Three Halves Make a Whole?
Beating the Half-Gates Lower Bound for Garbled Circuits

Mike Rosulek* Lawrence Roy*

June 3, 2021

Abstract

We describe a garbling scheme for boolean circuits, in which XOR gates are free and AND gates require communication of $1.5\kappa+5$ bits. This improves over the state-of-the-art "half-gates" scheme of Zahur, Rosulek, and Evans (Eurocrypt 2015), in which XOR gates are free and AND gates cost 2κ bits. The half-gates paper proved a lower bound of 2κ bits per AND gate, in a model that captured all known garbling techniques at the time. We bypass this lower bound with a novel technique that we call **slicing and dicing**, which involves slicing wire labels in half and operating separately on those halves. Ours is the first to bypass the lower bound while being fully compatible with free-XOR, making it a drop-in replacement for half-gates. Our construction is proven secure from a similar assumption to prior free-XOR garbling (circular correlation-robust hash), and uses only slightly more computation than half-gates.

1 Introduction

Garbled circuits (GC) were introduced by Yao in the 1980s [Yao82] in one of the first secure two-party computation protocols. They remain the leading technique for constant-round two-party computation. Garbled circuits exclusively use extremely efficient symmetric-key operations (*e.g.*, a few calls to AES per gate of the circuit), making communication rather than computation the bottleneck in realistic deployments — the parties must exchange $O(\kappa)$ bits per gate. For that reason, most improvements to garbled circuits have focused heavily on reducing their concrete size [BMR90, NPS99, KS08, PSSW09, KMR14, GLNP15]. The current state of the art for garbled (boolean) circuits is the *half-gates* construction of Zahur, Rosulek, and Evans [ZRE15]. In the half-gates scheme, AND gates are garbled with size 2κ bits, while XOR gates are free, requiring no communication.

The half-gates paper also establishes a lower bound for the size of garbled circuits. Specifically, the authors define a model of **linear garbling** — which captured all known techniques at the time — and proved that a garbled AND gate in this model requires 2κ bits. Thus, half-gates is optimal among linear garbling schemes. In response, there has been a line of work focused on finding ways around the lower bound. Several works [KKS16, BMR16, WmM17] were successful in constructing an AND gate using only κ bits, using techniques outside of the linear-garbling model. However, these constructions work *only for a single AND gate in isolation,* so they do not result in any improvement to half-gates for garbling general circuits.[1] Garbling an entire *arbitrary circuit* with

*Oregon State University, {rosulekm,royl}@oregonstate.edu. First author partially supported by NSF award #1617197. Second author supported by a DoE CSGF Fellowship.

[1] These constructions require the input labels to have a certain correlation that they do not guarantee for the gate's output labels.

less than 2κ bits per AND-gate remained an open problem. We discuss the linear garbling lower bound and different paths around it later in Section 7.

1.1 Our Results

We show a garbling scheme for general boolean circuits, in which XOR gates are free and AND gates cost only $1.5\kappa+5$ bits. This is the first scheme to successfully bypass the linear-garbling lower bound for all AND gates in a circuit, not just a single isolated AND gate. For the typical case of $\kappa = 128$ this is a concrete reduction of 23% in the size of garbled circuits relative to half-gates. Our construction compares to half-gates along other dimensions as follows:

- **Hardness assumption:** All free-XOR-based garbling schemes require a function H with output length κ and satsifying a *circular correlation-robust* property. In short, this means that terms of the form $H(X \oplus \Delta)$ and $H(X \oplus \Delta) \oplus \Delta$ are indistinguishable from random, for adversarially chosen X and global, secret Δ. Our construction requires a slight generalization. First, we require H that gives outputs of length $\kappa/2$. Second, the secret Δ is split into two halves $\Delta = \Delta_L \| \Delta_R$, and we require terms like $H(X \oplus \Delta) \oplus \Delta_L$, $H(X \oplus \Delta) \oplus \Delta_L \oplus \Delta_R$, etc. to be indistinguishable from random.

- **Computation:** Our scheme requires 50% more calls to H per AND gate than half-gates (6 vs 4 for the garbler, and 3 vs 2 for the evaluators). Similar to other work, we can instantiate the necessary H using just 1 call to AES with a key that is fixed for the entire circuit. As a result, the computational cost of our scheme is comparable to prior work.

  Additionally, since we require H with only $\kappa/2$ bits of output, certain queries to H for different AND-gates can be combined into a single query to a κ-bit-output function. The effect of this optimization depends on the circuit topology but in some cases our construction can have identical or better computation to half-gates (see Section 6.2).

We bypass the [ZRE15] lower bound by using two techniques that are outside of its linear-garbling model. We refer to the techniques collectively as **slicing-and-dicing.**

- **Slicing:** In our construction the evaluator **slices wire labels** into halves, and uses (*possibly different!*) linear combinations to compute each half. To the best of our knowledge, this technique is novel in garbled circuits. As we demonstrate in detail later, introducing more linear combinations for the evaluator increases the linear-algebraic dimension in which the scheme operates, in a way that lets us exploit more linear-algebraic structures that prior schemes could not exploit.

- **Dicing:** The evaluator first decrypts a constant-size ciphertext containing "**control bits**", which determine the linear combinations (of input label [halves], gate ciphertexts, and H-outputs) he/she will use to compute the output label [halves]. The control bits are chosen randomly by the garbler (*i.e.*, by tossing "dice") in a particular way. Randomized control bits are outside of the linear garbling model, which requires the evaluator's linear combinations to be *fixed.* This technique first appeared in [KKS16].

We also describe a variant of our scheme that can garble *any* kind of gate (*e.g.*, XOR gates, even constant-output gates) for $1.5\kappa + 10$ bits, in a way that hides the gate's truth table from the evaluator. This improves on the state of the art for *gate-hiding* garbling, due to Rosulek [Ros17], in which each gate is garbled for $2\kappa+8$ bits, and constant-output gates are not supported. Additionally, our gate-hiding construction is fully compatible with free-XOR, meaning that the circuit can contain

scheme	GC size (κ bits / gate) AND	XOR	calls to H per gate: garbler AND	garbler XOR	evaluator AND	evaluator XOR	assump.
unoptimized textbook Yao	8	8	4	4	2.5	2.5	PRF
Yao + point-permute [BMR90]	4	4	4	4	1	1	PRF
$4 \to 3$ row reduction [NPS99]	3	3	4	4	1	1	PRF
$4 \to 2$ row reduction [PSSW09]	2	2	4	4	1	1	PRF
free-XOR [KS08]	3	0	4	0	1	0	CCR
fleXOR [KMR14]	2	$\{0,1,2\}$	4	$\{0,2,4\}$	1	$\{0,1,2\}$	CCR
half-gates [ZRE15]	2	0	4	0	2	0	CCR
[GLNP15]	2	1	4	3	2	1.5	PRF
ours	**1.5**	**0**	≤ 6	0	≤ 3	0	CCR

Figure 1: Comparison of efficient garbling schemes. Gate size ignores small constant additive term (*i.e.*, "2" means $2\kappa + O(1)$ bits per gate). CCR = circular correlation robust hash function.

both "public" XOR gates (evaluator knows that this gate is an XOR) and "private" XOR gates (only the garbler knows that this gate is an XOR), with the public ones being free.

1.2 Related Work

The garbled circuits technique was first introduced by Yao [Yao82], although the first complete description and security proof for Yao's protocol was given much later [LP09]. Bellare, Hoang, and Rogaway [BHR12] promoted garbled circuits from a *technique* to well-defined cryptographic *primitive* with standardized security properties, which they dubbed a **garbling scheme.** In this work, we use their framework to formally express our schemes and prove security.

The garbling scheme formalization captures many techniques, but in this work we focus on "practical" GC techniques built from symmetric-key tools (PRFs, hash functions, but not homomorphic encryption or obfuscation). In the realm of practical garbling, there have been many quantitative and qualitative improvements over the years, especially focused on reducing the size of garbled circuits. These works are showcased in Figure 1. Of particular note are the Free-XOR technique of Kolesnikov & Schneider [KS08] and the half-gates consruction [ZRE15], mentioned above. Free-XOR allows XOR gates in the circuit to be garbled with no communication, and our construction inherits this technique to achieve the same feature. The free-XOR technique requires a cryptographic hash with a property called circular correlation-resistance [CKKZ12]. As mentioned above, the half-gates paper introduced a lower bound for garbling, which several works have bypassed in some limited manner. We discuss the lower bound and these related works in more detail in Section 7.

Several garbling schemes are tailored to support both AND and XOR gates while hiding the type of gate from the evaluator [KKS16, WmM17, Ros17]. These works are compared in Figure 2. They differ in the exact class of boolean gates they can support — all gates, all symmetric gates (satisfying $g(0,1) = g(1,0)$), or all non-constant gates.

2 Preliminaries

2.1 Circuits

We represent a circuit $f = (\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}, \mathsf{eval})$ by choosing a topological order of the $|f|$ inputs and gates in the circuit. Let inputs be the number of inputs in the circuit, which we require

scheme	GC size (κ bits/gate)	calls to H per gate garbler	evaluator	supported gates	assump.
Yao + point-permute [BMR90]	4	4	1	all	PRF
$4 \to 3$ row reduction [NPS99]	3	4	1	all	PRF
[KKS16]	2	3	1	symmetric	CCR
[WmM17]	2	3	1	symmetric	CCR
[Ros17]	2	4	1	non-const	PRF
ours	**1.5**	≤ 6	≤ 3	**all**	CCR

Figure 2: Comparison of **gate-hiding** garbling schemes, where the garbled circuit leaks only the topology of the circuit and not the type of each gate. Gate size ignores small constant additive term (*i.e.*, "2" means $2\kappa + O(1)$ bits per gate). CCR = circular correlation robust hash function. "Symmetric" means all gates g with $g(0,1) = g(1,0)$. "Non-const" means all gates g except $g(a,b) = 0$ and $g(a,b) = 1$.

to come first in the ordering. Each gate is then labeled by its index in the order. For every gate index g in the circuit, its two input indices[2] are $\mathsf{in}_1(g)$ and $\mathsf{in}_2(g)$, where $\mathsf{in}_i(g) < g$. Each gate can be evaluated using a function $\mathsf{eval}(g)\colon \{0,1\}^2 \to \{0,1\}$. Finally, the outputs are a subset of the indices $\mathsf{outputs} \subseteq [1, |f|]$.

Garbling only hides only partial information about the circuit. What is revealed is contained in the "leakage function" $\Phi(f)$. Sometimes two gates in a circuit may both be e.g. XOR-gates, but one will publicly be XOR while the operation performed by the other gate will be hidden. To support this, each gate is associated with some leakage $\mathsf{leak}(g)$. Gates with different leakages may compute the same function, but have different rules about how much information is revealed. We then define $\Phi(f)$ to be $(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak})$, containing the circuit topology and partial information about the gates' truth tables.

2.2 Garbling Schemes

We use a slightly modified version of the garbling definitions of [BHR12].

Definition 1. *A* garbling scheme *consists of four algorithms:*

- $(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$.
- $X := \mathsf{Encode}(e, x)$. *(deterministic)*
- $Y := \mathsf{Eval}(F, X)$. *(deterministic)*
- $y := \mathsf{Decode}(d, Y)$. *(deterministic)*

such that the following conditions hold.

Correctness: *For any circuit f and input x, after sampling $(F, e, d) \leftarrow \mathsf{Garble}(1^\kappa, f)$, $f(x) = \mathsf{Decode}(d, \mathsf{Eval}(\mathsf{Encode}(e, x)))$ holds with all but negligible probability.*

[2] We assume that all gates take two inputs. NOT gates can be merged them into downstream gates — *e.g.* if x goes into a NOT gate, and then into an AND gate with another input y, this is equivalent to a single $\overline{x} \wedge y$ gate.

Privacy with respect to leakage Φ: *There must be a simulator $\mathcal{S}$ such that for any circuit f and input x the following distributions are indistinguishable.*

$$\boxed{\begin{array}{l}(F,e,d) \leftarrow \mathsf{Garble}(1^\kappa, f)\\ X := \mathsf{Encode}(e,x)\\ \text{return } (F,X,d)\end{array}} \qquad \boxed{\begin{array}{l}(F,X,d) \leftarrow \mathcal{S}(1^\kappa, \Phi(f), f(x))\\ \text{return } (F,X,d)\end{array}}$$

Obliviousness w.r.t. leakage Φ: *There must be a simulator $\mathcal{S}$ such that for any circuit f and input x the following distributions are indistinguishable.*

$$\boxed{\begin{array}{l}(F,e,d) \leftarrow \mathsf{Garble}(1^\kappa, f)\\ X := \mathsf{Encode}(e,x)\\ \text{return } (F,X)\end{array}} \qquad \boxed{\begin{array}{l}(F,X) \leftarrow \mathcal{S}(1^\kappa, \Phi(f))\\ \text{return } (F,X)\end{array}}$$

Authenticity: *For any circuit f and input x, no PPT adversary $\mathcal{A}$ can make the following distribution output* TRUE *with non-negligible probability.*

$$\boxed{\begin{array}{l}(F,e,d) \leftarrow \mathsf{Garble}(1^\kappa, f)\\ X := \mathsf{Encode}(e,x)\\ Y \leftarrow \mathcal{A}(F,d,X)\\ \text{return } \mathsf{Decode}(d,Y) \notin \{f(x), \bot\}\end{array}}$$

The definitions differ from [BHR12] in two ways. First, we change correctness to allow a negligible failure probability.[3] Secondly, we strengthen the authenticity property by giving d to the adversary. This stronger property is easy to achieve by simply changing what one takes as garbled output Y.

2.3 Circular Correlation Robust Hashes

Our construction requires a hash function H with a property called *circular correlation robustness (CCR).* A comprehensive treatment of this property is presented in [CKKZ12, GKWY20].

The relevant definition of [GKWY20] is *tweakable CCR* (TCCR). For a hash function H, define a related oracle $\mathcal{O}_\Delta(X,\tau,b) = H(X \oplus \Delta, \tau) \oplus b\Delta$. Then H is a TCCR if $\mathcal{O}_\Delta$ is indistinguishable from a random oracle, provided that the distinguisher never repeats a (X,τ) pair in calls to the oracle.

We modify their definition in several important ways:

- We require H to have different input and output lengths. In the original definition, the adversary used the argument $b \in \{0,1\}$ to determine whether Δ was XOR'ed with the output of H. We generalize so that the adversary can choose a linear function of (the bits of) Δ that will be XOR'ed with the output of H. Our construction ultimately needs only 4 linear functions reflecting our slicing of wire labels in half: $L_{a,b}(\Delta_L \| \Delta_R) = a\Delta_L \oplus b\Delta_R$, for $a,b \in \{0,1\}$.

[3] Most garbling schemes actually do not have perfect correctness. If an output wire has labels W_0, W_1, then d will contain both $H(W_0)$ and $H(W_1)$. Correctness is violated if $H(W_0) = H(W_1)$.

- [GKWY20] observe that a "full" TCCR is stronger than what is needed for garbled circuits. In order to construct a TCCR that uses only one call to an ideal permutation, they prove TCCR security against adversaries that query only on "*naturally derived*" keys. It is somewhat cumbersome to generalize "*naturally derived*" keys to our setting, where the values are sliced into pieces.

  We instead relax TCCR so that H is drawn from a *family* of hashes, and the adversary only receives the description of H after making all of its oracle queries. This relaxation suffices for garbled circuits (the garbler chooses H and reveals it only in the garbled circuit description, after all queries to H have been made), and simplifies both our definition and our proof.

Definition 2. *A family of hash functions $\mathcal{H}$, where each $H \in \mathcal{H}$ maps $\{0,1\}^n \times \mathcal{T} \to \{0,1\}^m$ for some set of tweaks $\mathcal{T}$, is **randomized tweakable circular correlation robust (RTCCR)** for a set of linear functions $\mathcal{L}$ from $\{0,1\}^n$ to $\{0,1\}^m$ if, for any PPTs $\mathcal{A}_1, \mathcal{A}_2$ that never repeat an oracle query to $\mathcal{O}_{H,\Delta}$ on the same (X,τ),*

$$\left| \Pr_{H,\Delta}\left[v \leftarrow \mathcal{A}_1^{H,\mathcal{O}_{H,\Delta}}; \mathcal{A}_2(v,H) = 1\right] - \Pr_{H,R}\left[v \leftarrow \mathcal{A}_1^{H,R}; \mathcal{A}_2(v,H) = 1\right] \right|$$

is negligible, where R is a random oracle and $\mathcal{O}_{H,\Delta}$ is defined as

$$\boxed{\begin{array}{l}\mathcal{O}_{H,\Delta}(X \in \{0,1\}^n, \tau \in \mathcal{T}, L \in \mathcal{L}): \\ \hline \text{return } H(X \oplus \Delta, \tau) \oplus L(\Delta)\end{array}}$$

In Appendix A we show that if $F_k(X)$ is both a (plain) CCR hash for $\mathcal{L}$ when k is fixed and a PRF when k is random, and $\{(X,\tau) \mapsto X \oplus U(\tau) \mid U \in \mathcal{U}\}$ is a universal hash family,[4] then $\left\{(X,\tau) \mapsto F_k(X \oplus U(\tau)) \mid k \in \{0,1\}^\kappa, U \in \mathcal{U}\right\}$ is a secure RTCCR hash family for $\mathcal{L}$.

For our recommended instantiation, let σ be a simple function of the form $\sigma(X_L \| X_R) = \alpha X_L \| \alpha X_R$, where α is any fixed element in $\mathrm{GF}(2^{n/2}) \setminus \mathrm{GF}(2^2)$. Then $\mathsf{AES}_k(X) \oplus \sigma(X)$ is both a PRF for random k, and a CCR for any fixed k (modelling AES_k as an ideal permutation). Hence we get an RTCCR of the form:

$$(X,\tau) \mapsto \mathsf{AES}_k\Big(X \oplus U(\tau)\Big) \oplus \sigma(X \oplus U(\tau))$$

U can likewise be a simple function, *e.g.*, when $|\tau| \le \kappa/2$ then we can use $U(\tau) = u_1\tau \| u_2\tau$ where u_1, u_2 are random elements of $\mathrm{GF}(2^{\kappa/2})$.

3 A Linear-Algebraic View of Garbling Schemes

In this section we present a linear-algebraic perspective of garbling schemes, which is necessary to understand our construction and its novelty. This perspective is inspired by the presentation of Rosulek [Ros17], where the evaluator's behavior (in each of the 4 different gate-input combinations) defines a set of linear equations that the garbler must satisfy, and we rearrange those equations to isolate the values that are outside of the garbler's control.

[4] Equivalently, $\mathcal{U}$ is $2^{-\kappa}$-almost-XOR-universal (AXU).

3.1 The Basic Linear Perspective

Throughout this section, we consider an AND gate whose input wires have labels (A_0, A_1) and (B_0, B_1). We will always consider the free-XOR setting [KS08], where all wires have labels that xor to a common global Δ; *i.e.*, $A_0 \oplus A_1 = B_0 \oplus B_1 = \Delta$. Our view of garbling will always start with the circuit evaluator's perspective; hence we consider the subscripts to be public. In other words, if the evaluator holds A_i, then he knows the value i. In some works these subscripts are called "color bits" or "permute bits." The garbler secretly knows which of $\{A_0, A_1\}$ represent true and which of $\{B_0, B_1\}$ represent true.

Let's take an example of a textbook Yao garbled gate, using the point-permute technique. The garbled gate consists of 4 ciphertexts $G_{00}, \ldots, G_{11}$. When the evaluator has input labels A_i, B_j, he computes the output label by decrypting the (i, j)'th ciphertext, as $H(A_i, B_j) \oplus G_{ij}$.[5] In order to correspond to an AND gate, this evaluation expression must result in some label C (which could be either C_0 or C_1) representing (false) in 3 cases and $C \oplus \Delta$ (true) in the other. Suppose (A_1, B_0) is the case corresponding to inputs (true,true), then the garbler needs to arrange for:

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

We can rearrange these equations as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C \\ G_{00} \\ G_{01} \\ G_{10} \\ G_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H(A_0, B_0) \\ H(A_0, B_1) \\ H(A_1, B_0) \\ H(A_1, B_1) \end{bmatrix} \oplus \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{t} \Delta$$

In this equation, values that the garbler **cannot control** are on the right, and the results of the garbling process (gate ciphertexts and output labels) are on the left. The vector marked t is the truth table of the gate (when inputs are ordered by color bits), and known only to the garbler.

In order for the scheme to work, for all possible values on the right-hand side (including all choices of secret t!) the garbler must be able to solve for the variables on the left-hand side. In this case the left-hand side is under-determined so solving is easy. The garbler can simply choose random C and move it to the right-hand side. Then the matrix remaining on the left-hand side is an invertible identity matrix. Multiplying by the inverse solves for the desired values. Clearly this can be done for any t, meaning that this approach works to garble any gate (not just AND gates).

3.2 Row-Reduction Techniques

Row reduction refers to any technique to reduce the size of the garbled gate below 4 ciphertexts. The simplest method works by removing the ciphertext G_{00}, and simply having the evaluator take $H(A_0, B_0)$ as the output label when he has inputs A_0, B_0.

$$\begin{aligned} C &= H(A_0, B_0) \\ C &= H(A_0, B_1) \oplus G_{01} \\ C \oplus \Delta &= H(A_1, B_0) \oplus G_{10} \\ C &= H(A_1, B_1) \oplus G_{11} \end{aligned} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C \\ G_{01} \\ G_{10} \\ G_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H(A_0, B_0) \\ H(A_0, B_1) \\ H(A_1, B_0) \\ H(A_1, B_1) \end{bmatrix} \oplus \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{t} \Delta$$

The matrix on the left is now a square matrix, and invertible. Thus for any choice of t, the garbler can solve for C and the G_{ij} values by multiplying by the inverse matrix.

[5] For now, assume H is a random oracle. We ignore including the gate ID as an additional argument to H.

3.3 Half-Gates

The previous example shows that decreasing the size of the garbled gate from 4 to 3 causes the matrix on the left to change from size 4×5 to 4×4. Reducing the garbled gate further (from 3 ciphertexts to 2) would cause the matrix to be 4×3, and the system of linear equations would be overdetermined! So how does the half-gates garbling scheme [ZRE15] actually achieve a 2-ciphertext AND gate?

Let us recall the gate-evaluation algorithm for the half-gates scheme, which is considerably different from all previous schemes. On inputs A_i, B_j the evaluator computes the output label as $H(A_i) \oplus H(B_j) \oplus i \cdot G_0 \oplus j(G_1 \oplus A_i)$, where G_0, G_1 are the two gate ciphertexts.

Suppose as before that A_1 and B_0 correspond to true. Then the garbler must arrange for the following to be true:

$$\begin{aligned} C &= H(A_0) \oplus H(B_0) \\ C &= H(A_0) \oplus H(B_1) \qquad \oplus G_1 \oplus A_0 \\ C \oplus \Delta &= H(A_1) \oplus H(B_0) \oplus G_0 \\ C &= H(A_1) \oplus H(B_1) \oplus G_0 \oplus G_1 \oplus \underbrace{(A_0 \oplus \Delta)}_{A_1} \end{aligned}$$

Rearranging in our usual way, we get:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} C \\ G_0 \\ G_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \oplus \left[\begin{array}{ccc|c} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{array}\right. \left.\begin{array}{c} 0 \\ 0 \\ 1 \\ \underbrace{0}_{t} \end{array}\right] \right) \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \\ A_0 \\ \Delta \end{bmatrix}$$

Note that Δ is used both in the truth table adjustment (t) and in the usual operations of the evaluator (implicitly, in the one case where he includes $A_1 = A_0 \oplus \Delta$ in the linear combination).

As promised, the matrix on the left is only 4×3. We cannot solve for the left-hand side by inverting this matrix as in the previous cases. Instead, the garbler takes advantage of the fact that the **matrices on both sides have the same column space.** Specifically, the columns on the left span the space of all even-parity vectors. For any choice of t containing just a single 1 (corresponding to the truth table of an AND gate), every column on the right also has even parity! Concretely, suppose the evaluator solved the first three rows of this system of linear equalities (which is possible since the first three rows on the left form an invertible matrix), then the fourth row would automatically be in equality since on both sides it is the sum of the first 3 rows.[6] One can see that this technique works only for gates whose truth table has odd parity (*e.g.*, AND gates).

Half-gates was the first garbling scheme to structure its oracle queries as $H(A_i)$ and $H(B_j)$, instead of $H(A_i, B_j)$. Our linear-algebraic perspective highlights the importance of this change. For a 2-ciphertext AND gate, the matrix on the left will be 4×3, so the matrix on the right must have rank 3. An expression like $H(A_i, B_j)$ can be used by the evaluator in only one combination of inputs, leading to an identity matrix minor that has rank 4. By contrast, each $H(A_i)$ and $H(B_j)$ term is used for two input combinations, so the corresponding matrix can have rank 3.

Our linear algebraic perspective confirms and provides an explanation for a prior finding of Carmer & Rosulek [CR16]. They used a SAT solver to show that no garbling scheme (in the linear

[6] More generally, multiplying by a **left-inverse** of the matrix on the left-hand side "just works," as in the case where the matrix on the left-hand side is invertible.

model of the half-gates paper) could achieve a 2-ciphertext AND gate, when the evaluator makes only one query to H. This reiterates the importance of half gates using $H(A), H(B)$ oracle queries to achieve a 2-ciphertext AND gate.

4 High-Level Overview of Our Scheme

In the previous section, we saw that it was important that the evaluator used oracle queries like $H(A_i)$ and $H(B_j)$ in the half-gates scheme. For every term of the form $H(A_i)$ there are two gate-input combinations in which the evaluator uses this term. This property led to a desirable redundancy in the matrix that relates H-queries to input combinations. Redundancies in this matrix lead to smaller garbled gates. We push this idea further using several key observations.

4.1 Observation #1: Get the Most out of the Oracle Queries

$H(A_i)$ and $H(B_j)$ are not the only oracle queries that can be made in two different gate-input combinations. We can also ask the evaluator to query $H(A_i \oplus B_j)$. Because of the free-XOR constraint, $A_0 \oplus B_0 = A_1 \oplus B_1$, and $A_0 \oplus B_1 = A_1 \oplus B_0$. This means that the following oracle queries can be made for each gate-input combination:

	$H(A_0)$	$H(A_1)$	$H(B_0)$	$H(B_1)$	$H(A_0 \oplus B_0)$	$H(A_0 \oplus B_1)$
gate input (0,0)	✓		✓		✓	
gate input (0,1)	✓			✓		✓
gate input (1,0)		✓	✓			✓
gate input (1,1)		✓		✓	✓	

(1)

Can we use queries of this form to introduce even more redundancy in the relevant matrices?

4.2 Observation #2: Increase Dimension by Slicing Wire Labels

Our linear-algebraic perspective of garbling includes only 4 linear equations, corresponding to the 4 different gate-inputs. Having only 4 linear equations makes it difficult to take advantage of any new structure introduced by observation #1. Our second observation, and perhaps the key to our entire approach, is to **split each wire label into a left and right half**, and let the evaluator compute the two halves (of the output label) with different linear combinations. This results in 8 linear equations in our linear-algebraic perspective — 2 equations for each of the 4 gate-input combinations.

Consider the following proposal,

	$H(A_0)$	$H(A_1)$	$H(B_0)$	$H(B_1)$	$H(A_0 \oplus B_0)$	$H(A_0 \oplus B_1)$
(0,0) left	✓				✓	
(0,0) right			✓		✓	
(0,1) left	✓					✓
(0,1) right				✓		✓
(1,0) left		✓				✓
(1,0) right			✓			✓
(1,1) left		✓			✓	
(1,1) right				✓	✓	

(2)

For example, on gate-input (0,0) the evaluator will compute the left half of the output label as $H(A_0) \oplus H(A_0 \oplus B_0) \oplus \cdots$ (plus other terms, involving gate ciphertexts and input labels). There are several important features of this table to note:

- $H(\cdot)$ is used in a linear equation to compute half of an output label, therefore $H(\cdot)$ is a function with $\kappa/2$ bits of output. Three of these half-sized hash functions are combined to encrypt the gate output.[7] However, we still will use the *entire* input wire labels as input to H — using wire-label halves as input to H would cut the effective security parameter in half.

- For an evaluator with gate-input (0,0), the values $H(A_1)$, $H(B_1)$, and $H(A_0 \oplus B_1)$ are all jointly indistinguishable from random. With that in mind, consider the linear combinations for any other gate-input. For example, in the (1,0) case the evaluator will compute the output as

$$\begin{aligned} \text{left} &= H(A_1) \oplus H(A_0 \oplus B_1) \oplus \cdots \\ \text{right} &= H(B_0) \oplus H(A_0 \oplus B_1) \oplus \cdots \end{aligned}$$

  Because $H(A_1)$ and $H(A_0 \oplus B_1)$ are pseudorandom, this makes both of these outputs *jointly* pseudorandom. The entire output of the (1,0) case is pseudorandom from the perspective of the evaluator in the (0,0) case. This is a necessary condition, since sometimes the (0,0) and (1,0) cases give different outputs. This pattern holds with respect to any pair of two gate-inputs.

- If we interpret Equation 2 as a matrix (✓=1, empty cell=0), we see that it has rank 5. This suggests that the garbling process can result in only 5 output values, where in this case each of these values is $\kappa/2$ bits. Two of the values are the halves of the output wire label C, leaving 3 values to comprise the garbled gate ciphertexts. In other words, we are on our way to a garbled gate with only $3\kappa/2$ bits, if only we can get all of the relevant linear equations to cooperate.

4.3 Observation #3: Randomize and Hide the Evaluator's Coefficients

Let us apply our observations so far to our linear perspective of Section 3. Since wire labels are divided into halves, we use notation like A_{0R} to denote the right half of A_0. Note that the free-XOR constraint applies independently to the wire label halves; *i.e.*, $A_{1R} = A_{0R} \oplus \Delta_R$ and so on.

The evaluator computes each half of the output label separately, using a linear combination of available information: oracle responses, gate ciphertexts, and the 4 (!) halves of the input labels. If we account for all 8 of the evaluator's linear equations, while using the oracle-query structure suggested in Equation 2, we obtain the following system:

$$\left[\begin{array}{ccccc} 1 & 0 & ? & ? & ? \\ 0 & 1 & ? & ? & ? \\ \hline 1 & 0 & ? & ? & ? \\ 0 & 1 & ? & ? & ? \\ \hline 1 & 0 & ? & ? & ? \\ 0 & 1 & ? & ? & ? \\ \hline 1 & 0 & ? & ? & ? \\ 0 & 1 & ? & ? & ? \end{array}\right] \left[\begin{array}{c} C_L \\ C_R \\ G_0 \\ G_1 \\ G_2 \end{array}\right] = \left(\left[\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & ? & ? & ? & ? & ? & ? \\ 0 & 0 & 1 & 0 & 1 & 0 & ? & ? & ? & ? & ? & ? \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & ? & ? & ? & ? & ? & ? \\ 0 & 0 & 0 & 1 & 0 & 1 & ? & ? & ? & ? & ? & ? \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & ? & ? & ? & ? & ? & ? \\ 0 & 0 & 1 & 0 & 0 & 1 & ? & ? & ? & ? & ? & ? \\ \hline 0 & 1 & 0 & 0 & 1 & 0 & ? & ? & ? & ? & ? & ? \\ 0 & 0 & 0 & 1 & 1 & 0 & ? & ? & ? & ? & ? & ? \end{array}\right] \oplus \left[\begin{array}{c|c} 0 \cdots\cdots 0 & \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \\ \vdots \quad\quad \vdots & \vdots \\ \hline \vdots \quad\quad \vdots & \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \\ \hline \vdots \quad\quad \vdots & \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \\ \hline 0 \cdots\cdots 0 & \underbrace{\begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array}}_{t} \end{array}\right] \right) \left[\begin{array}{c} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \\ A_{0L} \\ A_{0R} \\ B_{0L} \\ B_{0R} \\ \Delta_L \\ \Delta_R \end{array}\right] \tag{3}$$

[7] Hence the title: "Three Halves Make a Whole".

The first row represents the evaluator's linear equation to compute the *left half* C_L of the output label on input A_0, B_0, etc. Note that the truth table t now consists of 2×2 identity blocks and 2×2 zero-blocks.

For everything to work correctly, we need to replace the "?" entries, so that for every choice of t, the matrices on both sides have the same column space.

- The columns on the right-hand side (representing the H outputs) already span a space of dimension 5, so there is no choice but to extend the left-hand side matrix to a basis of that space.
- The "?" entries on the right are subject to other constraints, so that they reflect what an evaluator can actually do in each input combination. For example, on input A_0, B_1, the evaluator cannot include B_{0R} in its linear combination, it can only include $B_{1R} = B_{0R} \oplus \Delta_R$. Note that the matrix is written in terms of B_0 only.

Unfortunately, it is not possible to complete the right-hand-side matrix subject to these constraints. **For every t, there is a valid way to replace the "?" entries**, but there is no one way that works for *all* t.

To get around this problem, we **randomize and encrypt the entries of the matrix**. To the best of our knowledge, the technique first appeared in the garbling scheme of [KKS16], and was also used in [WmM17, Ros17]. The garbler will complete the matrices so that the system of equations can be solved (*i.e.*, the column spaces coincide). This causes the matrix entries to now depend on the garbler's secret t. Next, the garbler will **encrypt these matrix entries**, so that when the evaluator has input A_i, B_j, he can decrypt *only those matrix entries needed for that particular input combination* — not the entire matrix. For example, the evaluator can use A_0, B_0 to decrypt the top two rows of the matrix — just enough to determine the coefficients of the linear combinations computing the output label. Unlike other schemes, there is a step of indirection (decrypting this additional ciphertext) before the evaluator determines which linear combinations to apply — the linear combination does not depend solely on the color bits of the input labels. We call the contents of these ciphertexts **control bits**, which tell the evaluator what linear combination to apply. The control bits are of small constant size, so encrypting them adds only a constant number of bits to the garbling scheme.

The garbler completes the missing entries in the matrix by drawing them randomly from a *distribution* over matrices. The distribution depends on t, as we mentioned — however, it can be arranged that **each marginal view of the matrix is independent of t.** Since the evaluator sees only such a marginal view, not the entire matrix, the value of t is hidden.

5 Details: Slicing & Dicing

In this section we complete the full picture of our construction. We direct the reader to a **guide to notation/symbols** in Figure 9.

5.1 Choosing the Matrices

Let us begin by filling out the question marks in Equation 3. We rewrite this equation using block matrices, and we group related parts together.

$$V \begin{bmatrix} C \\ \vec{G} \end{bmatrix} = M\vec{H} \oplus \left(R \oplus \left[0 \cdots 0 \,\middle|\, t\right]\right) \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix} \tag{4}$$

Here C, A_0, B_0, and Δ are two-element (column) vectors representing the two halves of these wire labels; $\vec{G}$ is the vector of gate ciphertexts; and $\vec{H} = \big[H(A_0)\ H(A_1)\ H(B_0)\ H(B_1)\ H(A_0 \oplus B_0)\ H(A_0 \oplus B_1)\big]^\top$ is the vector of H-outputs. t is the 8×2 truth table matrix, which contains a 2×2 identity matrix block for each case of the gate that should output true. We have already filled out M — it is the portion of the right-hand side matrix in Equation 3 with no question marks, that operates on the hash outputs $\vec{H}$. R is called the **control matrix** because it determines which pieces of input labels are added to the output.

Choosing V. Recall that the matrices on both sides of the equation must have the same column space, and that M already spans this 5-dimensional space. Call this common column space the *gate space* $\mathcal{G}$. Then

$$\mathcal{G} = \text{colspace}(V) = \text{colspace}(M) \supseteq \text{colspace}\big(R \oplus \big[0 \cdots 0 \,\big|\, t\big]\big).$$

It will be more convenient to represent $\mathcal{G}$ using linear constraints, rather than as the span of the columns of M. We use a matrix K as a basis for the cokernel of M, so that any vector v is in $\mathcal{G}$ if and only if $Kv = 0$. Then V must satisfy $\text{rank}(V) = 5$ and $KV = 0$.

Any K and V satisfying these constraints will suffice, and we will use the following:

$$K = \left[\begin{array}{cc|cc|cc|cc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right] \qquad V = \left[\begin{array}{c} 1\ 0\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0 \\ 0\ 1\ 0\ 1\ 0 \end{array}\right]$$

Note that the columns of V corresponding to the gate ciphertexts (the 3 rightmost columns) are the same as the columns in M corresponding to hash outputs $H(A_1), H(B_1), H(A_0 \oplus B_1)$, so they are clearly in the column space of M.

Constraints on choosing R. It remains to see how we choose the control matrix R. Using our new notation, $\text{colspace}\big(R \oplus \big[0 \cdots 0 \,\big|\, t\big]\big) \subseteq \mathcal{G}$ is equivalent to $KR = K\big[0 \cdots 0 \,\big|\, t\big]$, so we must choose R to match Kt. Because t is composed of 2×2 zero or identity blocks, we can deduce:

$$KR = K\big[0 \cdots 0 \,\big|\, t\big] = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 0 & 0 & p \\ 0 & 0 & 0 & 0 & a & b \end{array}\right] \tag{5}$$

for some $a, b \in \{0, 1\}$, where p is the parity of the truth table. In our main construction, $p = 1$ since it only considers garbling AND gates. However, the bits a, b reveal more than the parity of the gate — they leak the position of the "1" in the truth table. Since R must depend on these a, b bits, we resort to randomizing the control matrix R to hide a, b.

We also need the control matrix to reflect linear combinations that the evaluator can actually do with the available wire labels. The linear constraints are expressed in terms of A_0, B_0, and Δ, but when the evaluator has wire label, say, A_1, he can either include it in the linear combination (adding both A_0 and Δ) or not (adding neither A_0 nor Δ) — he cannot include only one of A_0, Δ

in the linear combination. This means that R must decompose into 2×2 matrices in the following way:

$$R = \begin{bmatrix} R_{00A} & R_{00B} & 0 \\ R_{01A} & R_{01B} & R_{01B} \\ R_{10A} & R_{10B} & R_{10A} \\ R_{11A} & R_{11B} & R_{11A} \oplus R_{11B} \end{bmatrix} \tag{6}$$

When the evaluator holds input labels A_i, B_j, the submatrix $R_{ij} = \begin{bmatrix} R_{ijA} & R_{ijB} \end{bmatrix}$ is enough to completely determine which linear combination should be applied. We call R_{ij} the **marginal view** for that input combination. We will randomize the choice of R, subject to the constraints listed above, so that any single marginal view leaks nothing about t. That is, we want to find a distribution $\mathcal{R}(t)$ such that when $R \leftarrow \mathcal{R}(t)$, $KR = K\left[0 \cdots 0 \,\middle|\, t\right]$ with probability 1, yet for every $i, j \in \{0,1\}$, if $t \leftarrow T$ and $R \leftarrow \mathcal{R}(t)$ then t and R_{ij} are independently distributed.

Basic approach to the distribution $\mathcal{R}(t)$: We must choose R to match the p, a, b bits defined above (which depend on the truth table t). Suppose we have a distribution $\mathcal{R}_0$ with the following properties:

- If $R_\$ \leftarrow \mathcal{R}_0$ then $KR_\$ = 0$
- For all $i, j \in \{0,1\}$, if $R_\$ \leftarrow \mathcal{R}_0$ then $(R_\$)_{ij}$ (the marginal view) is uniform

and we also have fixed matrices R_p, R_a, R_b such that:

$$KR_p = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \quad KR_a = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \quad KR_b = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right], \tag{7}$$

Define $\mathcal{R}(t)$ to first sample $R_\$ \leftarrow \mathcal{R}_0$ and output $R = pR_p \oplus aR_a \oplus bR_b \oplus R_\$$. The result R will always satisfy the condition of Equation 5. The randomness in $R_\$$ also causes marginal views of R_{ij} to be uniform and therefore hide p, a, b. See Appendix B for details of sampling $R_\$$. Concrete values for R_p, R_a, R_b are given in Figures 3 and 4, as part of a different construction.

If $\mathcal{R}_0$ is the *uniform distribution* over all matrices satisfying $KR = 0$, then the garbler must encrypt the full marginal views R_{ij} at 8 bits per view. A more thoughtful choice of distribution will allow the garbler to convey R_{ij} marginal views with fewer bits.

Compressing the marginal views: Each marginal view R_{ij} is a 2×4 matrix. We can "compress" these if we manage to restrict all R_{ij} to some linear subspace $S = \text{span}\{S_1, S_2, \ldots, S_d\}$ of 2×4 matrices (presumably with dimension $d < 8$), while still maintaining the other properties needed.

Let $\bar{R}_{ij}$ denote the representation of R_{ij} with respect to the basis S — *i.e.*, a vector of length d. Then the garbler can encrypt only the $\bar{R}_{ij}$'s to convey the marginal views of R. The choice of the subspace S depends on the class of truth tables that need to be hidden.

Parity-leaking gates: We performed an exhaustive computer search of low dimensional subspaces to determine how to pick the basis S for different types of gates. For even-parity gates (e.g. XOR or constant gates) we found a 2-dimensional subspace that works. Details of the $\mathcal{R}(t)$ distribution are given in Figure 3. For odd-parity gates (like AND, OR) we simply use the even-parity distribution and add a public constant R_p (from Figure 4) to the result. This approach works

$$S_1 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \qquad S_2 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$\bar{R}_a = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \bar{R}_b = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad \bar{R}_\$ \leftarrow \text{span}\left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}$$

$$R_a = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \qquad R_b = \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$R_\$ \leftarrow \text{span}\left\{ \left[\begin{array}{cc|cc|cc} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{array}\right], \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right] \right\}$$

Figure 3: Control matrices for even-parity gates. The top row contains the two basis matrices for S. The bottom row shows the full control matrices (R_p is not needed for even-parity gates). The middle row shows the "compressed" representation of the control matrices, in terms of the basis $\{S_1, S_2\}$ (*i.e.*, each row expresses which linear combination of S_1, S_2 appears in the corresponding blocks of the control matrix). The reader can verify that (1) each row in $\bar{R}_\$$ is individually uniform; (2) $KR_\$ = 0$; and (3) Equation 7 holds.

$$S_1 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \quad S_2 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right] \quad S_3 = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \quad S_4 = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

$$\bar{R}_p = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \bar{R}_\$ \leftarrow \text{span}\left\{ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \ldots \right\}$$

$$R_p = \left[\begin{array}{cc|cc|cc} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$R_\$ \leftarrow \text{span}\left\{ \left[\begin{array}{cc|cc|cc} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{array}\right], \left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right], \ldots \right\}$$

Figure 4: Control matrices for gate-hiding garbling. The top row contains the basis matrices for S. The basis of Figure 3 is a subset of this basis, so we can use the same R_a and R_b as Figure 3. The distributions on $\bar{R}_\$$ and $R_\$$ also include the matrices from Figure 3 (omitted with "..." here). The middle row gives the control matrices in terms of the new basis, while the bottom row shows them directly. The reader may verify that (1) each row of $\bar{R}_\$$ is individually uniform; (2) $KR_\$ = 0$; and (3) Equation 7 holds.

when the parity of the gate is public, since the evaluator must know to add R_p when decoding the description of their marginal view R_{ij}.

The construction for odd-parity gates is our primary construction, which would be used in most applications of garbling (in combination with free XOR gates).

Parity-hiding gates: To make the garbling scheme gate-hiding, we also need to hide the parity of the truth table. In other words, the distribution on $R_\$$ must be random enough to mask the presence (or absence) of a matrix R_p as in Equation 7. The R_p in Figure 4 is not in the subspace S of control matrices in Figure 3. Hence, to support parity-hiding we have had to extend that subspace with two additional basis elements (the basis matrices S_1, S_2 are as in the parity-leaking case). Our parity-hiding gates require 4 (compressed) control bits per gate-input combination, corresponding to the 4-dimensional basis S. See Figure 4 for details.

5.2 Garbling the Control Bits

So far we have glossed over the details of how the control bits actually get encrypted and sent to the evaluator. We know that there will be some $4 \times d$ ($d = 2$ for parity-leaking gates and $d = 4$ for parity hiding gates) matrix $\bar{R}$, and that the evaluator should only get to see a single row $\bar{R}_{ij}$ of $\bar{R}$ telling them what linear combination of $S_1, \ldots, S_d$ to use as control bits. The garbler can easily encrypt these values so that on input A_i, B_j the evaluator can decrypt only $\bar{R}_{ij}$.

In order to reuse the calls to H that the evaluator already uses, it turns out that we can use our new garbling construction to *garble the control bits as well.* At first it looks like this would just give infinite recursion, as if we used something like Equation 4 to garble the control bits then that garbling would need its own control bits, which would need to be garbled, and so on. In reality, the compressed control bits actually have a structure that allows us to garble them without recursive control bits.

Conceptually, we can treat the bits of $\bar{R}$ as wire labels and slice them as we do regular wire labels. Collect the bits from odd and even-indexed positions of $\bar{R}_{ij}$ into numbers $\overline{r}_{ijL}$ and $\overline{r}_{ijR} \in \mathrm{GF}(2^{d/2})$, respectively. Define the vector

$$\vec{r} = \left[\overline{r}_{00L}\ \overline{r}_{00R}\ \overline{r}_{01L}\ \overline{r}_{01R}\ \overline{r}_{10L}\ \overline{r}_{10R}\ \overline{r}_{11L}\ \overline{r}_{11R}\right]^\top$$

We observed that for both our parity-leaking and parity-hiding constructions, this vector is always in the gate subspace $\mathcal{G}$ — *i.e.*, that $K\vec{r} = 0$. Looking at Figure 3, the reader can check that this holds for any possible $\vec{r}$ (which in this case is the same as $\bar{R}$ read in row-major order). And similarly for Figure 4; this time the test for $\bar{R}$ is equivalent to checking its two 4×2 blocks individually.

Since the control bits, when expressed as $\vec{r}$, are always in the gate subspace $\mathcal{G}$, they can be garbled without needing their own control bits. The garbler can compute a constant-size ciphertext $\vec{z}$ such that:

$$V\vec{z} \oplus M\,\mathrm{lsb}_{\frac{d}{2}}(\vec{H}) = \vec{r}, \tag{8}$$

where $V, M, \vec{H}$ are as in Equation 4. Here we assume that every hash has been extended by an extra $d/2$ bits (or more realistically given that block ciphers have a fixed size, each wire label slice has been shrunk by $d/2$ bits to make room), and that these extra bit can be extracted with $\mathrm{lsb}_{\frac{d}{2}}$. The remainder of the hash vector, $\mathrm{msb}_{\frac{\kappa}{2}}(\vec{H})$, is used for garbling the wire labels themselves. By the same reasoning as for usual garbling, when the evaluator has input labels A_i, B_j, he can learn only the $\vec{r}_{ij}$ portions of $\vec{r}$.

We can combine Equations 4 and 8 into a single system, allowing the whole gate to be garbled at once.

$$V\left(\vec{z}\,\middle\|\,\begin{bmatrix}C\\ \vec{G}\end{bmatrix}\right)\oplus M\vec{H}=\vec{r}\,\middle\|\left(\left(R\oplus\left[0\cdots0\,\middle|\,t\right]\right)\begin{bmatrix}A_0\\ B_0\\ \Delta\end{bmatrix}\right), \tag{9}$$

where $\|$ denotes element wise concatenation, so e.g. the bits of $\overline{r}_{00L}\in \mathrm{GF}(2^{d/2})$ get concatenated with some $x\in\mathrm{GF}(2^{\kappa/2})$ to get a value in $\mathrm{GF}(2^{(\kappa+d)/2})$. We write the bits in little endian order, so $\mathrm{lsb}_{\frac{d}{2}}(\vec{H})\,\|\,\mathrm{msb}_{\frac{\kappa}{2}}(\vec{H})=\vec{H}$.

5.3 The Construction

We can now describe our garbling scheme formally. All of our different types of gates are compatible, so we describe a single unified scheme. The circuit has a leak function that indicates what information about each gate is public (which affects the cost of garbling each gate):

- EVEN: even-parity gate
- ODD: odd-parity gate
- XOR: free XOR gate
- NONE: no leakage (gate-hiding)

Because we need different control matrices depending on what kind of gate is being garbled, we use the notation $\mathcal{R}(L,t)$, for $L\in\{\mathsf{EVEN},\mathsf{ODD},\mathsf{NONE}\}$ to denote the appropriate distribution over control matrices. For EVEN/ODD gates, the distribution is as in Figure 3 (with R_p added in the case of ODD), and for NONE the distribution is as in Figure 4.

Our garbling scheme is shown in Figures 5 and 6. The garbler associates the kth wire in the circuit with a wire label W_k (and its opposite label $W_k\oplus\Delta$) and a point-and-permute bit π_k. W_k is the label with color bit $\mathrm{lsb}(W_k)=0$ (visible to the evaluator). The label $W_k\oplus\pi_k\Delta$ is the wire label representing `false` on that wire. Equivalently, W_k is the wire label representing logical value π_k.

SampleR(t,leak):
  $R\leftarrow\mathcal{R}(\mathsf{leak},t)$
  for $i,j\in\{0,1\}$:
    find coeffs c s.t. $R_{ij}=\bigoplus_k c_k\,S_k(\mathsf{leak})$
    $\overline{r}_{ijL}:=c_1\,\|\cdots\|\,c_{d-1}$ // *odd positions*
    $\overline{r}_{ijR}:=c_2\,\|\cdots\|\,c_d$ // *even positions*
  $\vec{r}=\left[\overline{r}_{00L}\;\overline{r}_{00R}\;\overline{r}_{01L}\;\overline{r}_{01R}\;\overline{r}_{10L}\;\overline{r}_{10R}\;\overline{r}_{11L}\;\overline{r}_{11R}\right]^\top$
  if leak = ODD:
    $R:=R\oplus R_p(\mathsf{ODD})$
  return $R,\vec{r}$

Encode$((\Delta,W,\pi),x)$:
  for $k=1$ to inputs:
    $E_k:=W_k\oplus(x_k\oplus\pi_k)\Delta$
  return E

DecodeR$(\vec{r},\mathsf{leak},i,j)$:
  $\begin{bmatrix}c_1\,\|\cdots\|\,c_{d-1}\\ c_2\,\|\cdots\|\,c_d\end{bmatrix}:=\vec{r}$
  $R_{ij}:=\bigoplus_k c_k\,S_k(\mathsf{leak})$
  if leak = ODD:
    $R_{ij}:=R_{ij}\oplus(R_p(\mathsf{ODD}))_{ij}$
  return R_{ij}

Decode$((\Phi,D),E)$:
  $(\mathsf{inputs},\mathsf{outputs},\mathsf{in},\mathsf{leak}):=\Phi$
  $y:=$ empty list
  for $k\in$ outputs:
    if $\exists j.\,D_k^j=H'(E_k,k)$:
      append j to y
    else: abort
  return y

Figure 5: Our garbling scheme (continued in Figure 6).

$\underline{\mathsf{Garble}(1^\kappa, f)}$:

$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}, \mathsf{eval}) := f$

$H \leftarrow \mathcal{H}$

$\Delta \leftarrow \begin{bmatrix} 1 \parallel \mathrm{GF}(2^{\kappa/2-1}) \\ \mathrm{GF}(2^{\kappa/2}) \end{bmatrix}$

for $k = 1$ to inputs:

$W_k \leftarrow \begin{bmatrix} 0 \parallel \mathrm{GF}(2^{\kappa/2-1}) \\ \mathrm{GF}(2^{\kappa/2}) \end{bmatrix}$

$\pi_k \leftarrow \{0,1\}$

for $k = \mathsf{inputs} + 1$ to $|f|$:

$A_0, B_0 := W_{\mathsf{in}_1(k)}, W_{\mathsf{in}_2(k)}$

$\pi_A, \pi_B := \pi_{\mathsf{in}_1(k)}, \pi_{\mathsf{in}_2(k)}$

if $\mathsf{leak}(k) = \mathsf{XOR}$:

$W_k := A_0 \oplus B_0$

$\pi_k := \pi_A \oplus \pi_B$

continue

$g := \mathsf{eval}(k)$

$t := \begin{bmatrix} g(\pi_A, \pi_B) & g(\pi_A, 1-\pi_B) & g(1-\pi_A, \pi_B) & g(1-\pi_A, 1-\pi_B) \end{bmatrix}^\top$

$R, \vec{r} := \mathsf{SampleR}(t, \mathsf{leak}(k))$

$$\vec{z}_k \parallel \begin{bmatrix} C \\ \vec{G}_k \end{bmatrix} := V^{-1} \left(\vec{r} \parallel \left(R \oplus \begin{bmatrix} 0 \cdots 0 \,\middle|\, t \end{bmatrix} \right) \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix} \right) \oplus V^{-1} M \begin{bmatrix} H(A_0, 3k-3) \\ H(A_0 \oplus \Delta, 3k-3) \\ H(B_0, 3k-2) \\ H(B_0 \oplus \Delta, 3k-2) \\ H(A_0 \oplus B_0, 3k-1) \\ H(A_0 \oplus B_0 \oplus \Delta, 3k-1) \end{bmatrix}$$

$\pi_k := \mathrm{lsb}(C)$

$W_k := C \oplus \pi_k \Delta$

for $k \in \mathsf{outputs}$, $j \in \{0,1\}$:

$D_k^j := H'(W_k \oplus (j \oplus \pi_k)\Delta, k)$

return $F = (\Phi(f), H, \vec{G}, \vec{z}),\ e = (\Delta, W, \pi),\ d = (\Phi(f), D)$

$\underline{\mathsf{Eval}(F = (\Phi, H, \vec{G}, \vec{z}), E)}$:

$(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}) := \Phi$

for $k = \mathsf{inputs} + 1$ to $|\Phi|$:

$A, B := E_{\mathsf{in}_1(k)}, E_{\mathsf{in}_2(k)}$

$i, j := \mathrm{lsb}(A), \mathrm{lsb}(B)$

if $\mathsf{leak}(k) = \mathsf{XOR}$:

$E_k := A \oplus B$

else

$$\vec{r} \parallel X_{ij} := V_{ij} \left(\vec{z}_k \parallel \begin{bmatrix} 0 \\ \vec{G}_k \end{bmatrix} \right) \oplus \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} H(A, 3k-3) \\ H(B, 3k-2) \\ H(A \oplus B, 3k-1) \end{bmatrix}$$

$R_{ij} := \mathsf{DecodeR}(\vec{r}, \mathsf{leak}, i, j)$

$E_k := X_{ij} \oplus R_{ij} \begin{bmatrix} A \\ B \end{bmatrix}$

return E

Figure 6: Our garbling scheme (continued from Figure 5). V^{-1} is a left inverse of V.

For each non-free gate, the garbler first samples a control matrix R and encodes its marginal views (*i.e.*, expresses each view in terms of the basis $\{S_j\}_j$). We have factored out this sampling procedure into a helper function SampleR, along with a corresponding decoding function DecodeR used by the evaluator to reconstruct its marginal view of the control matrix. One thing to note about SampleR is that in the case of a ODD gate, the control matrices include the term R_p, but R_p is not in the subspace spanned by the basis $\{S_j\}_j$. The compressed representation of each marginal view excludes the contribution of R_p, but in these cases it is publicly known that the evaluator should compensate by manually adding R_p.

For each gate k, we have a master evaluation equation in the style of Equation 9. This equation expresses constraints that must be true about that gate, but the garbler is interested in computing garbled gate ciphertexts $\vec{G}_k$, control bit ciphertexts $\vec{z}_k$, and output wire label that satisfy the constraints. As previously discussed, we can solve for these values by multipying both sides by V^{-1}, a left inverse of V. One possible choice of V^{-1} is given below:

$$V^{-1} = \left[\begin{array}{cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \tag{10}$$

The queries to hash function H include tweaks based on the gate ID, for domain separation. Finally, for each output wire, the garbler computes hashes of the wire labels, which will be used in Decode to authenticate labels and determine their logical value (true or false). These hashes need κ bits for authenticity, so they are computed using another hash function $H'(E,k)$ with output length κ instead $\frac{\kappa+d}{2}$. It is simplest to set $H'(E,k) = \mathrm{msb}_{\frac{\kappa}{2}}(H(E,3|f|+2k)) \,\|\, \mathrm{msb}_{\frac{\kappa}{2}}(H(E,3|f|+2k+1))$, which puts together κ bits from two evaluations of H, while avoiding any overlaps in tweaks.

The evaluator follows a similar process. Starting with the input wire labels E, it evaluates the garbled circuit one gate at a time. The invariant is that on wire k, the evaluator will hold the "active" wire label $E_k = W_k \oplus (x_k \oplus \pi_k)\Delta$, where x_k is the logical value on that wire, for the given circuit input. If A, B are the active wire labels on the input wires of this gate, then the evaluator computes terms of the form $H(A), H(B), H(A \oplus B)$ and evaluates the gate according to Equation 9. The evaluator only knows enough for two rows of Equation 9, depending on the color bits $i = \mathrm{lsb}(A)$, $j = \mathrm{lsb}(B)$, so we let V_{ij} be the corresponding pair of rows from V. It only evaluates the gate partially at first, in order to find the encoded control bits so that it can decode them with DecodeR and use them to finally compute the output wire label.

5.4 Security Proof

Theorem 3. *Let $\mathcal{H}$ be a family of hash functions, with output length $(\kappa+d)/2$ bits, that is RTCCR for $\mathcal{L} = \{L_{ab}(\Delta_L \| \Delta_R) = 0^{d/2} \| a\Delta_L \oplus b\Delta_R \mid a, b, \in \{0,1\}\}$. Then our construction (Figures 5 and 6) is a secure garbling scheme.*

Proof. We need to prove four properties of the construction.

Correctness: We need to prove an invariant: $E_k = W_k \oplus (x_k \oplus \pi_k)\Delta$ for all k, if x_k is the plaintext value on that wire. Encode chooses the inputs in this way, so at least it's true for $k \leq$ inputs, and it is trivially maintained for free-XOR gates. For any $v \in \mathrm{colspace}(V) = \mathcal{G}$, we have $VV^{-1}v = v$, as there exists some u such that $v = Vu$ and $VV^{-1}Vu = Vu = v$ because V^{-1} is a left inverse of V. In Section 5.1 we showed that $\mathrm{colspace}(M) = \mathcal{G}$, $\mathrm{colspace}(R \oplus \left[0 \cdots 0 \middle| t\right]) \subseteq \mathcal{G}$, and $\vec{r} \in \mathcal{G}$,

so after multiplying both sides of garbler's equation by V on the left, the VV^{-1}s will cancel, and taking a two-row piece of this equation gives the evaluator's equation. In this equation, X_{ij} is the two rows of

$$\vec{X} = C \oplus \left(R \oplus \left[0 \cdots 0 \,\middle|\, t\right]\right) \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix}, \tag{11}$$

corresponding to the evaluation case i, j. The structure of R (see Equation 6) implies that the evaluator's row pair of $R[A_0^\top \; B_0^\top \; \Delta^\top]^\top$ will be $R_{ij}[A^\top \; B^\top]^\top$. Therefore

$$E_k = X_{ij} \oplus R \begin{bmatrix} A \\ B \end{bmatrix} = C \oplus t_{ij}\Delta = W_k \oplus (\mathsf{eval}(k)(\pi_A \oplus i, \pi_B \oplus j) \oplus \pi_k)\Delta,$$

which maintains this invariant because

$$i = \mathrm{lsb}(E_{\mathsf{in}_1(k)}) = \mathrm{lsb}\big(W_{\mathsf{in}_1(k)} \oplus (x_{\mathsf{in}_1(k)} \oplus \pi_{\mathsf{in}_1(k)})\Delta\big) = x_{\mathsf{in}_1(k)} \oplus \pi_{\mathsf{in}_1(k)},$$

and similarly for j. Finally, Decode will correctly find that $D_k^{x_k} = H'\big(W_k \oplus (x_k \oplus \pi_k)\Delta, k\big) = H'(E_k, k)$, assuming that $D_k^{x_k} \neq D_k^{1-x_k}$, which has only negligible probability of failing. Therefore it gives the correct result.

Privacy: We need to prove that generating $(\Phi, \vec{G}, \vec{z}), E, (\Phi, D)$ with Garble and Encode is indistinguishable from the output of $\mathcal{S}_{\mathrm{priv}}$. We give a sequence of intermediate hybrids, going from the real garbler to the simulator.

Hybrid 1: This hybrid switches from the garbler's perspective to the evaluator's perspective when garbling the circuit. Instead of keeping track of the "zero" wire label W_k for every gate, we keep track of the "active" wire label E_k, and rewrite the garbling procedure in terms of the "active" labels. This basically involves a change of variable names throughout the garbling algorithm. The changes are extensive, and given in detail in Figure 7:

- Replace point-and-permute bits π_k with the equivalent expression $x_k \oplus \mathrm{lsb}(E_k)$.
- Write the control matrix part of the garbling equation in terms of active wire labels $A = E_{\mathsf{in}_1(k)}$ and $B = E_{\mathsf{in}_2(k)}$ instead of A_0 and B_0.

  $$\text{replace } R \times \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix} \text{ with equivalent } R' \times \begin{bmatrix} A \\ B \\ \Delta \end{bmatrix}.$$

  where a change of basis has been applied to R, that expresses A_0 as the appropriate linear combination of A and Δ, and expresses B_0 in terms of B and Δ.
- Partition $\vec{H}$ into two pieces:

  $$\vec{H}_0 = [H(A) \; H(B) \; H(A \oplus B)]^\top$$
  $$\vec{H}_\Delta = [H(A \oplus \Delta) \; H(B \oplus \Delta) \; H(A \oplus B \oplus \Delta)]^\top$$

  where again A and B are the active wire labels. Similarly partition the matrix M into M_0 and M_Δ, and replace $M \times \vec{H}$ with $(M_0\vec{H}_0 \oplus M_\Delta\vec{H}_\Delta)$.

$\underline{\mathcal{S}_{\text{priv}}(1^\kappa, \Phi, x)}$:
  $(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}) := \Phi$
  $(F, E) \leftarrow \mathcal{S}_{\text{obliv}}(1^\kappa, \Phi)$
  $E' := \mathsf{Eval}(F, E)$
  for $k \in \mathsf{outputs}$:
    $D_k^{x_k} := H'(E'_k, k)$
    $D_k^{1-x_k} \leftarrow \mathrm{GF}(2^\kappa)$
  return $F, E, (\Phi, D)$

$\underline{\mathcal{S}_{\text{obliv}}(1^\kappa, \Phi)}$:
  $(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}) := \Phi$
  $H \leftarrow \mathcal{H}$
  for $k = 1$ to inputs:
    $E_k \leftarrow \mathrm{GF}(2^{\kappa/2})^2$
  for $k = \mathsf{inputs} + 1$ to $|\Phi|$:
    if $\mathsf{leak}(k) = \mathsf{XOR}$: continue
    $\vec{G}_k \leftarrow \mathrm{GF}(2^{\kappa/2})^3$
    $\vec{z}_k \leftarrow \mathrm{GF}(2^{d(\mathsf{leak}(k))/2})^5$
  return $(\Phi, H, \vec{G}, \vec{z}), E$

$\underline{\mathsf{Hybrid1}(1^\kappa, f, x)}$:
  $(\mathsf{inputs}, \mathsf{outputs}, \mathsf{in}, \mathsf{leak}, \mathsf{eval}) := f$
  $H \leftarrow \mathcal{H}$
  $\Delta \leftarrow \begin{bmatrix} 1 \,\|\, \mathrm{GF}(2^{\kappa/2-1}) \\ \mathrm{GF}(2^{\kappa/2}) \end{bmatrix}$
  for $k = 1$ to inputs:
    $E_k \leftarrow \mathrm{GF}(2^{\kappa/2})^2$
  for $k = \mathsf{inputs} + 1$ to $|f|$:
    $A, B := E_{\mathsf{in}_1(k)}, E_{\mathsf{in}_2(k)}$
    $i, j := \mathrm{lsb}(A), \mathrm{lsb}(B)$
    $x_A, x_B := x_{\mathsf{in}_1(k)}, x_{\mathsf{in}_2(k)}$
    if $\mathsf{leak}(k) = \mathsf{XOR}$:
      $E_k := A \oplus B$
      $x_k := x_A \oplus x_B$
      continue
    $g := \mathsf{eval}(k)$
    $x_k := g(x_A, x_B)$
    $t := \begin{bmatrix} g(x_A \oplus i, x_B \oplus j) \\ g(x_A \oplus i, x_B \oplus j \oplus 1) \\ g(x_A \oplus i \oplus 1, x_B \oplus j) \\ g(x_A \oplus i \oplus 1, x_B \oplus j \oplus 1) \end{bmatrix}$
    $R, \vec{r} := \mathsf{SampleR}(t, \mathsf{leak}(k))$
    $R' := R \begin{bmatrix} 1 & 0 & 0 & 0 & i & 0 \\ 0 & 1 & 0 & 0 & 0 & i \\ 0 & 0 & 1 & 0 & j & 0 \\ 0 & 0 & 0 & 1 & 0 & j \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
    $\vec{H}_0 := \begin{bmatrix} H(A, 3k-3) \\ H(B, 3k-2) \\ H(A \oplus B, 3k-1) \end{bmatrix}$
    $\vec{H}_\Delta := \begin{bmatrix} H(A \oplus \Delta, 3k-3) \\ H(B \oplus \Delta, 3k-2) \\ H(A \oplus B \oplus \Delta, 3k-1) \end{bmatrix}$
    $(\vec{z}_k)_{\mathsf{bot}} \,\|\, \vec{G}_k := V_{\mathsf{gate}}^{-1} \left(\vec{r} \,\|\, \left(R' \oplus \left[0 \cdots 0 \,\middle|\, t\right]\right) \begin{bmatrix} A \\ B \\ \Delta \end{bmatrix} \right)$
      $\oplus\, V_{\mathsf{gate}}^{-1}(M_0 \vec{H}_0 \oplus M_\Delta \vec{H}_\Delta)$
    $E_k := V_{ij} \begin{bmatrix} 0 \\ \vec{G}_k \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \mathrm{msb}_{\frac{\kappa}{2}}(\vec{H}_0) \oplus R_{ij} \begin{bmatrix} A \\ B \end{bmatrix}$
    $(\vec{z}_k)_{\mathsf{top}} := V_{ij} \begin{bmatrix} 0 \\ (\vec{z}_k)_{\mathsf{bot}} \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \mathrm{lsb}_{\frac{d}{2}}(\vec{H}_0) \oplus \vec{r}_{ij}$
  for $k \in \mathsf{outputs}$, $j \in \{0, 1\}$:
    $D_k^j := H'(E_k \oplus (j \oplus x_k)\Delta, k)$
  return $(\Phi(f), \vec{G}, \vec{z}), E, (\Phi(f), D)$

Figure 7: Left: simulators for privacy and obliviousness. Right: a hybrid for privacy.

- Note that the matrix V^{-1} has 5 rows, where the first 2 correspond to slices of the output label and the last 3 correspond to the gate ciphertexts. Denote this division of V^{-1} by V^{-1}_{label} and V^{-1}_{gate}. Instead of multiplying on the left by V^{-1} to solve for the output label and gate ciphertexts, we now multiply on the left by V^{-1}_{gate} to solve for only the gate ciphertexts. We then evaluate those gate ciphertexts with A and B to learn the (active) output label E_k. This different approach has the same result by the correctness of the scheme.

  We can similarly partition the control bit ciphertexts $\vec{z}_k$ into $\vec{z}_k = [(\vec{z}_k)_{\mathsf{top}}\ (\vec{z}_k)_{\mathsf{bot}}]$, use V^{-1}_{gate} to compute $(\vec{z}_k)_{\mathsf{bot}}$, and then use the evaluator's computation to solve for $(\vec{z}_k)_{\mathsf{top}}$. Solving for $(\vec{z}_k)_{\mathsf{top}}$ is simplified by the first two columns of V_{ij} being the identity matrix. In this case, we solve for the missing positions using knowledge of the compressed control bits $\overline{r}_{ij}$.

All of the changes are simple variable substitutions or basis changes in the linear algebra, so this hybrid is distributed identically to the real garbling.

Hybrid 2: In this hybrid, we apply the RTCCR property of H to all oracle queries of the form $H(\cdot \oplus \Delta)$. We must show that Δ is used in a way that can be achieved by calling the oracle from the RTCCR security game.

We focus on the term

$$V^{-1}_{\mathsf{gate}} M \vec{H} = V^{-1}_{\mathsf{gate}}(M_0 \vec{H}_0 \oplus M_\Delta \vec{H}_\Delta)$$

First, consider the expression $V^{-1} \times M$, and recall that M is written in terms of the zero-labels A_0, B_0. Using the V^{-1} given in Equation 10, we can compute:

$$V^{-1}M = \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \tag{12}$$

Thus $V^{-1}_{\mathsf{gate}} \times M$ will consist of the bottom three rows of Equation 12.

Recall that the columns of M correspond to oracle queries $H(A_0), H(A_0 \oplus \Delta), H(B_0), H(B_0 \oplus \Delta), H(A \oplus B), H(A \oplus B \oplus \Delta)$, in that order. In the current hybrid M is partitioned into M_0 (corresponding to H-queries on active labels) and M_Δ (corresponding to the other queries). In other words, M_Δ will consist of exactly one of rows $\{1, 2\}$, exactly one of rows $\{3, 4\}$, and exactly one of rows $\{5, 6\}$ from M. In all cases, the result of $V^{-1}_{\mathsf{gate}} M_\Delta$ (*i.e.*, the bottom 3 rows of $V^{-1} M_\Delta$) is the 3×3 identity matrix!

This means we can rewrite the hybrid in the following way:

$$\begin{aligned}(\vec{z}_k)_{\mathsf{bot}} \left\| \vec{G}_k \right. &:= V^{-1}_{\mathsf{gate}} \left(\vec{r} \left\| \left(R' \oplus \left[0 \cdots 0 \,\middle|\, t\right]\right) \begin{bmatrix} A \\ B \\ \Delta \end{bmatrix} \right.\right) \oplus V^{-1}_{\mathsf{gate}}(M_0 \vec{H}_0 \oplus M_\Delta \vec{H}_\Delta) \\ &= \vec{H}_\Delta \oplus [\text{linear combinations of } \Delta] \oplus \cdots\end{aligned}$$

Since all the H-queries in $\vec{H}_\Delta$ include a Δ term, we can compute this expression with 3 suitable calls to the RTCCR oracle.[8] Finally, $D_k^{1-x_k} = H'(E_k \oplus \Delta, k)$ also uses Δ, and will become two calls to the RTCCR oracle. These transformations successfully moves all references to Δ into the RTCCR oracle.

[8] Note also that the calls to H have globally distinct tweaks.

Applying RTCCR security, it has negligible effect to replace the results of these H-queries with uniformly random values. This has the effect of making the entire expression uniform, *i.e.*:

$$(\vec{z}_k)_{\mathsf{bot}} \;\Big\|\; \vec{G}_k \leftarrow \mathrm{GF}(2^{(\kappa+d)/2})^3$$

Also, $D_k^{1-x_k}$ is now sampled uniformly at random in $\mathrm{GF}(2^\kappa)$.

Hybrid 3: After making the previous change, the only place that R is used is when we use the marginal views R_{ij} and $\vec{r}_{ij}$ to solve for the output label and for the missing pieces of the control bit ciphertexts. In Section 5.1 we specifically chose $\mathcal{R}$ so that this marginal views is uniform for all t and all i, j. Therefore instead of doing $R, \vec{r} \leftarrow \mathsf{SampleR}(t, \mathsf{leak}(k))$, we can simply choose uniform $\vec{r}_{ij}$ and use $\mathsf{DecodeR}$ to reconstruct R_{ij}. The change has no effect on the overall view of the adversary.

Note that after making this change, the control-bit ciphertexts $(\vec{z}_k)_{\mathsf{top}}$ become uniform since $\vec{r}_{ij}$ acts as a one-time pad.

Hybrid 4: As a result of the previous change, the hybrid no longer uses t. Additionally, t was the only place where the plaintext values x_k were used, other than in the computation of D. But D only uses plaintext values for the circuit's output wires. In other words, the entire hybrid can be computed knowing only the circuit output $f(x)$. Additionally, all garbled gate ciphertexts and control bit ciphertexts are chosen uniformly, and the active wire labels on output wires are determined by the scheme's evaluation procedure. Hence, the hybrid exactly matches what happens in $\mathcal{S}_{\text{priv}}$.

Obliviousness: Notice that $\mathcal{S}_{\text{priv}}$ calls $\mathcal{S}_{\text{obliv}}$ to generate (F, E), then samples some more random bits for decoding and returns it all. Therefore, any adversary for obliviousness could be turned into one for privacy by only looking at (F, E) and ignoring the rest.

Authenticity: The first two steps of the authenticity distribution are exactly the same as the real privacy distribution, so we can swap them for the simulated distribution $\mathcal{S}_{\text{priv}}$ in a hybrid. Then to break authenticity the adversary must cause Decode to choose $j = 1 - x_k$ for at least one output k, as otherwise it will either produce the correct answer or abort. But $D_k^{1-x_k}$ is fresh uniform randomness, so the probability that $D_k^{1-x_k} = H'(E_k, k)$ is $2^{-\kappa}$. □

5.5 Discussion

Concrete costs. The garbler makes 6 calls to H per non-free gate, while the evaluator makes 3 calls to H per non-free gate.

Each non-free garbled gate consists of gate ciphertexts $\vec{G}$ and encrypted control bits $\vec{z}$. There are 3 gate ciphertexts, each being $\kappa/2$ bits long. The encrypted control bits are a vector of length 5, where each component of the vector has length $d/2$ (where d is the dimension of the control matrix subspace). For the standard (parity-leaking) instantiation of our scheme, $d = 2$ and we get that the total size of a garbled gate is $1.5\kappa + 5$ bits. For the gate-hiding instantiation, $d = 4$ and we get a size of $1.5\kappa + 10$ bits.

Comparison to half-gates. We assume that calls to H are the computational bottleneck, in any implementation of both our scheme and in half-gates [ZRE15]. The following analysis therefore ignores the cost of xor'ing wire labels and bit-fiddling related to color bits and control bits.

In the time it takes to call H 12 times, half-gates generates 3 gates and sends 6κ bits (4 calls to H and 2κ bits per gate), while our scheme generates 2 gates and sends 3κ bits (6 calls to H and 1.5κ bits per gate). Thus, a CPU-bound implementation of our scheme will produce garbled output at half the rate of half-gates. We evaluated the optimized half-gates garbling algorithm from the ABY3 library [MR18], and found it capable of generating garbled output at a rate of ∼850 Mbyte/s on single core of a i7-7500U laptop processor running at 3.5GHz. Thus, we conservatively estimate that a comparable implementation of our scheme could generate garbled output at ∼400 Mbyte/s = 3.2 Gbit/s. This rate would still leave our scheme network-bound in most situations and applications of garbled circuits. When both half-gates and our scheme are network bound, our scheme is expected to be ∼25% faster by virtue of reducing communication by 25%.

6 Optimizations

6.1 Optimizing Control Bit Encryptions

In our scheme the control bit encryptions $\vec{z}$ is a vector of length 5, where the components in that vector are each a single bit (in the case of parity-leaking gates) or 2 bits (in the case of parity-hiding gates). These ciphertexts therefore contribute 5 or 10 bits to the size of each garbled gate.

We remark that it is possible to use ideas of garbled row reduction [NPS99, PSSW09] to reduce $\vec{z}$ to a length-3 vector. This will result in these ciphertexts contributing 3 or 6 bits to the garbled gate. Such an optimization may be convenient in parity-hiding case, where the change from 10 to 6 bits allows these control bit ciphertexts to fit in a single byte.

Recall that in the security proof, we partition the control bit ciphertexts $\vec{z}$ into $(\vec{z})_{\mathsf{top}}$ (2 components) and $(\vec{z})_{\mathsf{bot}}$ (3 components). Our idea to reduce their size is to simply fix $(\vec{z})_{\mathsf{top}}$ to zeroes, so that these components do not need to be explicitly included in the garbled gate. The evaluator can act exactly as before, taking the missing values from $\vec{z}$ to be zeroes. The garbler must sample the control matrix subject to it causing $(\vec{z})_{\mathsf{top}} = 0$.

A drawback to this optimization is that it significantly complicates the security proof (and hence why we only sketch it here). When we apply the security of RTCCR in the security proof, the hybrid acts as follows:

1. It uses the $d/2$ least significant bits of the H-outputs to determine how the control bits are going to be "masked".

2. Based on these masks, it chooses a consistent control matrix R that causes the first two components of $\vec{z}$ to be 0.

3. The choice of R determines which linear combinations of wire label slices (including slices of Δ) are applied.

So the reduction to RTCCR security must first read the low bits of several $H(\cdot \oplus \Delta)$ queries before it decides which linear combination of Δ should be XOR'ed with the remaining output of H. Of course the RTCCR oracle requires the choice of linear combination to be provided when H is called. It is indeed possible to formally account for this, but only by modeling the two parts of H's output (for masking wire label slices and for masking control bits) as separate hash functions for the purposes of the security proof.

circuit	baseline	optimized	improvement	half-gates [ZRE15]
64-bit adder	6.00	6.00	0%	4.00
64-bit division	6.00	5.75	4.1%	4.00
64-bit multiplication	6.00	4.99	16.8%	4.00
AES-128	6.00	4.31	28.2%	4.00
SHA-256	6.00	5.77	3.8%	4.00
Keccak f	6.00	4.00	33.3%	4.00

Figure 8: Number of calls to κ-bit H' RTCCR function (per AND gate) to garble each circuit, with and without the optimization of Section 6.2. Evaluating the garbled circuit costs exactly half this number of calls to H'.

6.2 Optimizing Computation

Our construction requires a RTCCR function H with output length $(\kappa + d)/2$. We propose an efficient instantiation of H which naturally results in κ-bit output, which is then truncated to $(\kappa + d)/2$. The hash produces nearly twice as many bits as needed, raising the question of whether we are "wasting" these extra bits. In fact, if we reduce the security parameter slightly so that H is derived from a $(\kappa + d)$-bit primitive, we can use these extra bits to reduce the computation cost.

Suppose H' is a [RT]CCR with $(\kappa + d)$ bits of output. Then define

$$H(X, \tau) = \begin{cases} \text{first half of } H'(X, \frac{\tau}{2}) & \tau \text{ even} \\ \text{second half of } H'(X, \frac{\tau-1}{2}) & \tau \text{ odd} \end{cases}$$

Clearly H is also a [RT]CCR with $(\kappa + d)/2$ bits of output. How can we use this H to reduce the total number of calls to the underlying H'?

When a wire with labels $(A, A \oplus \Delta)$ is used as input to an AND gate, our scheme makes calls of the form $H(A, j), H(A \oplus \Delta, j)$ where j is the ID of that AND gate. Let us slightly change how the tweaks are used. Suppose this wire with label $(A, A \oplus \Delta)$ is used as input in n different AND gates. Then those gates should make calls of the form $H(A, 0 \parallel i), H(A, 1 \parallel i), \ldots, H(A, n-1 \parallel i)$, where i is now the index of *the wire whose labels are* $(A, A \oplus \Delta)$. When H is defined as above, these queries can be computed with only $\lceil n/2 \rceil$ queries to H'.

Note that both the garbler and evaluator can take advantage of this optimization, with the garbler always requiring exactly twice as many calls to H' (if in some scenario the evaluator needs $H'(X)$ then the garbler will need $H'(X)$ and $H'(X \oplus \Delta)$). Our AND gates require calls to H of the form $H(A), H(B), H(A \oplus B)$, and so far we have discussed optimizing only the $H(A)$ and $H(B)$ queries. Similar logic can be applied to the queries of the form $H(A \oplus B)$; for example, if a circuit contains gates $a \wedge b$ and $(a \oplus b) \wedge c$, then both of those AND gates will require $H(A \oplus B)$ terms that can be optimized in this way.

We explored the effect of this optimization for a selection of circuits.[9] The results are shown in Figure 8. The improvement ranges from 0% to 33.3%. As a reference, our baseline construction requires 6 calls to ($(\kappa + d)/2$-bit output) H to garble an AND gate, while half-gates requires 4 calls (to a κ-bit function). Interestingly, in the Keccak f-function every wire used as input to an *even number* of AND gates, so that our optimized scheme has the same computation cost as half-gates (4 calls to H' per AND gate). In principle, this optimization can result in as few as 3 calls to H' per AND gate,[10] but typical circuits do not appear to be nearly so favorable.

[9] Circuits were obtained from https://homes.esat.kuleuven.be/~nsmart/MPC/

[10] This can happen, *e.g.*, when for every $a \wedge b$ gate there is a corresponding $a \vee b = \overline{\overline{a} \wedge \overline{b}}$ gate.

7 The Linear Garbling Lower Bound

In [ZRE15], the authors present a lower bound for garbled AND gates in a model that they call **linear garbling.** The linear garbling model considers schemes with the following properties:

- Wire labels have an associated *color bit* which must be $\{0, 1\}$.
- To evaluate the garbled gate, the evaluator makes a sequence of calls to a random oracle (that depend only on the input wire labels), and then outputs some linear combination of input labels, gate ciphertexts, and random oracle outputs. The linear combination must depend only on the color bits of the input labels.

The bound of [ZRE15] considers only linear combinations over the field $\mathrm{GF}(2^\kappa)$, and it is unclear to what extent the results generalize to other fields.

Several works have bypassed this lower bound, and we summarize them below. All of these works show how to garble an AND gate for $\kappa + O(1)$ bits, but only a single AND gate in isolation. These constructions all require the input wire labels to satisfy a certain structure, but do not guarantee that the output labels also satisfy that structure.

- Kempka, Kikuchi, and Suzuki [KKS16] and Wang & Malluhi [WmM17] both use a technique of randomizing the control bits. The evaluator decrypts a constant-size ciphertext to determine which linear combination to apply. This approach is outside of the linear garbling model, which requires that the linear combination depend only on the color bits. These works also add wire labels in $\mathbb{Z}_{2^\kappa}$ rather than XOR them (as in $\mathrm{GF}(2^\kappa)$). Apart from these similiarities, the two approaches are quite different.
- Ball, Malkin, and Rosulek [BMR16] deviate from the linear garbling model by letting each wire label have a color "trit" from $\mathbb{Z}_3$ instead of a color bit from $\mathbb{Z}_2$. There is no further "indirection" of the evaluator's linear combination — it depends only on the colors of the input labels. They also perform some linear combinations on wire labels over a field of characteristic 3.

As described earlier, we bypass the lower bound by adopting the control-bit randomization technique of [KKS16] but also introducing the wire-label-slicing technique.

8 Open Problems

We conclude by listing several open problems suggested by our work.

Optimality. Is 1.5κ bits optimal for garbled AND gates in a more inclusive model than the one in [ZRE15]? A natural model that excludes "heavy machinery" like fully homomorphic encryption is Minicrypt, in which all parties are computationally unbounded but have bounded access to a random oracle. Conversely, can one do better — say, $4\kappa/3$ bits per AND gate? Does it help to sacrifice compatibility with free-XOR? In our construction, free-XOR seems crucial.

Computation Cost. In Section 6.2 we described how to reduce the number of queries to an underlying κ-bit primitive, with an optimization that depends on topology of the circuit. Is there a way to reduce the computation cost of our scheme (measured in number of calls to, say, a κ-bit ideal permutation), for *all* circuits?

In the best case, we can garble a circuit for only 3 (amortized) calls per AND gate, whereas all prior schemes require 4. Setting aside garbled circuit size and free-XOR compatibility, is there any scheme that can garble arbitrary circuits for less than 4 (amortized) calls to a κ-bit primitive per AND gate?

Hardness Assumption. Free-XOR garbling requires some kind of circular correlation-robust assumption (see [CKKZ12] for a formal statement). The state-of-the-art garbling scheme based on the minimal assumption of PRF is due to Gueron *et al.* [GLNP15], where AND gates cost 2κ and XOR gates cost κ bits. Can our new techniques be used to improve on garbling from the PRF assumption, or alternatively can the optimality of [GLNP15] be proven? Again, our construction seems to rely heavily on the free-XOR structure of wire labels, which (apparently) makes circular correlation robustness necessary.

Privacy-Free Garbling. Frederiksen *et al.* [FNO15] introduced *privacy-free* garbled circuits, in which only the authenticity property is required of the garbling scheme. The state-of-the-art privacy-free scheme is due to [ZRE15], where XOR gates are free and AND gates cost κ bits. Can our new techniques lead to a privacy-free garbling scheme with less than κ bits per AND gate (with or without free-XOR)?

Simpler Description. Is there a way to describe our construction as the clean composition of simpler components, similar to how the half-gates construction is described in terms of simpler "half gate" objects? The challenge in our scheme is the way in which left-slices and right-slices of the wire labels are used together.

References

[BHK+99] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 216–233. Springer, Heidelberg, August 1999.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

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

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

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

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

[CR16] Brent Carmer and Mike Rosulek. Linicrypt: A model for practical cryptography. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 416–445. Springer, Heidelberg, August 2016.

[FNO15] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 191–219. Springer, Heidelberg, April 2015.

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

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

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

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

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

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

[MR18] Payman Mohassel and Peter Rindal. ABY^3: A mixed protocol framework for machine learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 35–52. ACM Press, October 2018.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, New York, NY, USA, 1999. ACM.

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

[Ros17] Mike Rosulek. Improvements for gate-hiding garbled circuits. In Arpita Patra and Nigel P. Smart, editors, *INDOCRYPT 2017*, volume 10698 of *LNCS*, pages 325–345. Springer, Heidelberg, December 2017.

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

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

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

Δ	free-XOR wire label offset (sometimes a vector of slices $[\Delta_L, \Delta_R]^\top$)
κ	security parameter (*e.g.*, 128)
π_k	point-permute bit of wire k: wire label W_k represents plaintext value π_k
Φ	circuit leakage function
A_0, B_0	input wire labels with color bit 0
A, B	"active" wire labels (sometimes a vector of slices $[A_L, A_R]^\top$)
C	an output wire label (sometimes a vector of slices $[C_L, C_R]^\top$)
d	dimension of basis S for control bit marginal views R_{ij}
E_k	active wire label on wire k
$\mathcal{G}$	"gate space" = column space of all matrices in (Equation 4)
$\vec{G}$	vector of gate ciphertexts ($\vec{G}_k$ for the kth gate)
$H, \mathcal{H}$	RTCCR hash function, family
$\vec{H}$	vector of responses from H: $[H(A), H(A \oplus \Delta), \ldots]$ (Equation 4)
$\vec{H}_0$, $\vec{H}_\Delta$	partition of $\vec{H}$ corresponding to active/inactive queries to H, respectively (in the security proof)
K	basis matrix for cokernel of gate space $\mathcal{G}$; *i.e.*, every $v \in \mathcal{G}$ satisfies $Kv = 0$
$\mathcal{L}$	set of linear functions for an RTCCR hash
M	matrix corresponding to linear combinations of H-responses (Equation 4)
M_0, M_Δ	partition of M corresponding to active/inactive queries to H, respectively (in the security proof)
R	control bit matrix (Equation 4), specifying how input label slices are used in linear combinations
$\mathcal{R}(t)$	distribution over control matrices R corresponding to gate truth table t; sometimes takes gate leakage as additional argument
R'	control bit matrix R after applying a basis change in the security proof
R_p	control matrix that is always included for odd-parity gates
R_{ij}	marginal view of control matrix R, for one gate-input combination
$\bar{R}$, $\bar{R}_{ij}$	compressed representation of control matrix R, or of marginal view R_{ij}, expressed in basis S_k
$\vec{r}$	vector encoding of control matrix R, for garbling the control bits
S	basis for control bit marginal views R_{ij} (basis elements S_j); sometimes has a leakage argument
t	truth table of the gate: an 8×2 matrix composed of 2×2 identity blocks and 2×2 zero blocks
t_{ij}	the 2×2 block of t corresponding to a single gate input combination (in correctness proof)
V	matrix on LHS of the main garbling equation (Equation 4), corresponding to output label and gate ciphertexts
V_{ij}	pair of rows from V used by the evaluator when $\mathrm{lsb}(A) = i$ and $\mathrm{lsb}(B) = j$
V^{-1}	a left-inverse of V
V^{-1}_{gate}, V^{-1}_{label}	partition of V^{-1} corresponding to gate ciphertexts and output label slices, respectively (in the security proof)
W_k	wire label on wire k with color bit 0
$\vec{X}$, X_{ij}	gate output wire labels (resp. wire label) before applying control matrix (Equation 11, Figure 6)
x_k	plaintext value on wire k (in security proof)
$\vec{z}$	garbling/encryption of control bit matrix R / its encoding $\vec{r}$ ($\vec{z}_k$ for the kth gate)

Figure 9: Guide to notation.

A Randomized Tweakable Circular Correlation Robust Functions

A.1 Circular Correlation Robustness

We build to our final construction in two steps, the first of which is to construct a simpler circular correlation robust hash in the ideal permutation model.

Below we give the definition of circular correlation robustness from [GKWY20], but generalized to support non-matching input/output lengths and a family of linear functions $\mathcal{L}$. The original definition corresponds to the case $n = m$ and $\mathcal{L}$ containing the identity function and the all-zeroes function.

Definition 4. *Let $H : \{0,1\}^n \to \{0,1\}^m$, and let $\mathcal{L}$ be a set of linear functions from $\{0,1\}^n \to \{0,1\}^m$. Then H is **circular correlation robust for** $\mathcal{L}$ if for all PPT $\mathcal{A}$,*

$$\left| \Pr_{\Delta}\left[\mathcal{A}^{\mathcal{O}_{H,\Delta}}() = 1\right] - \Pr_{R}\left[\mathcal{A}^{R}() = 1\right] \right|$$

is negligible, where $\mathcal{O}^{\mathsf{ccr}}_{H,\Delta}$ is defined as:

$$\boxed{\begin{array}{l}\mathcal{O}_{H,\Delta}(X \in \{0,1\}^n, L \in \mathcal{L}): \\ \quad \text{return } H(X \oplus \Delta) \oplus L(\Delta)\end{array}}$$

We show how to construct such a function for $n = m$ in the ideal permutation model. Our constructions require $m \leq n$, and such a function can be obtained by simply truncating one with $n = m$.

Lemma 5. *Fix a set of linear transformations $\mathcal{L}$, and let σ be any function such that:*

- *sigma is linear, so that $\sigma(X \oplus Y) = \sigma(X) \oplus \sigma(Y)$*
- *$X \mapsto L(X) \oplus \sigma(X)$ is invertible for all $L \in \mathcal{L}$.*

If π is an ideal permutation (all parties have oracle access to a random permutation π and its inverse π^{-1}), then

$$H(X) = \pi(X) \oplus \sigma(X)$$

is circular correlation robust for $\mathcal{L}$.

This construction is the natural generalization of the one from [GKWY20], who consider $\mathcal{L}$ to contain only the zero-function and the identity function. In that case, our restrictions on σ amount to requiring that σ is an *orthomorphism.*

Proof. Consider the following game, in the style of Bellare & Rogaway [BR06]:

$\Delta \leftarrow \{0,1\}^n$

$\underline{\pi(A)}$:
- if $\exists B : (A,B) \in \Pi^*$:
  - $bad_1 = 1$
  - return B
- $B \leftarrow \{0,1\}^n \setminus \mathsf{right}(\Pi)$
- while $\exists A' : (A',B) \in \Pi^*$:
  - $bad_3 = 1$
  - $B \leftarrow \{0,1\}^n \setminus \mathsf{right}(\Pi)$

  } $B \leftarrow \{0,1\}^n \setminus \mathsf{right}(\Pi \cup \Pi^*)$
- add (A,B) to Π
- return B

$\underline{\pi^{-1}(B)}$:
- if $\exists A : (A,B) \in \Pi^*$:
  - $bad_2 = 1$
  - return A
- $A \leftarrow \{0,1\}^n \setminus \mathsf{left}(\Pi)$
- while $\exists B' : (A,B') \in \Pi^*$:
  - $bad_4 = 1$
  - $A \leftarrow \{0,1\}^n \setminus \mathsf{left}(\Pi)$

  } $A \leftarrow \{0,1\}^n \setminus \mathsf{left}(\Pi \cup \Pi^*)$
- add (A,B) to Π
- return A

$\underline{\mathcal{O}(X,L)}$:
- if $\exists B : (X \oplus \Delta, B) \in \Pi$:
  - $bad_1 = 1$
  - return $B \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$
- $Z \leftarrow \{0,1\}^n$
- $B = Z \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$
- while $\exists A : (A,B) \in \Pi \cup \Pi^*$:
  - $bad_5 = 1$
  - $Z \leftarrow \{0,1\}^n$
  - $B = Z \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$

  } $B \leftarrow \{0,1\}^n \setminus \mathsf{right}(\Pi \cup \Pi^*)$, $Z = B \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$
- add $(X \oplus \Delta, B)$ to Π^*
- return Z

The adversary gets oracle access to these 3 oracles and finally outputs a bit. Without loss of generality, assume that the adversary does not repeat identical queries, does not query π on a previous output of π^{-1}, and does not query π^{-1} on a previous output of π — in all of these cases, the answer to the query is already known. We make the following observations:

- **Without the highlighted lines, the game matches the "ideal" CCR experiment.** The $\pi^{\pm}$ oracles instantiate an ideal permutation on the fly, in the usual way, keeping track of

the input/output pairs in the set Π. $\mathsf{left}(\Pi)$ and $\mathsf{right}(\Pi)$ denote the left/right element of all tuples in Π, respectively. $\mathcal{O}$ is instantiated as an independent random function.

- **With the highlighted lines, the game matches the "real" CCR experiment.** A query of the form $\mathcal{O}(X, L) = Z$ corresponds precisely to an internal query of the form $\pi(X \oplus \Delta) = Z \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$. This game maintains a set Π^* of input/output pairs for $\pi^{\pm}$ that are defined as a result of a query to $\mathcal{O}$. The added if-statements ensure consistency when the same π input/output pair is used in both a query to $\pi^{\pm}$ and to $\mathcal{O}$. The added while-statements ensure that π always remains a permutation. Note that the added while-statements, along with the preceding lines, can be simplified as we show to the right.
- **The two games are identical-until-bad.** With or without the highlighted lines, the games execute identical statements until one of the bad_i flags is set to 1.

From [BR06], we have that the distinguishing advantage of the adversary is bounded by Pr[any bad_i is set to 1]. This probability can be calculated with respect to the ideal game (highlighted code is not executed, except to set the bad_i flags). Since the bad_i flags do not affect the execution in the ideal game, it is most convenient to consider that the adversary makes *all* queries, and only when the game is over do we inspect the execution and determine whether the bad_i flags are set. It suffices to show that each bad_i flag is set to 1 with only negligible probability.

- bad_1 is set to 1 (in either of two places) only if the adversary manages to directly query $\pi(A)$ and also $\mathcal{O}(X, L)$ where $X = A \oplus \Delta$ — the two queries can happen in either order. Note that in the ideal game variant, Δ is independent of the adversary's view. It is equivalent to choose Δ at the end of the execution, after all queries have been made, and when we are checking whether any bad_i flag was set. For a two specific queries (one to π and one to $\mathcal{O}$), the probability that they satisfy $X = A \oplus \Delta$ is $1/2^n$. Hence if the adversary makes a total of q oracle queries, bad_1 is set with probability at most $q^2/2^n$ by a simple union bound.
- bad_2 is set to 1 only if the adversary manages to query $\mathcal{O}(X, L) = Z$ and later query $\pi^{-1}(B)$ where $Z = B \oplus \sigma(X \oplus \Delta) \oplus L(\Delta)$. As above, imagine Δ being chosen after all queries have been made. For a specific pair of queries (one to π^{-1} and one to $\mathcal{O}$), the probability that

$$\begin{aligned} Z &= B \oplus \sigma(X \oplus \Delta) \oplus L(\Delta) \\ &= B \oplus \sigma(X) \oplus \Big(\sigma(\Delta) \oplus L(\Delta)\Big) \end{aligned}$$

  is $1/2^n$ since $\sigma \oplus L$ is a bijection. Hence if the adversary makes a total of q oracle queries, bad_2 is set with probability at most $q^2/2^n$ by a simple union bound.
- bad_3 is set to 1 only if B is chosen from $\mathsf{right}(\Pi^*)$. At any given time the size of $\mathsf{right}(\Pi^*)$ is bounded by q, the total number of queries made by the adversary. Hence the total probability that bad_3 gets set is bounded by $q^2/2^n$.
- bad_4 is similar to bad_3: it is set to 1 only if A is chosen from $\mathsf{left}(\Pi^*)$.
- bad_5 is set to 1 only if $B \in \mathsf{right}(\Pi \cup \Pi^*)$. Note that when Z is uniform, so is B. So as above, on each call to $\mathcal{O}$ the probability that bad_5 is set is at most $q/2^n$, and the overall probability of bad_5 being set is at most $q^2/2^n$.

Since each bad_i flag is set to 1 with negligible probability, the two games are indistinguishable, and the construction satisfies the CCR security definition. □

Instantiations. Our main construction truncates a CCR from κ to $\kappa/2$ bits. Let the input X to the CCR be κ bits and split it into two halves as $X = X_L \| X_R$. Our construction uses linear functions $L_{ab}(\Delta_L \| \Delta_R) = (a\Delta_L \oplus b\Delta_R) \| 0^{\kappa/2}$, for $a, b \in \{0, 1\}$.

Our optimization in Section 6.2 uses a single κ-bit CCR to derive two calls to a $\kappa/2$-bit CCR, *each with possibly different linear transformations.* This corresponds to a κ-bit CCR with linear functions $L_{abcd}(\Delta_L \| \Delta_R) = (a\Delta_L \oplus b\Delta_R) \| (c\Delta_L \oplus d\Delta_R)$, for $a, b, c, d \in \{0, 1\}$.

Our construction requires an XOR-homomorphic function σ such that $\sigma(X) \oplus L(X)$ is invertible for any L in this class. The simplest examples of such a σ is as follows: split the input X into two halves $X_L \| X_R$, then define $\sigma(X_L \| X_R) = (\alpha X_L) \| (\alpha X_R)$, where α is any fixed element in $\mathrm{GF}(2^{\kappa/2}) \setminus \mathrm{GF}(2^2)$, and the multiplication is in $\mathrm{GF}(2^{\kappa/2})$. Then we get

$$\sigma(\Delta) \oplus L_{abcd}(\Delta) = \begin{bmatrix} \alpha \oplus a & b \\ c & \alpha \oplus d \end{bmatrix} \begin{bmatrix} \Delta_L \\ \Delta_R \end{bmatrix}$$

Since $a, b, c, d \in \{0, 1\}$, the determinant of this matrix is a degree-2 polynomial in α with binary coefficients. And since α is chosen not to be in $\mathrm{GF}(2^2)$, it is not the root of any such degree-2 polynomial. Hence the determinant of the matrix is nonzero, and $\sigma(\Delta) \oplus L_{abcd}(\Delta)$ is invertible.

A.2 Randomized Tweakable CCR

Lemma 6. *Define $H^*_{k,U}$ as:*

$$H^*_{k,U}(X, \tau) = F_k(X \oplus U(\tau))$$

and denote the family of all such function as $\mathcal{H}^* = \{H_{k,U} \mid k \in \{0,1\}^\kappa, U \in \mathcal{U}\}$. If F is a secure PRF, and for every (fixed) k, F_k is CCR for $\mathcal{L}$, and $\{(X, \tau) \mapsto X \oplus U(\tau) \mid U \in \mathcal{U}\}$ is a universal hash family, then $\mathcal{H}^*$ is a secure RTCCR hash family for $\mathcal{L}$.

Proof. Consider an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in the "real" RTCCR experiment. Its first phase $\mathcal{A}_1$ makes queries to $H^* \in \mathcal{H}^*$ and to $\mathcal{O}^{\mathsf{rtccr}}_{H^*,\Delta}$ before learning the parameters k and U.

Claim: it is with only negligible probability that $\mathcal{A}_1$ makes distinct queries (X, τ), (X', τ') to its oracles such that $X \oplus U(\tau) = X' \oplus U(\tau')$.

Proof of claim: Consider the following reduction algorithm M which has oracle access to the construction $H^*_{k,U}$, with k and U uniform. It internally runs $\mathcal{A}_1$ and chooses a random Δ. When $\mathcal{A}_1$ queries its H^* oracle, M relays that query directly to its oracle. When $\mathcal{A}_1$ queries its $\mathcal{O}$ oracle on (X, τ, L), M queries its H^* oracle and returns $H^*_{k,U}(\Delta \oplus X, \tau) \oplus L(\Delta)$.

Clearly M perfectly simulates the view of $\mathcal{A}_1$. Since F_k is a PRF and $(X, \tau) \mapsto X \oplus U(\tau)$ is a universal hash function, the construction $H^*_{k,U}(X, \tau) = F_k(X \oplus U(\tau))$ is exactly a Carter-Wegman MAC/PRF [BHK+99]. The usual security proof of Carter-Wegman establishes that it is only with negligible probability that an adversary with oracle access to the MAC causes an internal collision in the universal hash.

Now consider the following reduction algorithm M' which has an oracle for just F_k and the (plain) CCR oracle $\mathcal{O}^{\mathsf{ccr}}_{F_k,\Delta}$. It internally runs $\mathcal{A}_1$ and chooses a random $U \leftarrow \mathcal{U}$. When $\mathcal{A}_1$ queries its H^* oracle at (X, τ), M' queries its F_k oracle at $X \oplus U(\tau)$. When $\mathcal{A}_1$ queries its $\mathcal{O}^{\mathsf{rtccr}}$ oracle at (X, τ, L), M' queries its $\mathcal{O}^{\mathsf{ccr}}_{F_k,\Delta}$ oracle at $(X \oplus U(\tau), L)$. Furthermore, M' aborts if two distinct queries ever result in the same $X \oplus U(\tau)$. After $\mathcal{A}_1$ finishes, M' runs $\mathcal{A}_2$ and gives it k and U.

Note that

$$\begin{aligned}\mathcal{O}^{\mathsf{ccr}}_{F_k,\Delta}(X \oplus U(\tau), L) &= F_k(\Delta \oplus X \oplus U(\tau)) \oplus L(\Delta) \\ &= H^*_{k,U}(X \oplus \Delta, \tau) \oplus L(\Delta) \\ &= \mathcal{O}^{\mathsf{rtccr}}_{H^*,\Delta}(X, \tau, L)\end{aligned}$$

and so, conditioned on M' not aborting, it perfectly simulates the view of $(\mathcal{A}_1, \mathcal{A}_2)$. By the argument above, M' aborts with only negligible probability. Note that M' is a valid adversary in the (plain) CCR experiment as it never repeats an argument $X \oplus U(\tau)$ to its $\mathcal{O}^{\mathsf{ccr}}_{F_k,\Delta}$ oracle. It is important that the CCR experiment allows F_k to be public, so that M' can give k to $\mathcal{A}_2$.

By the fact that F_k is CCR (for fixed k), this interaction is indistinguishable from one in which $\mathcal{O}^{\mathsf{ccr}}_{F_k,\Delta}$ is replaced with a random function. Making such a replacement causes $\mathcal{A}_1$'s $\mathcal{O}^{\mathsf{rtccr}}$ oracle to be replaced by a random function. The fact that this view for $(\mathcal{A}_1, \mathcal{A}_2)$ is indistinguishable shows that $\mathcal{H}^*$ is RTCCR. □

Instantiation A simple choice of U is multiplication (in $\mathrm{GF}(2^\kappa)$) by a random field element. Consider the map $(X, \tau) \mapsto X \oplus U(\tau) = X \oplus u \cdot \tau$ where $u \leftarrow \mathrm{GF}(2^\kappa)$. For fixed $(X, \tau) \neq (X', \tau')$, we have the following cases:

- If $\tau = \tau'$ then $X \neq X'$ so $X \oplus u\tau \neq X' \oplus u\tau'$.
- If $\tau \neq \tau'$ then $\Pr[X \oplus u\tau = X' \oplus u\tau'] = \Pr[(X \oplus X')(\tau \oplus \tau')^{-1} = u] = 1/2^\kappa$ since u is uniform.

In either case, the probability of (X, τ) and (X', τ') being a collision is negligible, so the map is a universal hash.

In the reasonable event that all tweaks are at most $\kappa/2$ bits, we can interpret τ as an element of $\mathrm{GF}(2^{\kappa/2})$ and define $U(\tau) = (u_L\tau) \| (u_R\tau)$ where u_L, u_R are independently uniform in $\mathrm{GF}(2^{\kappa/2})$. For $\kappa = 128$, this way of defining U involves more efficient 64-bit operations.

B Randomizing the Entire Control Matrix

We used computerized linear algebra to generate a basis for the 14-dimensional linear space of matrices $R_\$$ such that $KR_\$ = 0$.

$$R_\$ \leftarrow \text{span} \left\{ \begin{array}{cccc}
\left[\begin{array}{cc|cc|cc} 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&1&0&1&0\\ 1&0&0&0&0&0\\ \hline 0&0&1&0&0&0\\ 1&0&1&0&1&0\\ \hline 1&0&0&0&1&0\\ 0&0&1&0&1&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&1&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&1&0&1\\ 0&1&0&0&0&0\\ \hline 0&0&0&1&0&0\\ 0&1&0&1&0&1\\ \hline 0&1&0&0&0&1\\ 0&0&0&1&0&1 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&1&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ \hline 0&0&1&0&0&0\\ 0&0&1&0&0&0\\ \hline 0&0&0&0&0&0\\ 1&0&1&0&0&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&1&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ \hline 0&0&0&1&0&0\\ 0&0&0&1&0&0\\ \hline 0&0&0&0&0&0\\ 0&1&0&1&0&0 \end{array}\right], \\
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ \hline 0&0&1&0&1&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 1&0&1&0&1&0\\ \hline 0&0&1&0&1&0\\ 0&0&1&0&1&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ \hline 0&0&0&1&0&1\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&1&0&1&0&1\\ \hline 0&0&0&1&0&1\\ 0&0&0&1&0&1 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&1&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&1&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{array}\right], \\
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 1&0&1&0&1&0\\ 1&0&0&0&0&0\\ \hline 0&0&1&0&0&0\\ 1&0&1&0&1&0\\ \hline 1&0&0&0&1&0\\ 0&0&1&0&1&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&1&0&1&0&1\\ 0&1&0&0&0&0\\ \hline 0&0&0&1&0&0\\ 0&1&0&1&0&1\\ \hline 0&1&0&0&0&1\\ 0&0&0&1&0&1 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&1&0&1&0\\ \hline 0&0&0&0&0&0\\ 0&0&1&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&1&0&1&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&1&0&1\\ \hline 0&0&0&0&0&0\\ 0&0&0&1&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&1&0&1 \end{array}\right], \\
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 1&0&0&0&1&0\\ 0&0&0&0&0&0\\ \hline 1&0&0&0&1&0\\ 0&0&0&0&0&0 \end{array}\right], &
\left[\begin{array}{cc|cc|cc} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ \hline 0&1&0&0&0&1\\ 0&0&0&0&0&0\\ \hline 0&1&0&0&0&1\\ 0&0&0&0&0&0 \end{array}\right]
\end{array} \right\}$$

As you can see, projecting this basis unto a single evaluation case's control matrix $R_{\$ij}$ will always give a uniformly random result, so this technique is sufficient to hide the entire control matrix when using control indirection. This is easiest to see with the case $i = j = 0$ (the top two rows of each matrix) because of the particular basis we chose, which is in echelon form when the matrices are written as vectors in row-major order.