,s_c}\begin{bmatrix}W_a\\ W_b\\ W_c\end{bmatrix}$;
    return $(W, \vec{G}, \vec{z})$
```

Figure 7: Hybrid 5

4.3 Garbling Circuits with Free-XOR and 2-Input AND Gates

We now describe how the garbling scheme described in Figure 1, can be used to garble a circuit with 2-input XOR and AND gates while maintaining the free-XOR optimization. Notice that the gadget $\mathcal{G}_{\text{tri}} = x \wedge (y \oplus z)$ can be reduced to a 2-input AND gate $x \wedge y$ by setting $z = 0$. Therefore, it should be possible to convert any circuit with just AND and XOR gates to a circuit with XOR gates and the $\mathcal{G}_{\text{tri}}$ gadgets by just replacing all the AND gates with $\mathcal{G}_{\text{tri}}$ gadgets and hard-coding the value of the wire $z = 0$ corresponding to each gadget. This implies additional κ bits. We hence have the following corollary,

Corollary 2 *Assuming the 3-TCCR assumption (Definition 3), the garbling scheme described in Figure 1 meets Definition 2 for any circuit comprised of 2-input XOR and AND gates, where the garbled circuit is of size $n \cdot \left(4\frac{\kappa}{3} + 63\right) + \kappa$ bits, where κ is the security parameter and n is the number of AND gates.*

Let us describe the above transformation formally. Let there be n 2-input AND gates in the original circuit. Let us denote these gates by

$$g_i = x_i \wedge y_i, \quad i \in [n]$$

where x_i and y_i are the inputs to the ith 2-input AND gate. The above attempt translates to having n $\mathcal{G}_{\text{tri}}$ gadgets replacing the n 2-input AND gates by introducing n additional inputs z_i and computing

$$g'_i = x_i \wedge (y_i \oplus z_i), \quad i \in [n]$$

by hardcoding the value for each $z_i = 0$. This introduces n additional wire labels; hence, the communication cost increases by $n\kappa$ bits. However, we notice that there is no need to introduce an additional input wire for every gadget since we can reuse the same wire label for all the gadgets. This would imply that the garbled circuit would still have n $\mathcal{G}_{\text{tri}}$ gadgets:

$$g'_i = x_i \wedge (y_i \oplus z), \quad i \in [n].$$

Notice that if we now hardcode $z = 0$, we still have the value of g'_i to be the same as g_i. However, we are not adding n inputs, but just one input z. This would only increase the communication complexity by κ bits. The garbling scheme is oblivious to the fact that the input wire z is used in multiple gadgets, and is hence correct. We conclude with a garbling scheme for a circuit with an arbitrary number of XOR gates and n 2-input AND gates, with a communication complexity of $n \cdot \left(\frac{4\kappa}{3} + \mathcal{O}(1)\right) + \kappa$ bits.

5 3-Input AND Gate Impossibility

In section 4.1, we discussed how one can efficiently garble the gadget $\mathcal{G}_{\text{tri}}$ with a scheme represented by the following equation:

$$V \begin{bmatrix} \vec{D}_0 \\ \vec{G} \end{bmatrix} = M\vec{H} \oplus R \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix} \oplus T\vec{\Delta} \tag{22}$$

A natural direction to pursue is to try extending this equation to garble other gadgets. Can a 3-input AND gate be garbled by this equation? We now show that the answer is negative. In fact, we show that one cannot garble a 3-input AND gate using any garbling scheme represented by Equation (22) when the oracle queries are restricted to linear functions of the input labels. We will later extend it to show that we cannot garble any higher input gadget, with a degree-3 term in its Boolean expression, when written in terms of XOR and AND gates.

The rest of the section is as follows: In Section 5.1 we rewrite Equation (22) such that the impossibility becomes more obvious. Then, in Section 5.2 we formalize what it means to restrict the oracle queries to linear functions. Finally, in Section 5.3 we prove that no garbling scheme in the form of Equation (22), when restricted to linear queries to the oracle, can be used to garble a 3-input AND gate. In particular, this impossibility captures half-gates [41], three-halves [34], our scheme presented in Section 4, as well as some natural extensions of these schemes.

5.1 Revisiting the Garbling Equation

Let n be the number of slices used by the garbling scheme. We already know that for n slices there are n equations in the system represented by Equation (22), one for each input (x, y, z). Let us rewrite this subsystem such the equations are parameterized by the input (x, y, z). Then,

$$V_{x,y,z} \begin{bmatrix} \vec{D}_0 \\ \vec{G} \end{bmatrix} = M_{x,y,z}\vec{H} \oplus R_{x,y,z} \begin{bmatrix} \vec{A}_0 \\ \vec{B}_0 \\ \vec{C}_0 \\ \vec{\Delta} \end{bmatrix} \oplus T_{x,y,z}\vec{\Delta}$$

where this system is restricted to the part corresponding to inputs (x, y, z). We further rewrite the equation to also incorporate the following changes:

1. Recall from Section 2, we defined T to be determined by the function

$$T_{\mathsf{out}}(x, y, z) = (x \oplus p_x) \wedge ((y \oplus p_y) \oplus (z \oplus p_z)).$$

   This function corresponds to the $\mathcal{G}_{\text{tri}}$ gadget in particular. Here, we are using a 3-input AND gate. Therefore, the changed function T'_{out} corresponding to the 3-input AND gate would be

$$T'_{\mathsf{out}}(x, y, z) = (x \oplus p_x) \wedge (y \oplus p_y) \wedge (z \oplus p_z) \tag{23}$$

Instead of including a separate matrix T, that is derived using Equation (23) acting on Δ, we replace the vector $\vec{D_0}$ with

$$\vec{D} = \vec{D_0} \oplus T'_{\mathsf{out}}(x, y, z)\vec{\Delta}.$$

We now have the following system corresponding to inputs (x, y, z):

$$V_{x,y,z} \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} = M_{x,y,z}\vec{H} \oplus R_{x,y,z} \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{C_0} \\ \vec{\Delta} \end{bmatrix}.$$

2. Similarly, recall that we imposed a particular structure on R to ensure that only active labels were selected by the evaluator. Another way to achieve the same is to rewrite the term corresponding to $R_{x,y,z}$ as follows:

$$R_{x,y,z} \begin{bmatrix} \vec{A_0} \\ \vec{B_0} \\ \vec{C_0} \\ \vec{\Delta} \end{bmatrix} \rightarrow R'_{x,y,z} \begin{bmatrix} \vec{A_0} \oplus (x \oplus p_x)\vec{\Delta} \\ \vec{B_0} \oplus (y \oplus p_y)\vec{\Delta} \\ \vec{C_0} \oplus (z \oplus p_z)\vec{\Delta} \end{bmatrix}.$$

Now, $R'_{x,y,z}$ does not require any explicit constraint as the selected label is based on the active input, which is already included in the vector. Therefore, $R'_{x,y,z}$ will be the same as $R_{x,y,z}$ except that it excludes those columns corresponding to Δ in $R_{x,y,z}$. Summarizing, we reduced Equation (22) to the following:

$$V_{x,y,z} \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} = M_{x,y,z}\vec{H} \oplus R'_{x,y,z} \begin{bmatrix} \vec{A_0} \oplus (x \oplus p_x)\vec{\Delta} \\ \vec{B_0} \oplus (y \oplus p_y)\vec{\Delta} \\ \vec{C_0} \oplus (z \oplus p_z)\vec{\Delta} \end{bmatrix}. \tag{24}$$

5.2 Linear Queries

As stated earlier, we restrict ourselves to the setting where the only queries allowed to the oracle are linear functions of input labels. The half-gates scheme [41], three-halves scheme [34], and our scheme in Section 4.1 all adhere to this restriction. In fact, using linear functions of the labels was a major source of improvement in communication costs for an AND gate in all these schemes.

Consider the following function $F_l : \{0,1\}^{3\kappa} \rightarrow \{0,1\}^\kappa$, parameterised by the 3-bit vector $l = \{l_A, l_B, l_C\}$:

$$F_l(A, B, C) = l_A \cdot A \oplus l_B \cdot B \oplus l_C \cdot C$$

where A, B and C are κ-bit strings. Let $\mathcal{F}_{\mathsf{Lin3},\kappa}$ be the family of functions containing all such functions F_l. Therefore, $\mathcal{F}_{\mathsf{Lin3},\kappa}$ can be represented by the set of all possible 3-bit vectors $\mathcal{F}_{\mathsf{Lin3},\kappa} = \{l\}$, where $\{l\}, l \in \{0,1\}^3$ is the mask for the linear combination queried to the oracle.

Recall that we are in the free-XOR setting [27], and if A_x, B_y and C_z are the wire labels corresponding to input tuple (x, y, z), then the function

$$F_l(A_x, B_y, C_z) = l_A \cdot A_x \oplus l_B \cdot B_y \oplus l_C \cdot C_z$$

is an output label for the gate computing the following linear function (in the plain domain):

$$f_l(x, y, z) = l_A \cdot x \oplus l_B \cdot y \oplus l_C \cdot z$$

where f_l is also parameterised by the same 3-bit vector $l = \{l_A, l_B, l_C\}$ as F_l. This is because the free-XOR setting reduces computing output labels for linear functions to computing the sum of corresponding input labels. Hence, $F_l(A_x, B_y, C_z)$ has only two possible values. Let us derive these two values, which are essentially the 0-label and the 1-label for the output wire of the gate computing f_l. We know that any label A_x corresponding to the plain input x can be rewritten as $A_x = A_0 \oplus x\Delta$. Therefore:

$$\begin{aligned}
F_l(A_x, B_y, C_z) &= l_A \cdot A_x \oplus l_B \cdot B_y \oplus l_C \cdot C_z \\
&= l_A \cdot (A_0 \oplus x\Delta) \oplus l_B \cdot (B_0 \oplus y\Delta) \oplus l_C \cdot (C_0 \oplus z\Delta) \\
&= l_A \cdot A_0 \oplus l_B \cdot B_0 \oplus l_C \cdot C_0 \oplus \Delta(l_A \cdot x \oplus l_B \cdot y \oplus l_C \cdot z) \\
&= F_l(A_0, B_0, C_0) \oplus f_l(x, y, z)\Delta.
\end{aligned}$$

Let $X_{l,f_l} = F_l(A_x, B_y, C_z)$ be the wire label representing the output of the gate computing $f_l(x, y, z)$. Then, without loss of generality, we have $X_{l,0} = F_l(A_0, B_0, C_0)$ and $X_{l,1} = X_{l,0} \oplus f_l(x, y, z)\Delta$.

Having determined the structure of the queries made, let us determine the structure of the oracle responses to linear queries. An oracle query on any label A_x can be written in the following way:

$$H(A_x) = (1 \oplus x) \cdot H(A_0) \oplus x \cdot H(A_1).$$

Generalizing this description with $X_{l,f_l} = F_l(A_x, B_y, C_z)$, we have that

$$H(F_l(A_x, B_y, C_z)) = (1 \oplus f_l(x, y, z)) \cdot H(X_{l,0}) \oplus f_l(x, y, z) \cdot H(X_{l,1}), \tag{25}$$

or alternatively:

$$H(F_l(A_x, B_y, C_z)) = \begin{bmatrix} 1 \oplus f_l(x, y, z) & f_l(x, y, z) \end{bmatrix} \begin{bmatrix} H(X_{l,0}) \\ H(X_{l,1}) \end{bmatrix} \tag{26}$$

Equation (26) establishes that when the queries to the RO are restricted to linear combinations on the input labels, then the coefficients of the oracle responses to these queries must be some linear combination of the plain input bits (x, y, z).

5.3 The Impossibility Proof

Using Equations (22)-(24), we prove the following theorem statement:

Theorem 3 *Let $\mathcal{G}$ be an n-sliced garbling scheme represented in the form of Equation (22). Then, $\mathcal{G}$ cannot garble a 3-input AND gate if the queries to the oracle $H(.)$ are restricted to be a linear function of the input labels A, B, C.*

Proof: First, we use the argument in Section 5.1 to rewrite Equation (22) as Equation (24). Then, we use the argument in Section 5.2 to show that having only linear queries means that the matrix M has values that are only linear functions of the inputs (x, y, z). Because M, V and R share the same column space, the values of V and R are also linear functions of the inputs. As a result, we show that to maintain correctness, such a scheme can only garble gadgets with degree at most 2. Since the 3-input AND gate has degree 3, it cannot be garbled using such a scheme. We now provide a formal proof.

Let $g(x, y, z) = x \wedge y \wedge z$ be a 3-input AND gate and assume towards contradiction that there exists a garbling scheme $\mathcal{G}$ represented by Equation (22) that can garble g correctly. For such a scheme, it is possible to break down the system of equations into 8 sub-systems having n equations each. Each subsystem corresponds to a specific input (x, y, z) and is represented by the equation displayed in Equation (24). Each equation in this subsystem corresponds to the encryption of one slice of the output label, with respect to the input tuple (x, y, z). A unique linear combination of the oracle responses is used to encrypt this output slice for the sake of privacy. For any linear query, recall that $F_{l_j}(A_x, B_y, C_z) = X_{l_j,f_{l_j}}$. Therefore, using Equation (25), we can rewrite the oracle responses as:

$$\bigoplus_{j=1}^{q} H(F_{l_j}(A_x, B_y, C_z)) = \bigoplus_{j=1}^{q} [(1 \oplus f_{l_j}(x,y,z)) \cdot H(X_{l_j,0})] \oplus [f_{l_j}(x,y,z) \cdot H(X_{l_j,1})]$$

$$= \begin{bmatrix} 1 \oplus f_{l_1}(x,y,z) & f_{l_1}(x,y,z) & \cdots & 1 \oplus f_{l_q}(x,y,z) & f_{l_q}(x,y,z) \end{bmatrix} \begin{bmatrix} H(X_{l_1,0}) \\ H(X_{l_1,1}) \\ \cdot \\ \cdot \\ \cdot \\ H(X_{l_q,0}) \\ H(X_{l_q,1}) \end{bmatrix}$$

where $F_{l_1}, \cdots, F_{l_q} \in \mathcal{F}_{\mathsf{Lin3},\kappa}$. Therefore it is clear that when a linear combination of the oracle responses is viewed as a product of a matrix and a vector, the coefficient of each response in the vector gets translated to components of the matrix at appropriate indices.

Recall Equation (24):

$$V_{x,y,z} \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} = M_{x,y,z}\vec{H} \oplus R'_{x,y,z} \begin{bmatrix} \vec{A_0} \oplus (x \oplus p_x)\vec{\Delta} \\ \vec{B_0} \oplus (y \oplus p_y)\vec{\Delta} \\ \vec{C_0} \oplus (z \oplus p_z)\vec{\Delta} \end{bmatrix}.$$

We need to now use the above observations to derive a structure for the term $M_{x,y,z}\vec{H}$. We will then use this to show that the structure imposes constraints on the matrices $V_{x,y,z}$ and $R_{x,y,z}$ creating an isolated Degree-3 term on the left hand side of the system, which is not present on the right hand side. The only solution then is when $\Delta = 0$, violating the correctness of the scheme.

Let $\mathcal{Q} = \{F_{l_1}(A_x, B_y, C_z), F_{l_2}(A_x, B_y, C_z), \cdots, F_{l_q}(A_x, B_y, C_z)\}$ be the set of oracle queries made by the garbler. We have showed that for any linear function $F_l(A_x, B_y, C_z)$ there are only two possible values, $X_{l,0}$ and $X_{l,1}$. Therefore, the set $\mathcal{Q}$ can be represented as $\mathcal{Q} = \{X_{l_1,0}, X_{l_1,1}, \cdots, X_{l_q,0}, X_{l_q,1}\}$. Let $\vec{H} = [H(X_{l_1,0}), H(X_{l_1,1}), \cdots, H(X_{l_q,0}), H(X_{l_q,1})]$ be the vector containing the oracle responses to the queries in $\mathcal{Q}$.

Recall that for a given input (x, y, z), there are n output slices, and the subsystem in Equation (24) has n equations corresponding to the encryption of these output slices. Let $M^i_{x,y,z} = \{m^i_1, m^i_2, \cdots m^i_q\}$ be the i'th row of the matrix $M_{x,y,z}$ corresponding to the i'th equation of the subsystem. We claim that every component of the vector $M^i_{x,y,z}$ is either a linear function of (x, y, z) or 0. This is because the term $M^i_{x,y,z}\vec{H}$ is the linear combination of the oracle responses that is used to mask the i'th output slice, represented as a product of a matrix and a vector of queries, as shown in Equation (25). The only difference is that $\vec{H}$ might contain responses that are not used in the particular linear combination. For all those responses, the coefficient would be 0. Therefore, as the coefficient of each query in the vector is a linear function of (x, y, z) or 0, each component of $M^i_{x,y,z}$ is a linear function of (x, y, z) or 0. This claim is true for all slices of the output label, and hence for all rows of the matrix $M_{x,y,z}$.

Let $M_{x,y,z}$ be a matrix as described above. Consider, $B^M_{x,y,z} = \{M^1_{x,y,z}, \cdots, M^r_{x,y,z}\}$ to be a basis for the column space of $M_{x,y,z}$, where r is the rank of $M_{x,y,z}$. Then, for every i, each component of the basis vector $M^i_{x,y,z}$ is also a linear function of (x, y, z) or 0. Since we know that $V_{x,y,z}$ needs to also have the same column space as $M_{x,y,z}$ and by the structure of the Equation (24)

$$\mathsf{colspace}(R'_{x,y,z}) \subseteq \mathsf{colspace}(M_{x,y,z}) = \mathsf{colspace}(V_{x,y,z}). \tag{27}$$

So reconsidering Equation(24):

$$V_{x,y,z} \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} = M_{x,y,z}\vec{H} \oplus R'_{x,y,z} \begin{bmatrix} \vec{A_0} \oplus (x \oplus p_x)\vec{\Delta} \\ \vec{B_0} \oplus (y \oplus p_y)\vec{\Delta} \\ \vec{C_0} \oplus (z \oplus p_z)\vec{\Delta} \end{bmatrix}$$

We know that the columns of $V_{x,y,z}$, $M_{x,y,z}$, and $R_{x,y,z}$ are linear functions of (x, y, z) or 0.

By rewriting this system as a linear equation in x, y, z, and equating the coefficients of x, y, z and the constant on both sides of the equation, we can see that Δ must be 0. Firstly, we stress that no column of $V_{x,y,z}$ is an all zeroes because by definition $V_{x,y,z}$ has linearly independent columns of $M_{x,y,z}$. Also, recall that

$$\vec{D} = \vec{D_0} \oplus (x \oplus p_x)(y \oplus p_y)(z \oplus p_z)\vec{\Delta}$$

which introduces a cubic term on the left hand side, irrespective of the value of $V_{x,y,z}$. The coefficient of this term is $\vec{\Delta}$. On the right hand side, $M\vec{H}$ generates only linear terms in x, y or z. The part corresponding to $R'_{x,y,z}$ generates at most a quadratic term, as from Equation (27), $R'_{x,y,z}$ has only linear terms, and the terms $(x \oplus p_x)\vec{\Delta}$, $(y \oplus p_y)\vec{\Delta}$ and $(z \oplus p_z)\vec{\Delta}$ introduces another linear term. Irrespective of the values of $M_{x,y,z}$ and $R'_{x,y,z}$, the maximum degree possible on the right hand side of the equation is 2. Therefore, the only way to balance the degree of the terms on both sides is to set $\vec{\Delta} = 0$, which would violate the correctness of the scheme.

We therefore conclude that $\mathcal{G}$ has a cubic term on the left hand side, and a term that is at most quadratic on the right hand side, resulting in the desired contradiction. $\square$

In the proof for Theorem 3, as stated earlier, we show that to maintain correctness, such a scheme can only garble gadgets with degree at most 2. Therefore, it is easy to see that this result would extend to any higher fan-in gadget with a degree greater than 3. Thus, we have the following corollary:

Corollary 4 *Let $\mathcal{G}$ be an n-sliced garbling scheme represented in the form of Equation (22). Then, $\mathcal{G}$ cannot garble any gadget that has a degree at least 3, if the queries to the oracle $H(.)$ are restricted to be a linear function of the input labels A, B, C.*

6 Oblivious Garbling

In this section, we introduce the notion of oblivious garbling, where the garbler remains oblivious to the circuit's functionality, knowing only its topology. We formally define this notion and present a scheme for arbitrary circuits consisting of 2-input AND and XOR gates with a communication cost of $O(|f|)$ bits, where $|f|$ is the size of the circuit, and as many OTs.

To accomplish this, we first replace all gates in the circuit with the $\mathcal{G}_{\text{tri}}$ gadget, as introduced in Section 4.1. In Section 4.3 we illustrated how this gadget can be adapted into an AND gate by setting $z = 0$ and utilizing x and y as input lines. Similarly, the gadget can be reduced to an XOR gate by setting $x = 1$ and using y and z as inputs. We use this observation to devise a garbling scheme for circuits containing AND and XOR gates, without disclosing the gate type during the garbling process.

6.1 Obliviously Garbling a Single Gate

We continue by illustrating the challenges involved in achieving oblivious garbling for an entire circuit, and explaining how we address them. As suggested above, the goal is to devise a technique that takes advantage of the fact that inputs to the $\mathcal{G}_{\text{tri}}$ gadgets can be programmed to reduce them to either an XOR or an AND gate. While programming inputs for input gates where the evaluator can control the inputs is straightforward, programming intermediate gates where the evaluator does not have access to the wire values is more challenging. Using the $\mathcal{G}_{\text{tri}}$ gadget to implement certain gates requires programming its inputs, effectively turning it into an input gadget itself. However, this reveals information about the type of gate: the wire used for programming (x for XOR or z for AND) reveals the functionality of the gate. Therefore, we need a solution that can let an evaluator reduce the gadget to an XOR gate or an AND gate, while hiding this reduction from the garbler.

We take a two-step approach to solving this problem. First, we use the garbling scheme from Section 4.2 to garble each gadget in the circuit independently. This preserves obliviousness during garbling, since the only leakage is the number of $\mathcal{G}_{\text{tri}}$ gadgets. However, it's incomplete, since the evaluation still requires revealing the input labels for subsequent gadgets in topological order. To solve this problem, we use the "soldering" technique of [16, 31]. This technique allows the dynamic construction of circuits from independently garbled gates during evaluation, facilitating the connection of the output wires of each gadget to the corresponding input wires of the next, effectively constructing the desired circuit without revealing any gate information. We will first discuss how the soldering technique works, and then illustrate how it can be used to obliviously garble a non-input gate.

Soldering Technique: The soldering technique, as the name suggests, allows one to connect independently generated gates. It was introduced in [31] to allow the transformation of the key representing bit b on one wire into the key representing bit b on another wire. This is done by computing some auxiliary information called the *soldering*. More formally, let's consider a scenario where we need to solder the output wire a of gate g_i to the first input wire b of gate g_{i+1}. To do this, we calculate the soldering value:

$$S_{i,i+1} = A_0 \oplus B_0,$$

where A_0, B_0 are the 0-labels for wires a, b, respectively. Given the label representing bit-j for wire a as A_j, one can derive the label representing bit-j for wire b (denoted as B_j) using the solder value $S_{i,i+1}$ through the following computation:

$$A_j \oplus S_{i,i+1} = A_0 \oplus j\Delta \oplus A_0 \oplus B_0 = B_0 \oplus j\Delta = B_j.$$

Garbling a single non-input gate: Consider the i-th non-input gate g'_i in the original circuit, with input wires x', y'. If we convert this to a $\mathcal{G}_{\text{tri}}$ gate g_i, say with wires x, y, z, the values these wires need to take depend on what g'_i needs to be programmed to do. The following list illustrates the options for AND and XOR gates:

- When g'_i is an XOR gate, the inputs of g should be $x = 1$, $y = y'$, $z = x'$.
- When g'_i is an AND gate, the inputs of g should be $x = x'$, $y = y'$, $z = 0$.

As discussed earlier, if only one of x, z is set to input in the topology, it becomes obvious which gate the gadget reduces to. We know that this is not a problem for input gates, since in this case all wires are input wires. Therefore, even in the case of non-input gates, we garble the gadget as if all wires were input wires, i.e., we garble each gadget independently, without considering that these wires might be the output wires of a gadget in the previous layer. The garbling scheme for the $\mathcal{G}_{\text{tri}}$ gadget in Section 4.2 accomplishes this. Let's assume that the inputs to gate g'_i in the original circuit were the outputs p, q of gates g'_{i-1} and g'_{i-2}, respectively. Since g_{i-1} and g_{i-2} were also garbled independently, evaluating them would only reveal the output labels P_{b_1}, Q_{b_2}, where b_1, b_2 are the values taken by wires p, q and P_{b_1}, Q_{b_2} are their corresponding labels. Having obtained these labels, if we wish to evaluate the gadget g_i as an AND gate, we need the labels X_{b_1}, Y_{b_2}, Z_0 for the wires x, y, z respectively. Similarly, to evaluate the gadget as an XOR gate, we will need the labels X_1, Y_{b_2}, Z_{b_1}. To achieve this, we use the soldering technique to transfer the bits b_1 and b_2 from P_{b_1} and Q_{b_2} to the corresponding labels of g_i. The garbling algorithm calculates these soldering values for both AND and XOR scenarios. We then transfer one set of values to the evaluator using a 1-out-of-2 OT protocol.

6.2 Defining Oblivious Garbling Schemes

We now define the notion of oblivious garbling. We modify the definition stated in Definition 2 to be able garble an entire circuit while hiding the circuit. In this case, the Garble algorithm no longer receives as input the circuit f, but rather just a leakage function Φ_g. We also allow Garble to output some auxiliary information that might be used by the evaluator to garble based on the choice of the circuit f.

Definition 4 *An oblivious garbling scheme consists of five algorithms:*

1. $\mathsf{Garble}(1^\kappa, \Phi_g(f)) \rightarrow (F, e, d, \mathsf{aux}_g)$ *which takes in as input the security parameter* κ*, and a leakage function* Φ_g *for the functionality* f *and returns the garbled circuit* F*, encoding information* e *for the wire labels and the decoding information* d *for the output wire labels, and some auxiliary information* aux_g*.*
2. $\mathsf{aux_e} := \mathsf{SelectGate}(\mathsf{aux}_g, f)$ *which takes as input the auxiliary information computed by* Garble *and outputs the auxiliary information needed for evaluation for a given circuit* f*.*
3. $X := \mathsf{Encode}(e, x)$ *which takes in as input the encoding information* e*, input* x *and gives the Garbled input* X*.*
4. $Y := \mathsf{Eval}(F, X, \mathsf{aux}_e)$ *which takes in as input the garbled circuit* F*, the garbled input* X*, the auxiliary information* aux_e *computed by* SelectGate *for* f *and returns the garbled evaluation of* f *on* x*, as* Y*.*
5. $y := \mathsf{Decode}(d, Y)$ *which takes in as input the decoding information* d*, garbled output* Y *and gives the output* y*.*

We want the following properties from a garbling scheme:

Correctness: *For any circuit* f *and input* x*, while* $(F, e, d, \mathsf{aux_g}) \leftarrow \mathsf{Garble}(1^\kappa, \Phi_g(f))$*, we have* $f(x) = \mathsf{Decode}(d, \mathsf{Eval}(F, \mathsf{Encode}(e, x), \mathsf{SelectGate}(\mathsf{aux}_g, f)))$ *except for negligible probability.*

Privacy: *There is a simulator* Sim *such that for any circuit* f *and input* x*, the following distributions are indistinguishable:*

```
$(F, e, d, \mathsf{aux}_g) \leftarrow \mathsf{Garble}(1^\kappa, \Phi_g(f))$
$X := \mathsf{Encode}(e, x)$
$\mathsf{aux}_e := \mathsf{SelectGate}(\mathsf{aux_g}, f)$
return $(F, X, d, \mathsf{aux}_e)$
```

```
$(F, X, d, \mathsf{aux}_e) \leftarrow \mathsf{Sim}(1^\kappa, f, f(x))$
return $(F, X, d, \mathsf{aux}_e)$
```

Obliviousness: *There is a simulator* Sim *such that for any circuit* f *and input* x*, the following distributions are indistinguishable:*

```
$(F, e, d, \mathsf{aux}_g) \leftarrow \mathsf{Garble}(1^\kappa, \Phi_g(f))$
$X := \mathsf{Encode}(e, x)$
$\mathsf{aux}_e := \mathsf{SelectGate}(\mathsf{aux_g}, f)$
return $(F, X, d, \mathsf{aux}_e)$
```

```
$(F, X, \mathsf{aux}_e) \leftarrow \mathsf{Sim}(1^\kappa, f)$
return $(F, X, \mathsf{aux}_e)$
```

Authenticity: *For any circuit* f *and input* x*, no PPT adversary* Adv *can make the following distribution output* TRUE *except with negligible probability:*

```
$(F, e, d, \mathsf{aux}_g) \leftarrow \mathsf{Garble}(1^\kappa, \Phi_g(f))$
$X := \mathsf{Encode}(e, x)$
$\mathsf{aux}_e := \mathsf{SelectGate}(\mathsf{aux_g}, f)$
$Y \leftarrow \mathsf{Adv}(F, d, X, \mathsf{aux}_e)$
return $\mathsf{Decode}(d, Y) \notin \{f(x), \bot\}$
```

6.3 Obliviously Garbling an Arbitrary Circuit

Formalising the techniques mentioned in Section 6.1, Figure 8 describes an oblivious garbling scheme that uses the GTri gadget to instantiate every gate in any circuit comprised of 2-input AND and XOR gates.

We next prove the following theorem.

Theorem 5 *Assuming the 3-TCCR assumption (Definition 3), the garbling scheme described in Figure 8 meets Definition 4 for any circuit with XOR and AND gates, implemented using* $\mathcal{G}_{\text{tri}}$ *gadgets. Each* $\mathcal{G}_{\text{tri}}$ *gadget has a garbled gate of size* $4\frac{\kappa}{3} + \mathcal{O}(1)$ *bits, where* κ *is the security parameter.*

```
procedure Init($\kappa$):
    $H := \mathcal{H}(\kappa)$;
procedure Garble($1^\kappa, \Phi_g(f) = \mathsf{Topology(f)}$):
    $\Delta :=_\$ \{0,1\}^{\kappa-1}||1$;
    for $i \in \mathsf{GadgetIn}(f)$ do
        $W_i^0 :=_\$ \{0,1\}^\kappa$;
        $W_i^1 := W_i^0 \oplus \Delta$;
        if $i \in \mathsf{CircuitIn}(f)$ then
            $e_i := W_i^0$;
    for $i \in \mathsf{GadgetOut}(f)\{\textit{in topo.order}\}$ do
        $\{a,b,c\} := \mathsf{GadgetInputs}(f,i)$;
        $(W_i^0, \vec{G}_i, \vec{z}_i) := \mathsf{Gb3GADGET}(W_a^0, W_b^0, W_c^0)$;
        $W_i^1 := W_i^0 \oplus \Delta$;
        $F_i := (\vec{G}_i, \vec{z}_i)$;
    for $l \in \mathsf{GadgetOut}(f)$ do
        $\{i,j,k\} := \mathsf{GadgetInputs}(f,l)$;
        if $i,k \in \mathsf{CircuitIn}(f)$ then
            continue;
        if $j \in \mathsf{CircuitIn}(f)$ then
            $\{a\} := \mathsf{IncomingWires}(f,i,j,k)$;
            $\mathsf{aux}_{\mathsf{xor}} := \{W_i^1, W_j^0, W_a^0 \oplus W_k^0\}$;
            $\mathsf{aux}_{\mathsf{and}} := \{W_a^0 \oplus W_i^0, W_j^0, W_k^0\}$;
            $\mathsf{aux}_{g,l} := \{\mathsf{aux}_{\mathsf{xor}}, \mathsf{aux}_{\mathsf{and}}\}$
        if $j \notin \mathsf{CircuitIn}(f)$ then
            $\{a,b\} := \mathsf{IncomingWires}(f,i,j,k)$;
            $\mathsf{aux}_{\mathsf{xor}} := \{W_i^1, W_b^0 \oplus W_j^0, W_a^0 \oplus W_k^0\}$;
            $\mathsf{aux}_{\mathsf{and}} := \{W_a^0 \oplus W_i^0, W_b^0 \oplus W_j^0, W_k^0\}$;
            $\mathsf{aux}_{g,l} := \{\mathsf{aux}_{\mathsf{xor}}, \mathsf{aux}_{\mathsf{and}}\}$
    for $i \in \mathsf{CircuitOut}(f), j \in \{0,1\}$ do
        $D_i^j := H'(W_i^0 \oplus \mathrm{lsb}(W_i)\Delta, i)$;
    return $(\hat{F}, \hat{e}, \hat{D}, \hat{\mathsf{aux}}_g)$;
procedure SelectGate($\hat{\mathsf{aux}}_g, f$):
    for $l \in \mathsf{GadgetIn}(f) \setminus \mathsf{CircuitIn}(f)$ do
        $\mathsf{aux}_{g,l} := \{\mathsf{aux}_{\mathsf{xor}}, \mathsf{aux}_{\mathsf{and}}\}$;
        if $l \in \mathsf{XORGates(f)}$ then
            $\mathsf{aux}_{e,l} = \mathsf{aux}_{\mathsf{xor}}$;
    else
        $\mathsf{aux}_{e,l} = \mathsf{aux}_{\mathsf{and}}$;
    return $\hat{\mathsf{aux}}_e$;

procedure Encode($\hat{e}, \hat{x}$):
    for $e_i \in \hat{e}$ do
        $X_i := e_i \oplus x_i\Delta$;
    return $\hat{X}$;
procedure Decode($\hat{D}, \hat{Y}$):
    for $D_i \in \hat{D}$ do
        if $\exists j \in \{0,1\}$ s.t. $D_i^j = H'(Y_i, i)$ : then
            $y_i := j \oplus \mathrm{lsb}(Y_i)$;
    return $\hat{Y}$;
procedure Eval($\hat{F}, \hat{X}, \hat{\mathsf{aux}}_e$):
    for $i \in \mathsf{GadgetIn}(\hat{F})$ do
        $W_i := X_i$;
    for $i \notin \mathsf{GadgetIn}(\hat{F})\{\textit{in topo.order}\}$ do
        $\{a,b,c\} := \mathsf{GadgetInputs}(\hat{F}, i)$;
        if $(a,b,c) \in \mathsf{CircuitIn}(f)$ then
            $E_i := \mathsf{Eval3GADGET}(F_i, W_a, W_b, W_c)$;
        else
            $\mathsf{aux}_{e,i} = \{\mathsf{aux}_a, \mathsf{aux}_b, \mathsf{aux}_c\}$;
            if $i \in \mathsf{XORGates(f)}$ then
                if $b \in \mathsf{CircuitIn}(f)$ then
                    $E_i := \mathsf{Eval3GADGET}(F_i, \mathsf{aux}_a, \mathsf{aux}_b, \mathsf{aux}_c \oplus W_c)$;
                else
                    $E_i := \mathsf{Eval3GADGET}(F_i, \mathsf{aux}_a, \mathsf{aux}_b \oplus W_b, \mathsf{aux}_c \oplus W_c)$;
            else
                if $b \in \mathsf{CircuitIn}(f)$ then
                    $E_i := \mathsf{Eval3GADGET}(F_i, \mathsf{aux}_a \oplus W_a, \mathsf{aux}_b, \mathsf{aux}_c)$;
                else
                    $E_i := \mathsf{Eval3GADGET}(F_i, \mathsf{aux}_a \oplus W_a, \mathsf{aux}_b \oplus W_b, \mathsf{aux}_c)$;
    Return $\hat{E}$;
```

Figure 8: Our Oblivious Garbling scheme

Proof Outline: As stated in Definition 4, we need to prove four properties the garbling scheme needs to satisfy.

Correctness: Correctness for the scheme follows from the correctness of the garbling scheme described for $\mathcal{G}_{\text{tri}}$ gadgets in Section 4.2 and the correctness of the soldering technique by [31] that ensures transfer of bit values from the active output wire labels of a gadget to the input wire labels of the subsequent gadget.

Privacy: We show in Section 4.2 a garbling scheme that can garble $\mathcal{G}_{\text{tri}}$ gadgets while satisfying the privacy definition. In this scheme, we additionally also need to simulate the auxiliary information aux_e. We build a simulator that does the following:

– Simulates the garbling of each $\mathcal{G}_{\text{tri}}$ gadget independently by sampling active labels uniformly at random.

– Computes the active output label, and the garbled gate.

– Computes soldering values for each wire connection as the XOR of the active label for the output wire of a given gadget and the active label for the input wire it is to be connected to in the circuit.

– Sets the auxiliary information aux_e according to the gate we are garbling in the original circuit.

Since each gadget is garbled independently, and the scheme used to garble them achieves privacy, the garbled gate and the input labels are indistinguishable from the actual garbled gates and input labels.

Obliviousness: The only difference between achieving obliviousness and privacy is that the simulator is not given $y = f(x)$ and is not expected to output the decoding information. Our simulator can be easily modified for this case, by only returning the fake garbled gates without the decoding information. This requires the simulator to not use y for creating the fake garbling, which is the case anyway.

Authenticity: For authenticity, we need to show that any PPT adversary $\mathcal{A}$ that receives as input $(\hat{F}, \hat{X}, \hat{D}, \mathsf{aux}_e)$ computed as in the real execution, cannot produce an inactive output label. That is, the adversary succeeds if it can get Decode to output $1 \oplus x_k$ for any output gate k with active value x_k. We have already shown in our proof for garbling a circuit with $\mathcal{G}_{\text{tri}}$ gadgets that this is negligible.

This concludes that the garbling scheme in Figure 8 satisfies Definition 4. $\square$

Computational Complexity: The garbling scheme makes 8 oracle calls to garble each $\mathcal{G}_{\text{tri}}$ gadget. In addition, we make OT calls linear in the circuit size. Therefore, for a circuit f of size $|f|$, we have the computational complexity of the scheme to be $\mathcal{O}(|f|)$. We observe that in spite of using the GRR3 optimization to garble the gadget, we garble each gadget independently and hence the garbling of each gadget can be done in parallel.

Communication Complexity: The garbling scheme generates a garbled gate of size $4\kappa/3 + \mathcal{O}(1)$ bits for each $\mathcal{G}_{\text{tri}}$ gadget. In addition, we also have OT communication linear in the circuit size to transfer all the soldering values. Therefore, for a circuit f of size $|f|$, we have the communication complexity of the scheme to be $\mathcal{O}(|f|)$. We stress that this is not a problem from a practical perspective, as one could use OT extensions for realising a high number of OTs.

Acknowledgements

Carmit Hazay and Rahul Satish were both partially supported by the United States-Israel Binational Science Foundation (BSF) through Grant No. 2020277 and by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Algorand Foundation. Rahul Satish was also supported by the Israeli Science Foundation (ISF) through Grant No. 2439/20.

References

1. Applebaum, B.: Key-dependent message security: Generic amplification and completeness. In: EUROCRYPT. pp. 527–546 (2011). https://doi.org/10.1007/978-3-642-20465-4_29
2. Applebaum, B.: Bootstrapping obfuscators via fast pseudorandom functions. In: ASIACRYPT. pp. 162–172 (2014). https://doi.org/10.1007/978-3-662-45608-8_9
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc^0. In: (FOCS. pp. 166–175 (2004). https://doi.org/10.1109/FOCS.2004.20
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Comput. Complex. **15**(2), 115–162 (2006). https://doi.org/10.1007/s00037-006-0211-8
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: ICALP. pp. 152–163 (2010). https://doi.org/10.1007/978-3-642-14165-2_14
6. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for boolean and arithmetic circuits. In: ACM CCS. p. 565–577 (2016). https://doi.org/10.1145/2976749.2978410
7. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: EUROCRYPT. pp. 423–444 (2010). https://doi.org/10.1007/978-3-642-13190-5_22

8. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: ACM. pp. 503–513 (1990). https://doi.org/10.1145/100216.100287
9. Bellare, M., Hoang, V.T., Rogaway, P.: Adaptively secure garbling with applications to one-time programs and secure outsourcing. In: ASIACRYPT. pp. 134–153 (2012). https://doi.org/10.1007/978-3-642-34961-4_10
10. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM CCS. pp. 784–796 (2012). https://doi.org/10.1145/2382196.2382279
11. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-round secure computation and secure autonomous mobile agents. In: ICALP. pp. 512–523 (2000). https://doi.org/10.1007/3-540-45022-X_43
12. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT. pp. 93–118 (2001). https://doi.org/10.1007/3-540-44987-6_7
13. Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.: On the security of the "free-xor" technique. In: TCC. pp. 39–53 (2012). https://doi.org/10.1007/978-3-642-28914-9_3
14. Döttling, N., Gajland, P., Malavolta, G.: Laconic function evaluation for turing machines. In: PKC. pp. 606–634 (2023). https://doi.org/10.1007/978-3-031-31371-4_21
15. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: STOC. pp. 554–563 (1994). https://doi.org/10.1145/195058.195408
16. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: Minilego: Efficient secure two-party computation from general assumptions. In: EUROCRYPT. pp. 537–556 (2013). https://doi.org/10.1007/978-3-642-38348-9_32
17. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: CRYPTO. pp. 465–482 (2010). https://doi.org/10.1007/978-3-642-14623-7_25
18. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: CRYPTO. pp. 39–56 (2008). https://doi.org/10.1007/978-3-540-85174-5_3
19. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: CRYPTO. pp. 162–179 (2012). https://doi.org/10.1007/978-3-642-32009-5_11
20. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: TCC. pp. 308–326 (2010). https://doi.org/10.1007/978-3-642-11799-2_19
21. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast garbling of circuits under standard assumptions. In: ACM CCS. p. 567–578 (2015). https://doi.org/10.1145/2810103.2813619
22. Guo, C., Katz, J., Wang, X., Yu, Y.: Efficient and secure multiparty computation from fixed-key block ciphers. In: IEEE S&P. pp. 825–841 (2020). https://doi.org/10.1109/SP40000.2020.00016
23. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS. pp. 294–304 (2000). https://doi.org/10.1109/SFCS.2000.892118
24. Kempka, C., Kikuchi, R., Suzuki, K.: How to circumvent the two-ciphertext lower bound for linear garbling schemes. In: ASIACRYPT. pp. 967–997 (2016). https://doi.org/10.1007/978-3-662-53890-6_32
25. Kiss, Á., Schneider, T.: Valiant's universal circuit is practical. In: EUROCRYPT. pp. 699–728 (2016). https://doi.org/10.1007/978-3-662-49890-3_27
26. Kolesnikov, V., Mohassel, P., Rosulek, M.: Flexor: Flexible garbling for XOR gates that beats free-xor. In: CRYPTO. pp. 440–457 (2014). https://doi.org/10.1007/978-3-662-44381-1_25
27. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: ICALP. pp. 486–498 (2008). https://doi.org/10.1007/978-3-540-70583-3_40
28. Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009). https://doi.org/10.1007/s00145-008-9036-8
29. Lipmaa, H., Mohassel, P., Sadeghian, S.S.: Valiant's universal circuit: Improvements, implementation, and applications. IACR Cryptol. ePrint Arch. p. 17 (2016), http://eprint.iacr.org/2016/017
30. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM-EC. pp. 129–139 (1999). https://doi.org/10.1145/336992.337028
31. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: TCC. pp. 368–386 (2009). https://doi.org/10.1007/978-3-642-00457-5_22
32. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: ASIACRYPT. pp. 250–267 (2009). https://doi.org/10.1007/978-3-642-10366-7_15
33. Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: IEEE FOCS. pp. 859–870 (2018). https://doi.org/10.1109/FOCS.2018.00086
34. Rosulek, M., Roy, L.: Three halves make a whole? beating the half-gates lower bound for garbled circuits. In: CRYPTO. pp. 94–124 (2021). https://doi.org/10.1007/978-3-030-84242-0_5
35. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: ACM CCS. pp. 463–472 (2010). https://doi.org/10.1145/1866307.1866359

36. Sander, T., Young, A.L., Yung, M.: Non-interactive cryptocomputing for nc^1. In: FOCS. pp. 554–567 (1999). https://doi.org/10.1109/SFFCS.1999.814630
37. Valiant, L.G.: Universal circuits (preliminary report). In: ACM STOC. pp. 196–203 (1976). https://doi.org/10.1145/800113.803649
38. Wang, Y., Malluhi, Q.M.: Reducing garbled circuit size while preserving circuit gate privacy. Cryptology ePrint Archive, Paper 2017/041 (2017), http://eprint.iacr.org/2017/041
39. Wegener, I.: The complexity of Boolean functions. Wiley-Teubner (1987), http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/
40. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: FoCS. pp. 162–167 (1986). https://doi.org/10.1109/SFCS.1986.25
41. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: EUROCRYPT. pp. 220–250 (2015). https://doi.org/10.1007/978-3-662-46803-6_8
42. Zhao, S., Yu, Y., Zhang, J., Liu, H.: Valiant's universal circuits revisited: an overall improvement and a lower bound. IACR Cryptol. ePrint Arch. p. 943 (2018), https://eprint.iacr.org/2018/943

Additional Supplementary Material

A Matrices used in the scheme

$$V = \begin{bmatrix}
1&0&0&0&0&0&0\\
0&1&0&0&0&0&0\\
0&0&1&0&0&0&0\\
1&0&0&0&0&0&1\\
0&1&0&0&0&1&1\\
0&0&1&0&0&1&1\\
1&0&0&0&1&0&1\\
0&1&0&0&1&0&1\\
0&0&1&1&0&0&1\\
1&0&0&0&1&0&0\\
0&1&0&0&1&1&0\\
0&0&1&1&0&1&0\\
1&0&0&1&0&0&1\\
0&1&0&0&0&0&1\\
0&0&1&0&0&0&1\\
1&0&0&1&0&0&0\\
0&1&0&0&0&1&0\\
0&0&1&0&0&1&0\\
1&0&0&1&1&0&0\\
0&1&0&0&1&0&0\\
0&0&1&1&0&0&0\\
1&0&0&1&1&0&1\\
0&1&0&0&1&1&1\\
0&0&1&1&0&1&1
\end{bmatrix}
\qquad
M = \begin{bmatrix}
1&0&1&0&0&0&1&0\\
0&0&1&0&1&0&1&0\\
1&0&0&0&1&0&1&0\\
1&0&1&0&0&0&0&1\\
0&0&1&0&0&1&0&1\\
1&0&0&0&0&1&0&1\\
1&0&0&1&0&0&0&1\\
0&0&0&1&1&0&0&1\\
1&0&0&0&1&0&0&1\\
1&0&0&1&0&0&1&0\\
0&0&0&1&0&1&1&0\\
1&0&0&0&0&1&1&0\\
0&1&1&0&0&0&0&1\\
0&0&1&0&1&0&0&1\\
0&1&0&0&1&0&0&1\\
0&1&1&0&0&0&1&0\\
0&0&1&0&0&1&1&0\\
0&1&0&0&0&1&1&0\\
0&1&0&1&0&0&1&0\\
0&0&0&1&1&0&1&0\\
0&1&0&0&1&0&1&0\\
0&1&0&1&0&0&0&1\\
0&0&0&1&0&1&0&1\\
0&1&0&0&0&1&0&1
\end{bmatrix}$$