Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates

Samee Zahur* Mike Rosulek† David Evans*

September 28, 2014

Abstract

The well-known classical constructions of garbled circuits use four ciphertexts per gate, although various methods have been proposed to reduce this cost. The best previously known methods for optimizing AND gates (two ciphertexts; Pinkas et al., ASIACRYPT 2009) and XOR gates (zero ciphertexts; Kolesnikov & Schneider, ICALP 2008) were incompatible, so most implementations used the best known method compatible with free-XOR gates (three ciphertexts; Kolesnikov & Schneider, ICALP 2008). In this work we show how to simultaneously garble AND gates using two ciphertexts and XOR gates using zero ciphertexts, resulting in smaller garbled circuits than any prior scheme. The main idea behind our construction is to break an AND gate into two *half-gates* — AND gates for which one party knows one input. Each half-gate can be garbled with a single ciphertext, so our construction uses two ciphertexts for each AND gate while being compatible with free-XOR gates. The price for the reduction in size is that the evaluator must perform two cryptographic operations per AND gate, rather than one as in previous schemes. We experimentally demonstrate that our garbling scheme leads to an overall decrease in time (up to 25%), bandwidth (up to 33%), and energy use (up to 20%) over several benchmark applications. We also initiate a study of lower bounds for garbled gate size, and show that our construction is optimal for a large class of garbling schemes encompassing all known practical garbling techniques.

1 Introduction

Yao's garbled circuit technique remains one of the most promising and actively studied methods for secure multi-party computation. The first implementation of secure two-party computation (2PC) [26] used Yao's basic garbled circuit approach, and it remains the primary (but not only) paradigm for the many 2PC implementations that have been developed over the past ten years [25, 28, 10, 14, 21, 12]. Because the generation and execution of gates benefits from advances in processor speed (in particular, hardware support for cryptographic operations) as well as the increasing availability of large numbers of cores, the computation time and cost for garbled circuit protocols has dropped dramatically. Thus, the main bottleneck for 2PC protocols is network bandwidth which is predominantly due to the transmission of garbled gates. A great deal of optimizations in 2PC have specifically focused on reducing the size of the garbled circuits themselves [27, 20, 19] and reducing the number of circuits required (in the case of malicious security) [24, 29, 22, 15, 6]. Our work reduces the overall size of garbled circuits by reducing the amount of data that needs to be transferred for each garbled gate.

1.1 Background

We assume some familiarity with garbled circuit constructions (for a comprehensive treatment of Yao's classical construction see Lindell & Pinkas [23]). In a garbled gate, each wire of the (boolean) circuit is associated with two random strings/keys called *wire labels* which encode TRUE and FALSE. In the "classical" construction of garbled circuits, the sender provides a garbled truth table for each gate, where each combination of input wire labels is used to encrypt the appropriate output wire label. Hence, there are four "ciphertexts" per gate — one for each input

*University of Virginia. {samee,evans}@virginia.edu. Partially funded by research grants from the National Science Foundation and Air Force Office of Scientific Research.

†Oregon State University. rosulekm@eecs.oregonstate.edu. Supported by NSF grant CCF-1149647.

combination to the gate — and the evaluator who only knows one label for each input wire can only open one of them. In general, we will be measuring the size of a garbled gate in units of such "ciphertexts."

We now give a brief history of work reducing the data needed to transmit a garbled gate, with a summary given in Table 1. In the **point-and-permute** optimization, introduced by Beaver, Micali & Rogaway [3], a *select bit* is appended to each wire label, so that the two labels on each wire have opposite select bits. The association between select bits and logical truth values is random and secret, but the garbled truth table can be arranged by these public select bits. While the result is still four ciphertexts per gate, the ciphertexts no longer need to be from a CPA-secure encryption scheme (and this indeed leads to a reduction in concrete size). Rather, they can be of the form $H(A\|B)\oplus C$, where A, B, and C are wire labels, and H is a hash function or key-derivation function. Further, instead of trying all four ciphertexts, the evaluator can simply select the appropriate one based on the select bits of visible wire labels.

Naor, Pinkas & Sumner introduced **garbled row-reduction** as a way of reducing the number of ciphertexts per gate [27]. The main observation is that, instead of choosing random wire labels for each wire, one of the wire labels can be chosen as $H(A\|B)$, where A and B are labels of the input wires. In doing so, one of the four ciphertexts in each gate (say, the first one) will always be the all-zeroes string. As such, it does not need to be sent, and three ciphertexts suffice per gate. We call this method *GRR3*. Going even further, Pinkas et al. [28] describe a way to further reduce each gate to 2 ciphertexts, applying a polynomial interpolation at each gate. We call this method *GRR2*.

Kolesnikov & Schneider [20] introduced the **free-XOR** technique. The idea is to choose wire labels of the form $(A, A \oplus R)$ for every wire, where R is secret and common to all wires. That way, when an evaluator has one of $(A, A \oplus R)$ and one of $(B, B \oplus R)$, he can simply XOR the two wire labels. The result will be either C or $C \oplus R$ (where $C = A \oplus B$), according to the logic of an XOR gate. Hence, no ciphertexts are required at all for an XOR gate. The technique is compatible with GRR3 for AND gates, but not GRR2. The reason is that the GRR2 technique chooses both output wire labels of a gate as fixed pseudorandom functions of the input wire labels. Hence, it is not possible to guarantee that the output wire labels are of the form $(C, C \oplus R)$ for some pre-specified R.

Recently, a generalization of free-XOR caled "fleXOR" (flexible XOR) was proposed by Kolesnikov, Mohassel & Rosulek [19]. In fleXOR, an XOR gate can be garbled using 0, 1, or 2 ciphertexts, depending on some structural/combinatorial properties of the circuit. However, fleXOR can be made compatible with GRR2 applied to the AND gates. For circuits with many AND gates, this method results in smaller circuits than with free-XOR (while the construction can actually collapse to free-XOR in other cases).

1.2 Our Contributions

Half-gates. We present a method for garbling AND gates that requires only 2 ciphertexts. However, unlike the GRR2 method, our method is compatible with free-XOR. That is, our method can guarantee that the output wires of an AND gate are indeed of the form $(C, C \oplus R)$, when the input wires are also of this form.

The main insight is to employ what we call **half-gates**: AND gates for which one party knows one of the inputs. We show how to garble generator half-gates and evaluator half-gates using one ciphertext each, in a way that is compatible with free-XOR. We then show how an AND gate can be written as a combination of XORs and two half-gates of opposite orientations. Hence, the resulting AND gate uses only two ciphertexts in combination with free-XOR. We prove the security of our scheme in Section 4.

Our half-gate technique leads to smaller garbled circuits than *all* previous methods, for *all* circuits (i.e., our row of Table 1 dominates all other rows). For many circuits (i.e., those for which free-XOR previously gave the smallest garbled circuits), our work gives a 33% reduction in garbled circuit size (and thus a similar reduction in cost for most protocols that rely on garbled circuits). This leads to reductions in overall latency (up to 25% in our benchmarks), as well as energy (which is the primary concern for data centers as well as mobile devices) since the extra computation required to compute the hash function twice is more than offset by the energy savings of reduced bandwidth. We provide experimental results in Section 5.

Privacy-free garbling. Frederiksen, Nielsen & Orlandi [8] showed that garbling schemes that satisfy only the *authenticity* security property (i.e., not the *privacy* property) can be significantly smaller than their fully-secure counterparts. These privacy-free schemes are useful in settings where the evaluator knows the entire (cleartext) input to the garbled circuit, as in the highly efficient zero-knowledge proof protocol of Jawurek, Kerschbaum & Orlandi [18].

Table 2 summarizes the three privacy-free garbling schemes introduced by Frederiksen, Nielsen & Orlandi [8], which are adaptations of fully-secure schemes. Their *GRR1* construction garbles all gates at a cost of one ciphertext

technique	size per gate		calls to H per gate			
			generator		evaluator	
	XOR	AND	XOR	AND	XOR	AND
classical [31]	4	4	4	4	4	4
point-permute [3]	4	4	4	4	1	1
row reduction (GRR3) [27]	3	3	4	4	1	1
row reduction (GRR2) [28]	2	2	4	4	1	1
free XOR + GRR3 [20]	0	3	0	4	0	1
fleXOR [19]	$\{0,1,2\}$	2	$\{0,2,4\}$	4	$\{0,1,2\}$	1
half gates [this work]	0	2	0	4	0	2

Table 1: Optimizations of garbled circuits. Size is reported in number of "ciphertexts" (multiples of k bits).

technique	size per gate		calls to H per gate			
			sender		receiver	
	XOR	AND	XOR	AND	XOR	AND
row reduction (GRR1)	1	1	0	3	0	1
free XOR + GRR2	0	2	0	3	0	1
fleXOR	$\{0,1,2\}$	1	0	3	0	1
half gates [this work]	0	1	0	2	0	1

Table 2: Optimizations of **privacy-free** garbled circuits. Size is reported in number of ciphertexts (multiples of k bits). The three prior privacy-free garbling schemes are from [8].

each. Their free-XOR adaptation garbles AND gates at a cost of two ciphertexts each (with XOR gates free). Their fleXOR adaptation garbles AND gates using one ciphertext each, with XOR gates costing 0, 1, or 2 ciphertexts each.

In Section 6, we show that our approach with half-gates also gives a similar improvement in this setting. We can simply garble all AND gates using our evaluator-half-gate. In this setting, the evaluator knows both inputs to all gates, but we only need to take advantage of her knowledge of one of the inputs to reduce the size of garbled AND gates to one ciphertext. Overall, we achieve a privacy-free garbled circuit containing one ciphertext per AND gate, and no ciphertexts for XOR gates. As above, our half-gates approach is strictly better than all previous constructions. For example, we reduce the size of a privacy-free garbled circuit for AES by 50%.

Optimality. For prior garbling schemes that we described above, it was always possible to reduce the size of garbled AND gates by one ciphertext by sacrificing compatibility with free-XOR. Given that we can now garble AND gates with two ciphertexts in a way that is compatible with free-XOR, one might wonder whether it is possible to garble an AND gate with just one ciphertext, in a way that is incompatible with free-XOR.[1] In Section 7, we show that in a reasonable model that captures all existing techniques, it is not possible to garble an AND gate (with privacy) using just one ciphertext, even if compatibility with free-XOR is sacrificed. Hence, our construction gives **optimally sized garbled circuits**, among garbling schemes whose gate-by-gate operations fall within our model.

To show optimality, we first introduce a new methodology for stating and proving such quantitative lower bounds on the size of garbled gates.We observe that all existing techniques for practical garbling (including our own) are *linear*

[1]When thinking about the size of garbled gates, instead of thinking about free-XOR compatibility, it turns out to be more instructive to think about the degrees of freedom available for choosing a gate's output wire labels. In the classical scheme that uses four ciphertexts, both output wire labels can be arbitrary; there are two degrees of freedom. In the GRR3 scheme that uses three ciphertexts, one of the output wire labels is fixed as soon as the input wire labels are fixed (since one output wire label is a hash of some input wire labels). Hence there is just one degree of freedom, for choosing the other wire label, and this is typically exploited to ensure free-XOR compatibility. In the GRR2 scheme that uses two ciphertexts, both output wire labels are fixed as soon as the input wire labels are fixed; there are no degrees of freedom. In our construction also, there are no degrees of freedom on the output wire labels. One is chosen as a hash of input wire labels, and, furthermore, the two output wire labels must have the same offset as one of the input wires.

in a certain sense. We formalize these techniques in a linear class of garbling schemes, and show that these schemes require two ciphertexts to garble a single AND gate. We similarly show that garbling a *privacy-free* AND gate requires one ciphertext in this model. These lower bounds suggest that any practical improvement over our scheme will require a dramatically different approach to garbled circuits in general.

2 Preliminaries

We use the *garbling schemes* abstraction introduced by Bellare, Hoang, and Rogaway [5]. Roughly speaking, a garbling scheme consist of the following algorithms:[2]

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

The correctness property is that, if $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$ then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x)$ for all x. Additionally, several security properties are described:

Privacy ($\mathsf{prv.sim}_{\mathcal{S}}$): Intuitively, the collection (F, X, d) should not reveal any more information about x than $f(x)$. More concretely, there must exist a simulator $\mathcal{S}$ that takes input $(1^k, f, f(x))$ and whose output is indistinguishable from (F, X, d) generated the usual way.

Obliviousness ($\mathsf{obv.sim}_{\mathcal{S}}$): Intuitively, (F, X) should reveal no information about x. More concretely, there must exist a simulator $\mathcal{S}$ that takes input $(1^k, f)$ and whose output is indistinguishable from (F, X) generated the usual way.

Authenticity (aut): Given input (F, X) alone, no adversary should be able to produce $\tilde{Y} \neq \mathsf{Ev}(F, X)$ such that $\mathsf{De}(d, \tilde{Y}) \neq \perp$, except with negligible probability.

A garbling scheme may satisfy any combination of these security properties. We refer the reader to Bellare, Hoang, and Rogaway [5] for the complete treatment of garbling schemes and further relations among the security properties.

3 Half-Gates Garbling Scheme

First, we give a high-level and self-contained overview of our construction of *half-gates*, which form the basis of our improved garbling schemes. Then, we present the details more formally.

3.1 Approach

Recall that a half-gate is a garbled AND gate for which one of the parties knows one of the inputs (in the clear). Let's say we want to compute the gate $c = a \wedge b$. We are in the free-XOR setting, so let $(A, A \oplus R)$ and $(B, B \oplus R)$ denote the input wire labels to this gate, and $(C, C \oplus R)$ denote the output wire labels, with A, B, and C each encoding FALSE. R is the free-XOR offset common to all wires. Finally, H will denote a hash (or key derivation) function.

We describe how to construct half-gates for two cases: when the garbled-circuit generator knows one of the inputs, and when the evaluator knows one of the inputs.

[2] The formalization of [5] allows for garbling of any form of computation. Here we specialize the notation for garbling *circuits*, as this is all that is required in our work.

Generator half-gate. We consider the case of an AND gate $c = a \wedge b$, where a and b are intermediate wires in the circuit and the generator somehow knows in advance what the value a will be. Conceptually, when $a = 0$, the generator will garble a unary gate that always outputs false; when $a = 1$, the generator will garble a unary identity gate. This idea was also used implicitly by Kolesnikov and Schneider [20, Fig. 2], in the context of programming components of a universal circuit.

Hence, the generator produces the two ciphertexts:

$$H(B) \oplus C$$
$$H(B \oplus R) \oplus C \oplus aR$$

These are then suitably permuted according to the select bits of B. The evaluator takes a hash of her wire label for B and decrypts the appropriate ciphertext. If $a = 0$, she will obtain output wire label C in both values of b. If $a = 1$, the evaluator will obtain either C or $C \oplus R$, depending on the bit b. Intuitively, the evaluator will never know both B and $B \oplus R$, hence the other ciphertext appears completely random.

Next, we eliminate one of the ciphertexts by applying a standard idea of garbled row-reduction [27]. Instead of choosing C uniformly, we choose C so that the first of the two ciphertexts is the all-zeroes ciphertext (i.e., we choose C as $H(B)$, $H(B \oplus R)$, or $H(B \oplus R) \oplus R$, depending on the select bits and the value a). As such, the first ciphertext does not actually need to be sent; in the case where the evaluator would have decrypted the first ciphertext, she can imagine it to be the all-zeroes string. Overall, this garbled half-gate consists of one ciphertext (k bits). The generator calls H twice; the evaluator calls H once.

Evaluator half-gate. We now consider the case of an AND gate $c = a \wedge b$, where a and b are intermediate wires in the circuit and the evaluator will somehow already know the value of a at the time of evaluation.

We exploit the fact that the evaluator can then behave differently based on the truth value of a. Intuitively, when $a = 0$ the evaluator should always obtain output wire label C; when $a = 1$, it is enough for the evaluator to obtain $\Delta = C \oplus B$. She can then XOR Δ with her other wire label (either B or $B \oplus R$) to obtain either C or $C \oplus R$ appropriately.

Hence, the generator provides the two ciphertexts:

$$H(A) \oplus C$$
$$H(A \oplus R) \oplus C \oplus B$$

The ciphertexts do not have to be permuted here. They can be arranged according to the truth value of a as shown here, since the evaluator already knows a. If $a = 0$, the evaluator uses wire label A to decrypt the first ciphertext. If $a = 1$, she uses wire label $A \oplus R$ to decrypt the second ciphertext and XORs the result with her wire label for b.

Again, we can remove the first ciphertext using garbled row-reduction. We choose $C = H(A)$ so that the first ciphertext becomes all-zeroes and is not sent. Overall, the cost of this garbled half-gate is the same as above: it consists of one ciphertext (k bits). The generator calls H twice; the evaluator calls H once.

Two halves make a whole. Now consider the case where we want to garble an AND gate $c = a \wedge b$ where both inputs are secret. Consider:

$$\begin{aligned} c &= a \wedge b \\ &= a \wedge (r \oplus r \oplus b) \\ &= (a \wedge r) \oplus (a \wedge (r \oplus b)) \end{aligned}$$

Suppose the generator chooses a uniformly random bit r. In that case, the first AND gate $(a \wedge r)$ can be garbled with a generator-half-gate. If we further arrange for the evaluator to learn the value $r \oplus b$, then the second AND gate $(a \wedge (r \oplus b))$ can be garbled with an evaluator-half-gate. Leaking this extra bit $r \oplus b$ to the evaluator is safe, as it carries no information about the sensitive value b. The remaining XOR is free, and the total cost is two ciphertexts.

We can actually convey $r \oplus b$ to the evaluator without any overhead. The generator will choose r to be the select bit of the false wire label on wire b. For security, select bits of wires are chosen (pseudo)randomly already. Then when a particular value b is on that wire, the evaluator will hold a wire label whose select bit is $b \oplus r$.

Thus, we garble a (full) AND gate with two ciphertexts, taking the XOR of two half-gates. The generator calls H four times; the evaluator calls H twice.

Computes: $f_G(v_a, p_b) :=$ $(v_a \oplus \alpha_a)(p_b \oplus \alpha_b) \oplus \alpha_c$
Before GRR and permutation: $H(W_a^0) \oplus f_G(0, p_b)R \oplus W_{Gc}^0$ $H(W_a^1) \oplus f_G(1, p_b)R \oplus W_{Gc}^0$
After GRR and permutation: $T_{Gc} \leftarrow H(W_a^0) \oplus H(W_a^1) \oplus (p_b \oplus \alpha_b)R$ $W_{Gc}^0 \leftarrow H(W_a^{p_a}) \oplus f_G(p_a, p_b)R$
Generator sends T_{Gc}

(a) Generator-side half gate: v_a known to generator.

Computes: $f_E(v_a, v_b \oplus p_b) :=$ $(v_a \oplus \alpha_a)(v_b \oplus p_b)$
Before GRR: $H(W_b^{p_b}) \oplus W_{Ec}^0$ $H(W_b^{p_b \oplus 1}) \oplus W_{Ec}^0 \oplus W_a^{\alpha_a}$
After GRR (permutation not needed): $T_{Ec} \leftarrow H(W_b^0) \oplus H(W_b^1) \oplus W_a^{\alpha_a}$ $W_{Ec}^0 \leftarrow H(W_b^{p_b})$
Generator sends T_{Ec}

(b) Evaluator-side half gate: $v_b \oplus p_b$ known to evaluator.

Figure 1: The construction of a non-free binary gate for computing $(v_a, v_b) \mapsto (v_a \oplus \alpha_a)(v_b \oplus \alpha_b) \oplus \alpha_c$, where $\alpha_a, \alpha_b, \alpha_c$ determines the type of the gate. After the two half-gates are evaluated, output label is obtained by computing $W_c = W_{Gc} \oplus W_{Ec}$

3.2 Details of Our Scheme

We now give a formal description of our garbling scheme, following the basic approach outlined above.

Notation and concepts. For a boolean circuit f, we associate each wire in the circuit with a numeric index. We let $\mathsf{Inputs}(f)$, $\mathsf{Outputs}(f)$, and $\mathsf{XorGates}(f)$ denote the set of wire indices of the input wires, output wires, xor gate output wires, respectively, in f. We abuse notation slightly and extend these functions as $\mathsf{Inputs}(\hat{F}), \mathsf{Outputs}(\hat{F})$ and $\mathsf{XorGates}(\hat{F})$, where $\hat{F}$ is a garbled version of f. We use v_i to denote the single-bit plaintext value of the ith wire in a circuit, when the input is understood from context. For non-input wires, we also refer to the *ith gate* to mean the logic gate whose output wire has index i.

Our garbling scheme follows standard paradigms of the free-XOR & point-and-permute optimizations. We use $W_i^0, W_i^1 \in \{0,1\}^k$ to denote the wire labels for FALSE and TRUE, respectively, on the ith wire. Here, and throughout the paper, k denotes the scheme's security parameter. For each wire label W, its least significant bit $\mathsf{lsb}\, W$ is reserved as a **select bit** that is used as in the point-and-permute technique. For the ith wire, define $p_i = \mathsf{lsb}\, W_i^0$. This value, which we call the *permute bit* of the wire, is a secret known only to the generator. Intuitively, when the evaulator holds a wire label for wire i whose select bit is s, that wire label is $W_i^{p_i \oplus s}$ — i.e., it corresponds to truth value $s \oplus p_i$. In the context of evaluating a garbled circuit, we typically omit the superscript from the wire label notation and write just W_i to indicate the fact that the evaluator does not in fact know v_i.

The value $R \in \{0,1\}^{k-1}1$ is a circuit-global, randomly chosen *free-XOR offset*; hence, $W_i^0 \oplus W_i^1 = R$ holds for each i in the circuit. We have $\mathsf{lsb}\, R = 1$ so that $\mathsf{lsb}\, W_i^0 \neq \mathsf{lsb}\, W_i^1$; i.e., complementary wires have opposite select bits.

Frequently, we will omit $\wedge$ and just juxtapose two symbols to indicate logical AND. So $ab = a \wedge b$. When a is a single bit and R is a long string, we write aR to mean R when $a = 1$ and $0^{|R|}$ when $a = 0$. We write sequences or tuples with a 'hat'; for example, $\hat{F} = (F_1, F_2, \ldots)$ or $\hat{X} = (X_1, X_2, \ldots)$.

Finally, we will use $H : \{0,1\}^k \times \mathbb{Z} \mapsto \{0,1\}^k$ to indicate a hash-function suitable for use in garbled circuits (see Section 4 for suitability criteria). In informal discussions, we will often shorten $H(W_i^b, j)$ to just $H(W_i^b)$, and it will be implicitly understood that we are using unique, but public, j for different groups of calls to H. In the formal descriptions, the value of j is always explicit.

Arbitrary gates. The approach just described can be used to garble any gate whose truth table contains an odd number of ones (e.g., AND, NAND, OR, NOR, etc.). All such gates can be expressed as the form

$$(v_a, v_b) \mapsto (\alpha_a \oplus v_a) \wedge (\alpha_b \oplus v_b) \oplus \alpha_c$$

for constants $\alpha_a, \alpha_b, \alpha_c$. For example, setting all to 0 results in an AND gate; setting all to 1 results in an OR gate. These α values need not (but can) be secret. We describe the general construction of these gates in Figure 1. We note that the evaluator's logic does not depend on the α values.

procedure $\mathsf{Gb}(1^k, f)$:
  $R \twoheadleftarrow \{0,1\}^{k-1}1$
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^0 \twoheadleftarrow \{0,1\}^k$
    $W_i^1 \leftarrow W_i^0 \oplus R$
    $e_i \leftarrow W_i^0$
  for $i \notin \mathsf{Inputs}(f)$ {*in topo. order*} **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(f, i)$
    if $i \in \mathsf{XorGates}(f)$ **then**
      $W_i^0 \leftarrow W_a^0 \oplus W_b^0$
    else
      $(W_i^0, T_{Gi}, T_{Ei}) \leftarrow \mathsf{GbAnd}(W_a^0, W_b^0)$
      $F_i \leftarrow (T_{Gi}, T_{Ei})$
    end if
    $W_i^1 \leftarrow W_i^0 \oplus R$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i \leftarrow \mathsf{lsb}(W_i^0)$
  return $(\hat{F}, \hat{e}, \hat{d})$

private procedure $\mathsf{GbAnd}(W_a^0, W_b^0)$:
  $p_a \leftarrow \mathsf{lsb}\, W_a^0$; $p_b \leftarrow \mathsf{lsb}\, W_b^0$
  $j \leftarrow \mathsf{NextIndex}()$; $j' \leftarrow \mathsf{NextIndex}()$
  {*First half gate*}
  $T_G \leftarrow H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$
  $W_G^0 \leftarrow H(W_a^0, j) \oplus p_a T_G$
  {*Second half gate*}
  $T_E \leftarrow H(W_b^0, j') \oplus H(W_b^1, j') \oplus W_a^0$
  $W_E^0 \leftarrow H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  {*Combine halves*}
  $W^0 \leftarrow W_G^0 \oplus W_E^0$
  return (W^0, T_G, T_E)

procedure $\mathsf{En}(\hat{e}, \hat{x})$:
  for $e_i \in \hat{e}$ **do**
    $X_i \leftarrow e_i \oplus x_i R$
  return $\hat{X}$

procedure $\mathsf{De}(\hat{d}, \hat{Y})$:
  for $d_i \in \hat{d}$ **do**
    $y_i \leftarrow d_i \oplus \mathsf{lsb}\, Y_i$
  return $\hat{y}$

procedure $\mathsf{Ev}(\hat{F}, \hat{X})$:
  for $i \in \mathsf{Inputs}(\hat{F})$ **do**
    $W_i \leftarrow X_i$
  for $i \notin \mathsf{Inputs}(\hat{F})$ {*in topo. order*} **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(\hat{F}, i)$
    if $i \in \mathsf{XorGates}(\hat{F})$ **then**
      $W_i \leftarrow W_a \oplus W_b$
    else
      $s_a \leftarrow \mathsf{lsb}\, W_a$; $s_b \leftarrow \mathsf{lsb}\, W_b$
      $j_1 \leftarrow \mathsf{NextIndex}()$; $j_2 \leftarrow \mathsf{NextIndex}()$
      $(T_{Gi}, T_{Ei}) \leftarrow F_i$
      $W_{Gi} \leftarrow H(W_a, j_1) \oplus s_a T_{Gi}$
      $W_{Ei} \leftarrow H(W_b, j_2) \oplus s_b(T_{Ei} \oplus W_a)$
      $W_i \leftarrow W_{Gi} \oplus W_{Ei}$
    end if
  for $i \in \mathsf{Outputs}(\hat{F})$ **do**
    $Y_i \leftarrow W_i$
  return $\hat{Y}$

Figure 2: Our complete garbling scheme. NextIndex is a stateful procedure that simply increments an internal counter.

Following the description in Section 3.1, we garble each gate using a composition of two half-gates. Conceptually, W_{Gi}^b and W_{Ei}^b denote the output wire labels for these two half-gates (generator-side and evaluator-side, respectively) that comprise the ith gate. The final logical output wire label for the ith gate is then set to be $W_i^0 = W_{Gi}^0 \oplus W_{Ei}^0$. Similarly, we use T_{Gi} and T_{Ei} to denote the single garbled row transmitted for each half gate used in the ith gate.

The first rows of Figure 1 show the function being computed by each half gate. In (a), generator knows p_b while in (b) the evaluator knows $v_b \oplus p_b = \mathsf{lsb}\, W_b$. The second rows show the two ciphertexts of each half-gate, before they are permuted according to their select bits (in case of (a)) and before garbled row reduction (GRR) is applied. Here, we have expanded $W_{Gc}^{f(x,p_b)}$ to $W_{Gc}^0 \oplus f(x, p_b)R$ to make the row reduction clearer in the next step. The third rows show the final result.

The complete scheme. The full garbling procedure for an entire circuit is shown in Figure 2. For simplicity of discussion and proof, the we assume all gates are either AND or XOR.

4 Security

We now prove the security of our scheme, using the $\mathsf{prv.sim}_{\mathcal{S}}$ and $\mathsf{obv.sim}_{\mathcal{S}}$ security definitions of Bellare, Hoang, and Rogaway [5]. The scheme shown in Figure 2 does not provide authenticity, simply because authenticity is not required in many use cases including semi-honest Yao's circuits. However, there are well-known, standard modifications to the decoding procedure that can add authenticity, which we describe separately in Section 4.3. Finally, since we only consider circuits with just AND and XOR gates, everything about the function f is public and we do not define a separate function $\Phi(f)$ to extract public information about f.

4.1 Circular Correlation Robustness for Naturally Derived Keys

We first describe the security property required of the hash/key-derivation function H. Roughly speaking, we can use either a circular-correlation-robust hash function, as defined by Choi et al. [7], or a Davis-Meyer construction in the ideal random permutation model [4]. Note that a result of using half gates is we need arguably simpler single-key functions instead of the previously proposed dual-key ones. So, we first present the single-key analogs of these two definitions. Then we define a weaker notion of security that is satisfied by both these classes of hash functions. Functions satisfying this new notion of security will be said to have circular correlation robustness *for naturally derived keys*. Finally we show that our garbling scheme is secure given any hash function that satisfies this new, weaker notion of security.

Circular correlation robustness. We revisit the definition of circular correlation robustness. The definition is the same as the one introduced in [7], except that we are able to simplify the notation for H that takes only one wire label / key. Given a hash function H, we define two oracles:

- $\mathsf{Circ}_R(x, i, b) = H(x \oplus R, i) \oplus bR$, where $R \in \{0,1\}^{k-1}1$
- $\mathsf{Rand}(x, i, b)$: random function with k-bit output.

Definition 1. *Say that a sequence of oracle queries of the form (x, i, b) is* legal *if the same value of (x, i) is never queried with different values of b. Then H is* circular correlation robust *if, for all all polynomial-time adversaries $\mathcal{A}$ making legal queries,*

$$\left| \Pr_R[\mathcal{A}^{\mathsf{Circ}_R}(1^k) = 1] - \Pr_{\mathsf{Rand}}[\mathcal{A}^{\mathsf{Rand}}(1^k) = 1] \right| \text{ is negligible.}$$

The restriction to *legal* queries prevents the adversary from trivially finding R. Note that for the single-key version here we do not need an extra parameter a to produce values of the form $H(x \oplus aR, i) \oplus bR$, since the definitions in Choi et al. [7] would have made it illegal to use $a = 0$ anyway.

Finally, we emphasize that the adversary is allowed unrestricted access to H. Thus, modeling H as a random oracle, the adversary has oracle access to H in addition to the oracle in the experiment. In the standard model, the adversary is allowed to depend arbitrarily on H.

Constructions from ideal permutations. Bellare et al. [4] construct a gate-level cipher in the ideal random permutation model. In this model, all parties have access to a randomly chosen permutation $\pi : \{0,1\}^k \to \{0,1\}^k$ and its inverse π^{-1}. This is meant to model a setting where a garbling scheme is based on AES with a (public) fixed key, which can be implemented very efficiently with AES-NI instructions.

Bellare et al. [4] do not abstract a concrete security property that their hash function must satisfy. Instead, they describe how to construct their hash function, and prove security of the entire garbling scheme directly from the underlying assumption of a random permutation. Our ultimate abstraction (robustness for naturally derived keys) can be seen as a formalization of the properties of H actually used in their proofs.

We first describe the hash function of [4], altered for our single-key setting:

Definition 2. *For a random permutation $\pi : \{0,1\}^k \mapsto \{0,1\}^k$, we define the hash function $H_\pi(x, i)$ to be $\pi(K) \oplus K$ where $K = 2x \oplus i$.*

For concreteness, $2x$ refers to doubling in GF(2^k). However, there are many alternative ways of constructing H_π from π, which do not affect our proof. We refer the reader to Bellare et al. [4] for these alternate constructions and how they affect the exact constants on the security bounds. We also point out that in the following, the adversary is assumed to have access to π and π^{-1}.

Our abstraction. We now define a security notion that is satisfied by both of the above constructions.

Definition 3. *Say that a sequence of queries of the form (x, i, b) to an oracle $\mathcal{O}$ are* natural *if they satisfy the following:*

- *for the qth query, we have $i = q$.*
- $b \in \{0,1\}$

- x *is* naturally derived*, meaning that it is obtained from one of these operations:*
  - $x \twoheadleftarrow \{0,1\}^k$
  - $x \leftarrow x_1 \oplus x_2$*, where* x_1 *and* x_2 *are naturally derived*
  - $x \leftarrow H(x_1, i)$*, where* x_1 *is naturally derived and* $i \in \mathbb{Z}$
  - $x \leftarrow \mathcal{O}(x_1, i, b)$ *where* x_1 *is naturally derived.*

Then H *is* circular correlation robust for natural keys *if, for all all polynomial-time adversaries* $\mathcal{A}$ *making natural queries,*

$$\left| \Pr_R[\mathcal{A}^{\mathsf{Circ}_R}(1^k) = 1] - \Pr_{\mathsf{Rand}}[\mathcal{A}^{\mathsf{Rand}}(1^k) = 1] \right| \text{ is negligible.}$$

Note that these restrictions only apply when querying $\mathcal{O}$ — the adversary is still allowed to make unrestricted queries to H directly (and π, π^{-1} in the ideal permutation model). While it is a weak notion of security (since the adversary is very restricted), it turns out to be enough to prove security of our garbling scheme (Section 4.2).

Achieving the definition. While it is evident that circular correlation robustness against naturally derived keys is a restricted version of circular correlation robustness defined in Definition 1, it may not be as obvious that the H_π ideal permutation construction satisfies this notion.

Intuitively, the purpose of the naturally-derived restrictions is to make it unlikely that the adversary can ever query $\mathcal{O}$ with both (x, i, b) and (x', i', b') where $2x \oplus i = 2x' \oplus i'$ even though $(x, i) \neq (x', i')$. That would have created a problem in the case where $\mathcal{O}$ uses H_π. This would in turn invoke $\pi(2x \oplus 2R \oplus i) = \pi(2x' \oplus 2R \oplus i')$. If the adversary uses $b \neq b'$ then the responses to these queries reveal R.

The proof that the H_π construction achieves our definition in the ideal permutation model basically follows directly from the security proofs in Bellare et al. [4]. There, the bulk of the proofs are devoted to bounding the probability of the adversary making a query of the above form. They use only the fact that wire labels in their constructions are naturally derived, in our terminology (or, at least, the obvious generalization of naturally-derived to the two-key setting).

Following their proofs, one can work out the advantage of an adversary that makes q queries to the oracle $\mathcal{O}$ and Q queries to π, π^{-1}, in our security game. The advantage comes out to be $O((qQ + q^2)/2^k)$. The quadratic terms in that expression come from the birthday bounds of hash functions with k-bit output. We did not derive the exact constants since, in practice, much larger constants are likely to arise when π is replaced by a concrete function (e.g. AES). In any case, it is negligible in k, and therefore satisfies our notion of security.

4.2 Proof of Privacy & Obliviousness

The first thing to note is that we can easily rewrite the scheme in Figure 2 such that it only uses R through the oracle Circ_R. In particular, we can rewrite the assignments to T_{Gi} and T_{Ei} as:

$$T_{Gi} \leftarrow H(W_a^0, j_1) \oplus \mathsf{Circ}_R(W_a^0, j_1, p_b)$$
$$T_{Ei} \leftarrow H(W_b^0, j_2) \oplus \mathsf{Circ}_R(W_b^0, j_2, 0) \oplus W_a^0$$

Moreover, observe that we are only ever invoking Circ_R with naturally derived keys, assuming NextIndex returns sequential integers. This is partly why we did not write the assignments to W_{Gi}^0 and W_{Ei}^0 in Figure 2 more naturally using if statements conditioned on p_a and p_b — we did not want to repeat j values between oracle calls. Second, we no longer need to explicitly use R anywhere in Gb outside of the oracle (W_i^1 values are no longer needed).

Theorem 4. *Our scheme satisfies the security notion of* $\mathsf{obv.sim}_{\mathcal{S}}$ *and* $\mathsf{prv.sim}_{\mathcal{S}}$ *with any* H *that has correlation robustness for naturally derived keys.*

Proof. The proof for $\mathsf{obv.sim}_{\mathcal{S}}$ is identical to that of $\mathsf{prv.sim}_{\mathcal{S}}$, except that the simulator does not receive $\hat{y}$ and does not need to compute $\hat{d}$. So we will only provide the proof for $\mathsf{prv.sim}_{\mathcal{S}}$. To prove indistinguishability between the simulator (Figure 3) and the real protocol (Figure 1) we use the following chain of hybrids:

1. $\mathcal{S} \equiv \mathcal{G}_1^{\mathsf{Rand}}$: Both generate uniformly random values for each component in $(\hat{F}, \hat{X}, \hat{d})$, and are therefore identically distributed. More concretely, $\mathcal{G}_1$ uses $\hat{x}$ to determine a truth value v_i on each wire (via evalWires). Yet these truth values $\hat{v}$ are used only as a superscript for W_i^v. We could have obtained the same result if we had named these variables W_i^0 for all i instead of $W_i^{v_i}$.

procedure $\mathcal{S}(1^k, f, \hat{y})$:
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^0 \twoheadleftarrow \{0,1\}^k$
    $X_i \leftarrow W_i^0$
  for $i \notin \mathsf{Inputs}(f)$ {*in topo. order*} **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(f, i)$
    if $i \in \mathsf{XorGates}(f)$ **then**
      $W_i^0 \leftarrow W_a^0 \oplus W_b^0$
    else
      $(W_i^0, T_{Gi}, T_{Ei}) \leftarrow \mathsf{SimAnd}(W_a^0, W_b^0)$
      $F_i \leftarrow (T_{Gi}, T_{Ei})$
    end if
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i \leftarrow \mathsf{lsb}\,(W_i^0) \oplus y_i$
  return $(\hat{F}, \hat{X}, \hat{d})$

procedure $\mathcal{G}_1^{\mathcal{O}}(1^k, f, \hat{x})$: // $\boxed{\mathcal{G}_2^{\mathsf{Circ}_R}}$
  $\hat{v} \leftarrow \mathsf{evalWires}(f, \hat{x})$
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^{v_i} \twoheadleftarrow \{0,1\}^k$; $\boxed{W_i^{\overline{v_i}} \leftarrow W_i^{v_i} \oplus R}$
    $X_i \leftarrow W_i^{v_i}$
  for $i \notin \mathsf{Inputs}(f)$ {*in topo. order*} **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(f, i)$
    if $i \in \mathsf{XorGates}(f)$ **then**
      $W_i^{v_i} \leftarrow W_a^{v_a} \oplus W_b^{v_b}$
    else
      $(W_i^{v_i}, T_{Gi}, T_{Ei}) \leftarrow \mathsf{SimAnd}_1^{\mathcal{O}}(W_a^{v_a}, W_b^{v_b}, v_a, v_b)$
      $F_i \leftarrow (T_{Gi}, T_{Ei})$
    end if
    $\boxed{W_i^{\overline{v_i}} \leftarrow W_i^{v_i} \oplus R}$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i \leftarrow \mathsf{lsb}\,(W_i^{v_i}) \oplus v_i$
  return $(\hat{F}, \hat{X}, \hat{d})$

procedure $\mathcal{G}_3(1^k, f, \hat{x})$:
  $R \twoheadleftarrow \{0,1\}^{k-1}1$
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^0 \twoheadleftarrow \{0,1\}^k$
    $W_i^1 \leftarrow W_i^0 \oplus R$
    $X_i \leftarrow W_i^{x_i}$
  for $i \notin \mathsf{Inputs}(f)$ {*in topo. order*} **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(f, i)$
    if $i \in \mathsf{XorGates}(f)$ **then**
      $W_i^0 \leftarrow W_a^0 \oplus W_b^0$
    else
      $(W_i^0, T_{Gi}, T_{Ei}) \leftarrow \mathsf{SimAnd}_3(W_a^0, W_b^0)$
      $F_i \leftarrow (T_{Gi}, T_{Ei})$
    end if
    $W_i^1 \leftarrow W_i^0 \oplus R$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i \leftarrow \mathsf{lsb}\,(W_i^0)$
  return $(\hat{F}, \hat{X}, \hat{d})$

private procedure $\mathsf{SimAnd}(W_a^0, W_b^0)$:
  $p_a \leftarrow \mathsf{lsb}\,W_a^0$; $p_b \leftarrow \mathsf{lsb}\,W_b^0$
  $j \leftarrow \mathsf{NextIndex}()$; $j' \leftarrow \mathsf{NextIndex}()$
  $T_G \leftarrow H(W_a^0, j) \oplus \mathsf{Rand}(W_a^0, j, p_b)$
  $W_G^0 \leftarrow H(W_a^0, j) \oplus p_a T_G$
  $T_E \leftarrow H(W_b^0, j') \oplus \mathsf{Rand}(W_b^0, j', 0) \oplus W_a^0$
  $W_E^0 \leftarrow H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  $W^0 \leftarrow W_G^0 \oplus W_E^0$
  return (W^0, T_G, T_E)

private procedure $\mathsf{SimAnd}_1^{\mathcal{O}}(W_a^{v_a}, W_b^{v_b}, v_a, v_b)$:
  $s_a \leftarrow \mathsf{lsb}\,W_a^{v_a}$; $s_b \leftarrow \mathsf{lsb}\,W_b^{v_b}$
  $j \leftarrow \mathsf{NextIndex}()$; $j' \leftarrow \mathsf{NextIndex}()$
  $T_G \leftarrow H(W_a^{v_a}, j) \oplus \mathcal{O}(W_a^{v_a}, j, v_b \oplus s_b)$
  $W_G^{v_a(v_b \oplus s_b)} \leftarrow H(W_a^{v_a}, j) \oplus s_a T_G$
  $T_E \leftarrow H(W_b^{v_b}, j') \oplus \mathcal{O}(W_b^{v_b}, j', v_a) \oplus W_a^{v_a}$
  $W_E^{v_a s_b} \leftarrow H(W_b^{v_b}, j') \oplus s_b(T_E \oplus W_a^{v_a})$
  $W^{v_a v_b} \leftarrow W_G^{v_a(v_b \oplus s_b)} \oplus W_E^{v_a s_b}$
  return $(W^{v_a v_b}, T_G, T_E)$

private procedure $\mathsf{evalWires}(f, \hat{x})$:
  for $i \in \mathsf{Inputs}(f)$ **do** $v_i \leftarrow x_i$
  for $i \notin \mathsf{Inputs}(f)$ **do**
    $\{a, b\} \leftarrow \mathsf{GateInputs}(f, i)$
    if $i \in \mathsf{XorGates}(f)$ **then**
      $v_i \leftarrow v_a \oplus v_b$
    else $v_i \leftarrow v_a \wedge v_b$
  return $\hat{v}$

private procedure $\mathsf{SimAnd}_3(W_a^0, W_b^0)$:
  $p_a \leftarrow \mathsf{lsb}\,W_a^0$; $p_b \leftarrow \mathsf{lsb}\,W_b^0$
  $j \leftarrow \mathsf{NextIndex}()$; $j' \leftarrow \mathsf{NextIndex}()$
  $T_G \leftarrow H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$
  $W_G^0 \leftarrow H(W_a^0, j) \oplus p_a T_G$
  $T_E \leftarrow H(W_b^0, j') \oplus H(W_b^1, j') \oplus W_a^0$
  $W_E^0 \leftarrow H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  $W^0 \leftarrow W_G^0 \oplus W_E^0$
  return (W^0, T_G, T_E)

Figure 3: The simulator for $\mathsf{prv.sim}_{\mathcal{S}}$ security, and the hybrids used in the proof.

$\{$*modify final loop of* Gb*:*$\}$
for $i \in \mathsf{Outputs}(f)$ **do**
  $j \leftarrow \mathsf{NextIndex}()$
  $d_i \leftarrow (H(W_i^0, j), H(W_i^1, j))$

procedure $\mathsf{De}(\hat{d}, \hat{Y})$:
  for $d_i \in \hat{d}$ **do**
    $j \leftarrow \mathsf{NextIndex}()$
    parse $(h_0, h_1) \leftarrow d_i$
    if $H(Y_i, j) = h_0$ **then** $y_i \leftarrow 0$
    else if $H(Y_i, j) = h_1$ **then** $y_i \leftarrow 1$
    else return $\perp$
  return $\hat{y}$

$\{$*modify final loop of* $\mathcal{S}$*:*$\}$
for $i \in \mathsf{Outputs}(f)$ **do**
  $j \leftarrow \mathsf{NextIndex}(); h \twoheadleftarrow \{0,1\}^k$
  if $y_i = 0$ **then** $d_i \leftarrow (H(W_i^0, j), h)$
  else $d_i \leftarrow (h, H(W_i^0, j))$

Figure 4: Changes to our scheme required to achieve authenticity.

2. $\mathcal{G}_1^{\mathsf{Rand}} \approx \mathcal{G}_1^{\mathsf{Circ}_R}$: We have just changed the oracle $\mathcal{O}$ from Rand to Circ_R. These two hybrids are indistinguishable simply by our assumption about the hash function. In Figure 3, $\mathcal{G}_1$ does *not* include the boxed statements.

3. $\mathcal{G}_1^{\mathsf{Circ}_R} \equiv \mathcal{G}_2^{\mathsf{Circ}_R}$: In Figure 3, we obtain $\mathcal{G}_2$ by adding the boxed statements to $\mathcal{G}_1$. We let the variable R in $\mathcal{G}_2$ refer to the R of the oracle Circ_R.

   The only difference between these two is that $\mathcal{G}_2$ computes some extra values that are never used (they will be used in $\mathcal{G}_3$). We couldn't compute these earlier since we couldn't use R while performing the previous step of the hybrid.

4. $\mathcal{G}_2^{\mathsf{Circ}_R} \equiv \mathcal{G}_3$: $\mathcal{G}_3$ induces identical distributions on every variable (i.e., $W_i^0, W_i^1, T_{Gi}, T_{Ei}$), but does so without explicitly having to compute v_i for non-input wires. For example, instead of randomly sampling $W_i^{v_i}$ and then setting $W_i^{\overline{v_i}} \leftarrow W_i^{v_i} \oplus R$, $\mathcal{G}_3$ randomly samples W_i^0 and then sets $W_i^1 \leftarrow W_i^0 \oplus R$. The algebraic relationships between each variable are still unchanged. We have also expanded the oracle calls in SimAnd_3 to correspond to $\mathcal{O} = \mathsf{Circ}_R$.

Finally, $\mathcal{G}_3$ computes $(\hat{F}, \hat{X}, \hat{d})$ as $(\hat{F}, \hat{e}, \hat{d}) \leftarrow \mathsf{Gb}(1^k, f)$; $\hat{X} \leftarrow \mathsf{En}(\hat{e}, x)$. This is precisely how these values are computed in the real interaction in the $\mathsf{prv.sim}_{\mathcal{S}}$ game. This completes our proof. □

4.3 Obtaining Authenticity

In the aut security game defined by Bellare et al. [5], an adversary is given $(\hat{F}, \hat{X})$. It is necessary to show that the adversary cannot produce $\tilde{Y} \neq \mathsf{Ev}(\hat{F}, \hat{X})$ such that $\mathsf{De}(\hat{d}, \tilde{Y}) \neq \perp$, except with negligible probability. This is clearly not the case for the scheme as we present it in Figure 2; in fact, De never returns $\perp$.

To achieve authenticity, we modify the scheme as described in Figure 4.

Theorem 5. *Our modified scheme (Figure 4) satisfies the security notion of* aut *with any H that has correlation robustness for naturally derived keys.*

Proof Sketch. Consider an interaction in which we run the $\mathsf{prv.sim}$-simulator $\mathcal{S}$ (with the change described in Figure 4) to generate $(\hat{F}, \hat{X}, \hat{d})$. We give $(\hat{F}, \hat{X})$ to the adversary and use $\hat{d}$ to run De and check whether the adversary succeeded in violating authenticity. In order to do so, the adversary would have to guess a value h that was chosen in the final loop of $\mathcal{S}$. But these values are independent of the adversary's view, so this can happen with probability at most $1/2^k$.

Now the rest of the proof follows an identical sequence of hybrids as the proof of Theorem 4. Eventually we reach an interaction that is identical to the aut game played against the adversary. By the indistinguishability of the hybrids, the adversary's success probability must be negligible. Note that the changes we have made to the scheme and simulator still allow the steps in the proof to retain *naturally derived* accesses to the oracles. □

5 Performance Comparison

We evaluate the performance of our scheme in comparison to previous garbling schemes using both analytical and experimental measurements.

Table 3 shows computations of the raw garbled circuit size in our scheme, calculated for several circuit designs. The table is derived from the one provided with fleXOR [19]; the circuits were obtained from [30, 11]. Our technique

outperforms all previous garbling schemes in this metric, achieving the expected maximum of 33% gain for most circuits. There are some AND-intensive circuits (e.g., the DES circuit used here) for which the previous fleXOR technique already does well, but we manage to improve a little upon that as well.

We selected a smaller, well-studied set of benchmark circuits for experimental evaluation. The aim here was to understand the cost tradeoffs for our scheme more clearly, in the context of a secure two-party computation protocol. In our scheme the evaluator performs one extra hash operation per gate while reducing network usage. Therefore, it is possible that we end up paying more in terms of computational resources, such as energy used.

Table 4 shows our measurements. Details of our experimental setup are provided below. We see that our scheme significantly reduces the total time and energy used by the evaluator in every test of the protocol. In our tests, we found that our scheme actually *increased* the power usage (i.e., higher wattage), but the increase was more than offset by the reduced runtime (i.e., lower total energy). It is conceivable that a very slow evaluator connected to a very fast LAN may not enjoy the same reduction in energy usage, but we did not have the equipment to run such a test and such a scenario seems unlikely to occur in practice. If the two parties have symmetric computational power, however, our protocol should always be better since the computational bottleneck would be the generator, who is performing four calls to H per AND gate in all schemes.

Experimental Setup. The experiments were performed using the Obliv-C system [32], where we hooked into the protocol execution to implement our own garbling scheme. This allowed us to easily reuse the exact same benchmark programs for both schemes. We executed Yao's standard semi-honest protocol for 2PC, with a security of 80-bit keys, and compared our scheme to Free-XOR with GRR3 AND gates. In both experimental setups, we used pipelining optimizations [14] and instantiated the H hash function in the garbling scheme using the fixed-key AES construction of [4] (described in Section 4). All measurements (time, network and energy) include the time for performing oblivious transfers and output sharing (which are not affected by the garbling scheme), hence the overall reductions support the argument that bulk of the bandwidth and computation is due to the garbled circuit execution.

The compilation was done using GCC 4.8.2, linked with libgcrypt 1.6.1 (older versions are much slower). The protocol was executed was between two LAN-connected desktops: one was an Intel Core i7-2600S at 2.8 GHz, running Ubuntu 14.04. The other was i7-2600 at 3.4 GHz running Ubuntu 13.10. Energy consumption was measured by using an electrical meter plugged in to the wall power outlet for one of the machines — the power meter had an USB interface that allowed us to measure power only for the duration of the job. For all jobs we report the average (time/energy) measurement over five runs, which was more than enough for obtaining statistically significant results (at $p < 0.05$).

6 Half-gates for Privacy-free Garbling

Jawurek, Kerschbaum, and Orlandi [18] described an elegant and practical zero-knowledge protocol based on garbled circuits. It allows a prover to prove statements of the form "$\exists x : C(x) = 1$", at a cost of just one garbled circuit for C.

In their protocol, the garbled circuit is evaluated by the prover, who knows the entire input to the garbled circuit, and the truth value along each wire. Hence, the *privacy* security property of garbled circuits (in the terminology of Bellare et al. [5]) is not required — only the *authenticity* property. Let us call a garbling scheme *privacy-free* if it satisfies only the authenticity property. Frederiksen, Nielsen & Orlandi [8] showed that privacy-free garbled circuits can be significantly smaller than their full-fledged counterparts.

Very roughly speaking, removing the privacy requirement saves one ciphertext per gate. Frederiksen et al. [8] adapt three garbling schemes to the privacy-free setting: GRR2, free-XOR, and fleXOR. Mirroring the situation with full-fledged garbled circuits, they showed how to garble an AND gate using just one ciphertext (i.e., *GRR1*), but in a way that is incompatible with free-XOR. When using free-XOR, it was necessary to garble AND gates using two ciphertexts.

Our approach using half-gates can also give a direct improvement in this privacy-free setting. Namely, one can garble a circuit with free-XOR gates, and garble AND gates using our evaluator-half-gate construction. In this setting, the evaluator knows *both* inputs to every AND gate, though our half-gate only takes advantage of the evaluator's knowledge of one input. Overall, we can perform privacy-free garbling at a cost of only one ciphertext per AND gate, and no cost for XOR gates. Interestingly, our construction of privacy-free garbling also results in less overall computation than the previous schemes — only two calls to H instead of three.

A summary of privacy-free garbling is given in Table 2, with concrete comparison of garbled circuit size on circuits of interest in Table 5. As before, our best improvements in this setting are on circuits for which free-XOR

circuit	GRR2 [28]	free-XOR [20]	fleXOR [19]	**this work**	↓%
DES	2.0	2.79	1.89	1.86	1%
AES	2.0	0.64	0.72	0.42	33%
SHA-1	2.0	1.82	1.39	1.21	12%
SHA-256	2.0	2.05	1.56	1.37	12%
Hamming distance	2.0	0.50	0.50	0.33	33%
minimum in set	2.0	0.87	0.87	0.58	33%
32×32 fast mult	2.0	0.90	0.94	0.60	33%
1024-bit millionaires	2.0	1.00	1.00	0.67	33%

Table 3: Comparison of garbled circuit size, for selected circuits of interest. Size measured in average number of ciphertexts per gate.

	Time (s)			Bandwidth (MB)			Energy (kJ)		
Benchmark	Whole	Half	↓%	Whole	Half	↓%	Whole	Half	↓%
Edit distance [14]	17.8	13.2	25.7%	200.4	133.6	33.3%	1.13	0.89	21.0%
AES [14]	18.2	17.0	7.0%	115.6	77.1	33.3%	1.25	1.18	5.3%
Set intersection [13]	37.0	29.7	19.7%	324.5	219.9	32.2%	2.41	2.03	15.5%

Table 4: Resource usage for three common programs. Edit distance refers to the Levenstein distance between two 200-byte strings. AES refers to 1 block of encryption and key expansion, iterated 10 times. Set intersection is performed on set of 1024, 32-bit integers, iterated 10 times. Each of these 3 jobs were in turn executed 5 times and measured separately, and the numbers are averages over these 5 runs. Whole denotes experimental setup using free-XOR with GRR2, while Half denotes a setup using our half-gates construction.

circuit	GRR1	free-XOR	fleXOR	**this work**	↓%
DES	1.0	1.86	0.96	0.93	3%
AES	1.0	0.43	0.51	0.21	50%
SHA-1	1.0	1.21	0.78	0.61	22%
SHA-256	1.0	1.37	0.87	0.68	22%

Table 5: Comparison of **privacy-free** garbled circuit size, for selected circuits of interest. Previous constructions and their statistics are from [8]. Size measured in average number of ciphertexts per gate.

was previously the best approach. Here, however, the relative improvement is more dramatic: we cut the size of the garbled circuit in half. Concretely, using the protocol of Jawurek, Kerschbaum, and Orlandi [18], it is possible to prove in zero knowledge a statement of the form "I know k such that $\text{AES}(k, m) = c$" (for public m, c) by sending only 108 kilobytes of garbled circuit (using 128-bit wire labels; for 80-bit wire labels, the garbled circuit is 68 kilobytes).

7 Lower Bounds on Garbled Circuits

In this section, we introduce a methodology for reasoning about lower bounds on the size of garbled gates. We then show that our construction from Section 3.2 is size-optimal for a large class of garbling schemes, which encompasses all known practical techniques.

7.1 Basic Methodology

There are many techniques that fall under the category of garbling schemes. We wish to focus on techniques based on (fast, practical) symmetric-key primitives only. Hence, in this section we model parties as computationally unbounded entities that can make polynomially many queries to a random oracle. This is the standard setting (initiated

by Impagliazzo and Rudich [17]) for proving lower bounds about Minicrypt.[3]

We wish to prove lower bounds relating to *concrete efficiency*; for example, prove that it is possible to garble an AND-gate with $2k$ bits of ciphertext but not with k bits. We say that a garbling scheme has **ideal security** if no adversary of the above form (computationally unbounded, with bounded queries to a random oracle) has advantage better than $\text{poly}(k)/2^k$ (rather than negligible) in the security games, where k is the security parameter and output length of the random oracle.

To see why it makes sense to restrict to ideal security in our setting, consider a garbling scheme where, with security parameter k, we apply our "two-ciphertext" construction for AND gates but with a $k/2$-bit random oracle. The resulting garbled gate is then only k bits, and indeed, no adversary has better than negligible advantage in the appropriate security games. However, it is possible to achieve advantage $\text{poly}(k)/2^{k/2}$.

Intuitively, a random oracle with security parameter (output length) k is an object that gives security $\text{poly}(k)/2^k$. We wish to consider only garbling schemes which do not "cheat" the size of the garbled gates by artificially degrading the security parameter of the random oracle relative to the security parameter of the garbling scheme.

Still, consider a garbling scheme that on security parameter k instantiates an ideally secure garbling scheme on security parameter $k - O(\log k)$. The result will yield security $\text{poly}(k)/2^{k-O(\log k)} = \text{poly}(k)/2^k$ — i.e., satisfy our ideal security definition as well. Hence, even with our model one cannot prove a clean lower bound of the form "$2k$ bits are required for an AND gate." Rather, one must prove something like "$2k - O(\log k)$ bits are required for an AND gate."[4] The special case we consider below, however, is already restricted to schemes whose gates are an integer multiple of k bits.

7.2 Linear Garbling Schemes

We first observe that, to the best of our knowledge, all techniques for practical garbling schemes (i.e., those discussed in the introduction) share certain features. Roughly speaking, the Gb and Ev procedures use only *linear* operations apart from queries to the random oracle (in this setting, we assume a random-oracle instantiation of the scheme), and choosing which linear operation to apply based on select bits of given wire labels (in the case of Ev) or on the association of select bits to TRUE/FALSE (in the case of Gb).

For example:

- In the classical garbling scheme, ciphertexts that comprise the garbled gate are all formed by taking an XOR of oracle responses with wire labels. Similarly, in most other schemes the garbled gate consists of values of the form $H(A\|B) \oplus C$, $H(A) \oplus C$, where A, B, and C are wire labels. The select bits and permute bits are used to decide which linear operations to apply (e.g., which ciphertext to decrypt in Ev).

- When using GRR3 row-reduction, one output wire label is chosen as $H(A\|B)$, hence linearly in the sense described above. Then behavior in Ev depends on the select bits of the given wire labels (i.e., whether to decrypt a ciphertext or simply take a hash of the input wire labels as the output), but in each case the resulting behavior is linear.

- In the GRR2 construction [28], generating and evaluating a gate involves interpolating polynomials that pass through points of the form $(t, H(A\|B))$. Since the values t are fixed, interpolation is a linear operation on outputs of H. Both the garbled gate itself and the output wire labels are the result of such interpolation. In Gb, the choice of which points to interpolate (hence, the choice of which linear operation to perform) depends on the assocation of select bits to TRUE/FALSE.

- In our scheme, Ev performs an additional XOR depending on the select bits of wire labels.

- When using free-XOR, wire labels are chosen subject to a linear relation $A_0 \oplus A_1 = B_0 \oplus B_1$.

We also observe the following properties common to existing garbling techniques:

[3] Minicrypt is one of Impagliazzo's hypothetical worlds [16], in which one-way functions exist but no stronger cryptography (in particular, public-key cryptography) exists. Since a random oracle models an ideal one-way function, we can model a world without cryptography beyond one-way functions as a world with computationally unbounded entities with access to a random oracle.

[4] Indeed, constructions that use the point-and-permute optimization degrade (by just one bit) the security of the underlying block cipher / hash function by using the least significant bit in a structured way.

- When garbling a circuit, the gates are processed in topological order. At the time a gate is processed, the labels of its input wires have already been determined, but the output wire labels may be determined as a result of garbling this gate.
- When restricted to operate on a **single gate**, the queries to the random oracle are made statically. That is, neither Ev nor Gb ever use the result of an oracle query to determine a future oracle query. For many schemes, this property is not true when garbling a larger circuit (an oracle query is used to determine an output wire label, which is then used to determine another oracle query in a downstream gate).

We argue that restrictions of this form capture **all existing practical approaches for garbled circuits**. Of course, we exclude techniques based on specific algebraic assumptions (e.g., [1, 2]) or more exotic tools like multilinear maps (e.g., [9]) which are arguably impractical and already ruled out by restricting our focus to Minicrypt.

The model. We formalize the observations above as follows. We restrict our focus to garbling schemes that garble a single AND gate. We say that a garbling scheme is **linear** if its procedures have the following form:

Gb: Parameterized by integers m, r, q and vectors $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\{\boldsymbol{C}_{a,b,0} \mid a,b \in \{0,1\}\}$, $\{\boldsymbol{C}_{a,b,1} \mid a,b \in \{0,1\}\}$, and $\{\boldsymbol{G}^{(i)}_{a,b} \mid a,b \in \{0,1\}, i \in [m]\}$. Each vector is of length $r+q$, with entries in $GF(2^k)$.

1. For $i \in [r]$, choose $R_i \leftarrow GF(2^k)$.
2. Make q distinct queries to the random oracle (which can be chosen as a deterministic function of the R_i values). Let $Q_1, \ldots, Q_q$ denote the responses to these queries. Define $\boldsymbol{S} = (R_1, \ldots, R_r, Q_1, \ldots, Q_q)$. These are the values on which the algorithm acts linearly.
3. Choose random permute bits $a, b \leftarrow \{0,1\}$ for the two input wires.
4. For $i \in \{0,1\}$, compute $A_i = \langle \boldsymbol{A}_i, \boldsymbol{S}\rangle$; $B_i = \langle \boldsymbol{B}_i, \boldsymbol{S}\rangle$; $C_i = \langle \boldsymbol{C}_{a,b,i}, \boldsymbol{S}\rangle$. Then $(A_0 \| 0, A_1 \| 1)$ and $(B_0 \| 0, B_1 \| 1)$ are taken as the input wire labels to the gate (i.e., the subscripts denote the public select bits), with A_a and B_b corresponding to FALSE. (C_0, C_1) are the output wire labels with C_1 corresponding to TRUE.
5. For $i \in [m]$, compute $G_i = \langle \boldsymbol{G}^{(i)}_{a,b}, \boldsymbol{S}\rangle$. The values $G_1, \ldots, G_m$ comprise the garbled circuit.

En: On input $x_a, x_b \in \{0,1\}$, set $\alpha = x_a \oplus a$ and $\beta = x_b \oplus b$, where a and b are the permute bits chosen above. Output $A_\alpha \| \alpha$ and $B_\beta \| \beta$.

Ev: Parameterized by integer q and vectors $\{\boldsymbol{V}_{\alpha,\beta} \mid \alpha, \beta \in \{0,1\}\}$, where each vector is of length $q+m+2$.

1. The input are wire labels $A_\alpha \| \alpha, B_\beta \| \beta$, tagged with their corresponding select bits, and the garbled circuit $G_1, \ldots, G_m$.
2. Make q distinct queries to the random oracle (which can be chosen as a deterministic function of the input wire labels). Let $Q'_1, \ldots, Q'_q$ denote the responses to these queries, and define $\boldsymbol{T} = (A_\alpha, B_\beta, Q'_1, \ldots, Q'_q, G_1, \ldots, G_m)$. These are the values on which Ev acts linearly.
3. Output the inner product $\langle \boldsymbol{V}_{\alpha,\beta}, \boldsymbol{T}\rangle$.

In Appendix A we show how well-known previous practical garbling schemes are linear in the above sense.

Limitations. We emphasize that our linear model of garbling schemes is most meaningful when garbling a single *atomic* gate. This is due to the issue regarding adaptive queries to the random oracle that happen when combining several garbled gates in a larger circuit.

For example, the best known way to garble an N-input AND gate is to garble it as a circuit of $N-1$, 2-input AND gates, for a total cost of $2N-2$ ciphertexts. But garbling in this way results in adaptive oracle queries, and the resulting scheme is not covered by our current model.

We suspect that it may be possible to augment our proof techniques for larger garbled circuits while accounting for adaptive oracle queries, but we leave this investigation to future work.

7.3 Lower Bound

Theorem 6. *Every ideally secure garbling scheme for AND gates that is linear in the above sense must have $m \geq 2$. That is, the garbled gate consists of at least $2k$ bits.*

Proof. From the correctness of the scheme, we must have $C_{(a\oplus\alpha)\wedge(b\oplus\beta)} = \langle \boldsymbol{V}_{\alpha,\beta}, \boldsymbol{T}\rangle$. Let us divide the vector $\boldsymbol{T}$ into a *public* and *private* part:

- The public part $\boldsymbol{T}^{pub}$ of $\boldsymbol{T}$ consists of the wire labels and oracle responses. Without loss of generality, the oracle queries made by Ev are a subset of the queries made by Gb. Any query made by Ev but not Gb will have an answer that is independent of all the activity of Gb. As such, correctness is violated if this oracle response is actually used in the evaluator's inner product.

  Hence the public portion of $\boldsymbol{T}$ is linear function of $\boldsymbol{S}$, and that linear function depends only on α, β, and not the secret permute bits a, b. We write $\boldsymbol{T}^{pub} = \mathbb{M}_{\alpha,\beta} \times \boldsymbol{S}^\top$.

- The private part $\boldsymbol{T}^{prv}$ of $\boldsymbol{T}$ consists of the garbled circuit components G_i. These are a linear function of $\boldsymbol{S}$ that can depend on the secret permute bits a, b. In particular, let $\mathbb{G}_{a,b}$ denote the matrix whose rows are $\boldsymbol{G}^{(1)}_{a,b}, \ldots, \boldsymbol{G}^{(m)}_{a,b}$. Then $\boldsymbol{T}^{prv} = \mathbb{G}_{a,b} \times \boldsymbol{S}^\top$. Our goal is to show that $\mathbb{G}_{a,b}$ must have at least 2 rows.

Let us also divide $\boldsymbol{V}_{\alpha,\beta}$ into a public and private portion, in an analogous way. We may thus rewrite the correctness condition as follows:

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

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

Now, the vector $\boldsymbol{S}$ is uniformly distributed. For this correctness probability to hold with probability 1 (or even noticeable probability) over the choice of $\boldsymbol{S}$, we must have the following equality of *vectors*:

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

Claim: Matrices $\{\mathbb{G}_{a,b} \mid a, b \in \{0,1\}\}$ are all distinct. Fix some permute bits a, b, then by the correctness condition, the values $\{\boldsymbol{Z}_{\alpha,\beta} + \boldsymbol{V}^{prv}_{\alpha,\beta} \times \mathbb{G}_{a,b}\}$ form a multi-set in which one element has multiplicity 3 and the other element has multiplicity 1. The element of multiplicity 1 is associated with a unique pair α, β. Changing the permute bits (and thus changing $\mathbb{G}_{a,b}$) must change which α, β is associated with the multiplicity-1 element. Hence the matrices $\mathbb{G}_{a,b}$ must be distinct.

Claim: Vectors $\{\boldsymbol{Z}_{\alpha,\beta} \mid \alpha, \beta \in \{0,1\}\}$ are pairwise linearly independent. To see why, suppose to the contrary that (by symmetry) $\boldsymbol{Z}_{0,1} = \sigma \boldsymbol{Z}_{0,0}$ for some scalar σ. Then consider an adversary given input wire labels corresponding to $\alpha = \beta = 0$. Instead of computing $\langle \boldsymbol{V}_{0,0}, \boldsymbol{T}\rangle$ as instructed, she can compute $\underline{\sigma} \cdot \langle \boldsymbol{V}^{pub}_{0,0}, \boldsymbol{T}^{pub}\rangle + \langle \underline{\boldsymbol{V}^{prv}_{0,1}}, \boldsymbol{T}^{prv}\rangle = \langle \boldsymbol{V}^{prv}_{0,1}, \boldsymbol{T}\rangle$. The result will reveal what the output of the garbled circuit would be if she had instead had input wires $\alpha = 0, \beta = 1$. For an AND gate, this is a violation of the privacy property (the output changes if and only if A_α encodes true).[5]

Claim: Vectors $\{\boldsymbol{V}^{prv}_{\alpha,\beta} \mid \alpha, \beta \in \{0,1\}\}$ are all distinct. To see why, consider the example of $\boldsymbol{V}^{prv}_{0,0}$ and $\boldsymbol{V}^{prv}_{0,1}$. With select bits either $(0,0)$ or $(0,1)$, and permute bits $(0,0)$, the garbled gate should evaluate to false. Hence:

$$\begin{aligned}
\boldsymbol{Z}_{0,0} + \boldsymbol{V}^{prv}_{0,0} \times \mathbb{G}_{0,0} &= \boldsymbol{C}_{0,0,0} \\
\boldsymbol{Z}_{0,1} + \boldsymbol{V}^{prv}_{0,1} \times \mathbb{G}_{0,0} &= \boldsymbol{C}_{0,0,0} \\
\implies (\boldsymbol{Z}_{0,0} - \boldsymbol{Z}_{0,1}) + (\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1})\mathbb{G}_{0,0} &= \boldsymbol{0}
\end{aligned}$$

[5]Note that this scenario does not violate security for an XOR gate. No matter what inputs the evaluator holds, she already knows that flipping one of the input bits will always flips the output.

Since $\boldsymbol{Z}_{0,0} - \boldsymbol{Z}_{0,1}$ is nonzero, we must have $\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1}$ nonzero as well. More generally, for any two elements of $\{\boldsymbol{V}^{prv}_{\alpha,\beta} \mid \alpha, \beta \in \{0,1\}\}$, one can choose permute bits a, b that cause those two input combinations to give the same output to the garbled gate.[6]

We now can prove the theorem. Consider two choices of select bits $(\alpha, \beta) \in \{(0,0),(0,1)\}$, and two choices of permute bits $(a, b) \in \{(0,0),(0,1)\}$. For all such combinations, the garbled gate must evaluate to false. Hence, we have:

$$\boldsymbol{C}_{0,0,0} = \boldsymbol{Z}_{0,0} + \boldsymbol{V}^{prv}_{0,0} \times \mathbb{G}_{0,0} \quad \text{(a)}$$
$$\boldsymbol{C}_{0,0,0} = \boldsymbol{Z}_{0,1} + \boldsymbol{V}^{prv}_{0,1} \times \mathbb{G}_{0,0} \quad \text{(b)}$$
$$\boldsymbol{C}_{0,1,0} = \boldsymbol{Z}_{0,0} + \boldsymbol{V}^{prv}_{0,0} \times \mathbb{G}_{0,1} \quad \text{(c)}$$
$$\boldsymbol{C}_{0,1,0} = \boldsymbol{Z}_{0,1} + \boldsymbol{V}^{prv}_{0,1} \times \mathbb{G}_{0,1} \quad \text{(d)}$$

If we combine these four equations as (a)-(b)-(c)+(d), we obtain:

$$\begin{aligned}\mathbf{0} &= \mathbf{0} + (\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1}) \times \mathbb{G}_{0,0} - (\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1}) \times \mathbb{G}_{0,1} \\ &= (\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1}) \times (\mathbb{G}_{0,0} - \mathbb{G}_{0,1})\end{aligned}$$

We see that $\boldsymbol{V}^{prv}_{0,0} - \boldsymbol{V}^{prv}_{0,1}$ is a *nonzero* vector in the left kernel of the *nonzero* matrix $\mathbb{G}_{0,0} - \mathbb{G}_{0,1}$. This implies that $\mathbb{G}_{0,0} - \mathbb{G}_{0,1}$ must have at least 2 rows. Hence, each $\mathbb{G}_{a,b}$ has at least 2 rows, and garbled gates consist of at least $2k$ bits, as desired. □

Discussion. Let us define the *parity* of a binary boolean gate as the number of 1s in its truth table. XOR, for instance, has even parity, while AND has odd parity. The proof of Theorem 6 applies to any odd-parity gate. We frequently used the facts that (a) the gate has one output with multiplicity 3 and another with multiplicity 1, and (b) depending on the permute bits, the output with multiplicity 1 could be associated with any of the 4 possible input combinations.

We are currently unable to prove a lower bound for completely arbitrary garbling schemes. As such, we cannot rule out the possibility of garbling an AND gate with only k bits. Yet, our lower bound shows that if such a method exists, then it must use (expensive) public-key primitives or be significantly non-linear in how it uses wire labels and outputs from the random oracle. Any non-linearity outside our model would represent an entirely new technical approach for garbled circuits.

What about the privacy-free setting? In arguing that the $\mathbb{G}_{a,b}$ matrices were distinct, we did not use the privacy property of the scheme. Privacy was only used to establish the other claims. Hence, for privacy-free garbled circuits we still have that the $\mathbb{G}_{a,b}$ matrices are distinct. As such, these cannot all be the empty matrix; they must contain at least one row. So for privacy-free garbling on an AND gate, we must have $m \geq 1$ (as in our construction); in other words, the garbled gate must contain at least k bits.

Acknowledgements

The authors would like to thank Jonathan Dorn for kindly lending and supporting their energy usage metering apparatus for our experiments.

References

[1] Applebaum, B.: Garbling XOR gates"for free" in the standard model. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 162–181. Springer (Mar 2013)

[2] Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 120–129. IEEE Computer Society Press (Oct 2011)

[6] This is another step of the proof that does not apply to XOR gates. Consider input wires with select bits $(0,0)$ or $(0,1)$. There is no choice of permute bits that could cause an XOR gate to give the same output for both.

[3] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press (May 1990)

[4] Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: 2013 IEEE Symposium on Security and Privacy. pp. 478–492. IEEE Computer Society Press (May 2013)

[5] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12. pp. 784–796. ACM Press (Oct 2012)

[6] Brandão, L.T.A.N.: Secure two-party computation with reusable bit-commitments, via a cut-and-choose with forge-and-lose technique - (extended abstract). In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 441–463. Springer (Dec 2013)

[7] Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.S.: On the security of the "free-XOR" technique. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 39–53. Springer (Mar 2012)

[8] Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. Cryptology ePrint Archive, Report 2014/598 (2014), http://eprint.iacr.org/

[9] Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013)

[10] Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10. pp. 451–462. ACM Press (Oct 2010)

[11] Henecka, W., Schneider, T.: Memory efficient secure function evaluation, https://code.google.com/p/me-sfe/

[12] Holzer, A., Franz, M., Katzenbeisser, S., Veith, H.: Secure two-party computations in ANSI C. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12. pp. 772–783. ACM Press (Oct 2012)

[13] Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: NDSS (2012)

[14] Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings. USENIX Association (2011)

[15] Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 18–35. Springer (Aug 2013)

[16] Impagliazzo, R.: A personal view of average-case complexity. In: Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995. pp. 134–147. IEEE Computer Society (1995), http://doi.ieeecomputersociety.org/10.1109/SCT.1995.514853

[17] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 8–26. Springer (Aug 1988)

[18] Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 955–966. ACM Press (Nov 2013)

[19] Kolesnikov, V., Mohassel, P., Rosulek, M.: Flexor: Flexible garbling for XOR gates that beats free-xor. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8617, pp. 440–457. Springer (2014)

[20] Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer (Jul 2008)

[21] Kreuter, B., Shelat, A., Shen, C.: Billion-gate secure computation with malicious adversaries. In: Kohno, T. (ed.) Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012. pp. 285–300. USENIX Association (2012), https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/kreu

[22] Lindell, Y.: Fast cut-and-choose based protocols for malicious and covert adversaries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 1–17. Springer (Aug 2013)

[23] Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology 22(2), 161–188 (Apr 2009)

[24] Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer (Mar 2011)

[25] Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., Prisco, R.D., Visconti, I. (eds.) SCN 08. LNCS, vol. 5229, pp. 2–20. Springer (Sep 2008)

[26] Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Blaze, M. (ed.) Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA. pp. 287–302. USENIX (2004), http://www.usenix.org/publications/library/proceedings/sec04/tech/malkhi.html

[27] Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce. pp. 129–139. ACM, New York, NY, USA (1999)

[28] Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer (Dec 2009)

[29] Shelat, A., Shen, C.H.: Two-output secure computation with malicious adversaries. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 386–405. Springer (May 2011)

[30] Tillich, S., Smart, N.: Circuits of basic functions suitable for MPC and FHE, http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/

[31] Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)

[32] Zahur, S.: Obliv-c: A lightweight compiler for data-oblivious computation. Workshop Applied Multi-Party Computation (2014), http://research.microsoft.com/apps/video/default.aspx?id=209896&l=i

A Linear Garbling Schemes

In this section we show that all existing garbling schemes are **linear** in the sense of Section 7.2. We show only the garbling procedure for AND gates, and use the notation of Section 7: (A_0, A_1) and (B_0, B_1) are the input wire labels, and (C_0, C_1) are the output wire labels. Bits a and b are secret so that A_a and B_b encode false. C_0 always encodes false.

Classical garbling: In a "classical" garbled circuit (with point-and-permute) optimization, the four ciphertexts comprising a garbled gate have the form $H(A\|B) \oplus C$, where the choice of C_0 or C_1 depends on the association between select bits and truth values. Below is an example of the linear operation of the scheme's operations. Highlighted entries are the positions that will vary based on a, b in Gb, or α, β in Ev.

$$\mathsf{Gb}: \quad \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C_0 \\ C_1 \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C_0 \\ C_1 \\ H(A_0\|B_0) \\ H(A_0\|B_1) \\ H(A_1\|B_0) \\ H(A_1\|B_1) \end{bmatrix} \qquad \text{for } a = b = 0$$

$$\mathsf{Ev}: \quad C = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} A_\alpha \\ B_\beta \\ H(A_\alpha\|B_\beta) \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} \qquad \text{for } \alpha = 0, \beta = 1$$

Row-reduction (GRR3). The row-reduction optimization of [27] sets one of the output wire labels to be $H(A\|B)$, so that one of the ciphertexts is no longer required (it becomes the all-zeroes string). Modifying the example from above, we have:

$$\mathsf{Gb}: \quad \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C_0 \\ C_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C \\ H(A_0\|B_0) \\ H(A_0\|B_1) \\ H(A_1\|B_0) \\ H(A_1\|B_1) \end{bmatrix} \qquad \text{for } a = b = 0$$

$$\mathsf{Ev}: \quad C = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} A_\alpha \\ B_\beta \\ H(A_\alpha\|B_\beta) \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} \qquad \text{for } \alpha = 0, \beta = 1$$

$$\mathsf{Ev}: \quad C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} A_\alpha \\ B_\beta \\ H(A_\alpha\|B_\beta) \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} \qquad \text{for } \alpha = \beta = 0$$

In this example, output wire label C_0 is chosen as $H(A_0\|B_0)$ because input combination A_0, B_0 should lead to the false wire label in this case ($a = b = 0$). The other output wire label C_1 is chosen randomly. In the case that $a = b = 1$, the two darkly shaded rows would be exchanged (and the three rows below would be changed accordingly).

In Ev, we compute the output wire label as $H(A_\alpha\|B_\beta)$ directly, when $\alpha = \beta = 0$. In other cases, we compute $H(A_\beta\|B_\beta)$ and use it to unmask one of the 3 ciphertexts.

Free-XOR + GRR3. In the free-XOR optimization [20], all wire label pairs are chosen as $(X, X \oplus R)$, where R is common to all wires. To achieve this, Gb is modified (from the previous example) as follows:

$$\mathsf{Gb}: \quad \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C_0 \\ C_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ R \\ H(A_0\|B_0) \\ H(A_0\|B_1) \\ H(A_1\|B_0) \\ H(A_1\|B_1) \end{bmatrix} \qquad \text{for } a = b = 0$$

Advanced row-reduction (GRR2). The garbled row-reduction optimization of [28] results in only 2 ciphertexts per AND gate. The idea is the following. For simplicity, assume $a = b = 0$, so that A_0, B_0 represent false. Then the evaluator should be able to obtain C_0 if he obtains any of $\{K_1 = H(A_0\|B_0), K_2 = H(A_0\|B_1), K_3 = H(A_1\|B_0)\}$, and obtain C_1 if he obtains $K_4 = H(A_1\|B_1)$.

We let P denote the unique degree-3 polynomial (over $GF(2^k)$) passing through points $\{(1, K_1), (2, K_2), (3, K_3)\}$. We then let Q denote the unique degree-3 polynomial passing through points $\{(4, K_4), (5, P(5)), (6, P(6))\}$. We give out values $P(5)$ and $P(6)$. Then if the evaluator who has input wire labels A_α, B_β interpolates a polynomial through $\{(2\alpha + \beta + 1, H(A_\alpha\|B_\beta)), (5, P(5)), (6, P(6))\}$, she will obtain either P or Q depending on the logic of the AND gate. Hence, we can set output wire labels $C_0 = P(0)$ and $C_1 = Q(0)$.

Let $V_{x,y,z}$ denote the 3×3 Vandermonde matrix that evaluates a polynomial-coefficient vector on points x, y, and z. Then $V^{-1}_{x,y,z}$ is the matrix that interpolates a polynomial's coefficients given its value at points x, y, and z. Hence, we have:

$$\mathsf{Gb}: \quad \begin{bmatrix} C_0 \\ P_5 \\ P_6 \end{bmatrix} = V_{0,5,6} \times V^{-1}_{1,2,3} \times \begin{bmatrix} H(A_0\|B_0) \\ H(A_0\|B_1) \\ H(A_1\|B_0) \end{bmatrix} \qquad \text{for } a = b = 0$$

$$\begin{bmatrix} C_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \times V^{-1}_{4,5,6} \times \begin{bmatrix} H(A_1\|B_1) \\ P_5 \\ P_6 \end{bmatrix}$$

$$\mathsf{Ev}: \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \times V^{-1}_{2\alpha+\beta+1,5,6} \times \begin{bmatrix} H(A_\alpha\|B_\beta) \\ P_5 \\ P_6 \end{bmatrix}$$

For different choices of a, b, different corresponding Vandermonde matrices are used in Gb.

For simplicity in Gb, we have written C_1 as a linear function of P_5, P_6. Clearly the linear operations compose, but we have not written out the tedious full expression for C_1 in terms of the $H(A_\alpha\|B_\beta)$ values.

Our scheme. In our scheme, the output wires of an AND gate will be $H(A_0) \oplus H(B_0)$ and $H(A_0) \oplus H(B_0) \oplus R$. The first (sender)half-gate is garbled as $H(A_0) \oplus H(A_1) \oplus bR$. The second (receiver)half-gate is garbled as $H(B_0) \oplus H(B_1) \oplus A_0 \oplus aR$.

$$\mathsf{Gb}: \quad \begin{bmatrix} A_0 \\ A_1 \\ B_0 \\ B_1 \\ C_0 \\ C_1 \\ G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & ab & 1 & 0 & 1 & 0 \\ 0 & 0 & 1-ab & 1 & 0 & 1 & 0 \\ 0 & 0 & b & 1 & 1 & 0 & 0 \\ 1 & 0 & a & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ R \\ H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix}$$

$$\mathsf{Ev}: \quad C = \begin{bmatrix} \beta & 0 & 1 & 1 & \alpha & \beta \end{bmatrix} \begin{bmatrix} A_\alpha \\ B_\beta \\ H(A_\alpha) \\ H(B_\beta) \\ G_1 \\ G_2 \end{bmatrix}$$

We can show the correctness of the scheme as follows. Recall that the result of evaluation should be $\gamma = (\alpha \oplus a) \wedge (\beta \oplus b)$. Since we are working in a field of characteristic 2, we have:

$$\begin{bmatrix} \beta & 0 & 1 & 1 & \alpha & \beta \end{bmatrix} \begin{bmatrix} A_\alpha \\ B_\beta \\ H(A_\alpha) \\ H(B_\beta) \\ G_1 \\ G_2 \end{bmatrix} = \begin{pmatrix} & \beta & [1 & 0 & \alpha & 0 & 0 & 0 & 0] \\ + & & [0 & 0 & 0 & 1-\alpha & \alpha & 0 & 0] \\ + & & [0 & 0 & 0 & 0 & 0 & 1-\beta & \beta] \\ + & \alpha & [0 & 0 & b & 1 & 1 & 0 & 0] \\ + & \beta & [1 & 0 & a & 0 & 0 & 1 & 1] \end{pmatrix} \begin{bmatrix} A_0 \\ B_0 \\ R \\ H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \alpha\beta + \alpha b + \beta a & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ R \\ H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \underbrace{(\alpha + a)(\beta + b)}_{\gamma} + ab & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ R \\ H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix} = C_\gamma$$