Better Concrete Security for Half-Gates Garbling (in the Multi-Instance Setting)

Chun Guo
Shandong University
chun.guo.sc@gmail.com

Jonathan Katz
George Mason University
jkatz2@gmail.com

Xiao Wang
Northwestern University
wangxiao@cs.northwestern.edu

Chenkai Weng
Northwestern University
ckweng@u.northwestern.edu

Yu Yu
Shanghai Jiao Tong University
yuyu@cs.sjtu.edu.cn

Abstract

We study the *concrete security* of high-performance implementations of half-gates garbling, which all rely on (hardware-accelerated) AES. We find that current instantiations using k-bit wire labels can be *completely broken*—in the sense that the circuit evaluator learns all the inputs of the circuit garbler—in time $O(2^k/C)$, where C is the total number of (non-free) gates that are garbled, possibly across multiple independent executions. The attack can be applied to existing circuit-garbling libraries using $k = 80$ when $C \approx 10^9$, and would require 267 machine-months and cost about $3500 to implement on the Google Cloud Platform. Since the attack can be entirely parallelized, the attack could be carried out in about a month using ≈ 250 machines.

With this as our motivation, we seek a way to instantiate the hash function in the half-gates scheme so as to achieve better concrete security. We present a construction based on AES that achieves optimal security in the single-instance setting (when only a single circuit is garbled). We also show how to modify the half-gates scheme so that its concrete security does not degrade in the multi-instance setting. Our modified scheme is as efficient as prior work in networks with up to 2 Gbps bandwidth.

1 Introduction

Roughly 35 years ago, Yao proposed the idea of *garbled circuits* for constant-round (semi-honest) secure two-party computation [44]. Over the past 15 years, spurred by initial implementations demonstrating its practicality [33, 37, 20], circuit garbling has received a considerable amount of attention and Yao's initial scheme has been significantly improved. Notable examples of such improvements include the point-and-permute technique [2], garbled row reduction [35], free-XOR [26], fleXOR [25], and half-gates garbling [46], as well as optimizations involving AES when modeled as a pseudorandom function [17] or when used with a fixed key and modeled as a random permutation [6]. Overall, these improvements have not only decreased the computational requirements of garbling, but have also—perhaps more importantly—reduced the communication complexity of garbled circuits. (Indeed, in current implementations of semi-honest secure two-party computation the overall running time is dominated by the communication time, and network bandwidth is the primary bottleneck.)

As these improvements to circuit garbling have been developed, however, there has been—perhaps somewhat surprisingly—a relative lack of attention on the *concrete security* [5] of these

proposals.[1] Understanding concrete security here is important for at least two reasons. First, as is well understood in the context of both public-key (e.g., [8, 12]) and symmetric-key (e.g., [5]) cryptography, when comparing the efficiency of different schemes it is important to take into account the concrete-security bound they each achieve; otherwise, the comparison may be inaccurate or misleading. Moreover, concrete security is critical for constructions based on symmetric-key primitives (such as AES), where there is no "security parameter" that can be arbitrarily adjusted as in the public-key world. (In particular, for AES the block size is fixed at 128 bits, and the maximum available key length is 256 bits.) There is thus the risk that a scheme that is only proven asymptotically secure—but has a poor concrete-security bound—may be insecure in practice for the available range of parameters.

When evaluating the concrete security of garbling, one can consider the "standard," single-instance setting where a single circuit is garbled, but it is also natural to consider a *multi-instance* setting where multiple circuits, for possibly different functions, are (independently) garbled—whether by the same user or distinct users—and, informally, the attacker succeeds if it is able to violate the security of any one of those garbled circuits. Concrete security in the multi-instance setting has received a lot of attention in the context of both public-key [4, 24] and symmetric-key [40, 3, 9, 19, 10] cryptography, but to the best of our knowledge it has not been previously considered in the setting of secure computation.[2]

1.1 Our Contributions

We study of the concrete security of garbling. We begin by examining the concrete security of existing, state-of-the-art implementations of the half-gates scheme (which is the most efficient garbling scheme currently known), and showing that it is worse than perhaps previously thought. We then propose a new way to instantiate the half-gates scheme that achieves better concrete-security bounds. Our results are described in further detail in what follows.

Concrete security of current half-gates garbling. The half-gates scheme—the most efficient garbling scheme currently known—is a technique for circuit garbling that is compatible with free-XOR (so uses no communication and negligible computation for XOR gates) and requires only $2k$ bits of communication for non-XOR gates, where k denotes the length of the wire labels. The half-gates scheme is based, abstractly, on a hash function H. Zahur et al. [46], motivated by JustGarble [6], propose to instantiate H using fixed-key AES in a particular way. Their suggestion was adopted by many existing implementations [43, 38, 45, 1, 42, 41, 15, 13] since it is much more efficient than instantiating H with a cryptographic hash function such as SHA-256 or SHA-3.

Let C be the number of (non-free) gates garbled. We show an attack on the half-gates scheme that results in a *complete break* (that is, the circuit evaluator learns all the inputs of the circuit garbler) in time $O(2^k/C)$. The attack works in the claimed time even when C denotes the *total* number of gates garbled across *multiple, independent* circuits. (In this case the attack completely violates privacy for at least *one* of the garbled circuits.)

We experimentally verify the feasibility of our attack against existing implementations of garbling that use "short" wire labels. In particular, we show that garbling 1 billion gates[3] (i.e.,

[1] This is even more surprising compared to the extensive research on the *statistical* security of circuit-garbling protocols in the *malicious* setting [29, 30, 39, 21, 28, 47].

[2] Work on *amortized* complexity of circuit garbling [22, 31] is close in spirit, but again focuses on statistical security of the overall protocol in the malicious setting rather than the concrete (computational) security of the garbling itself.

[3] Note that secure computation of 1 billion gates is now commonplace [27, 20, 32], and circuits with as many as

$C = 10^9$) with existing implementations of half-gates garbling that use wire labels of length $k = 80$ is vulnerable to an attack that can be carried out in 267 machine-months at a cost of $3500. Since the attack can be entirely parallelized, this means the attack can be carried out in about a month using ≈ 250 machines. Due to our attack, we urge users of the half-gates scheme to no longer use 80-bit wire labels (unless the scheme is modified as discussed below).

Better concrete security for half-gates garbling. Looking more closely at our attack, we observe that it does not arise due to any weakness in the half-gates scheme itself, but instead is possible *because of the way H is instantiated*. In particular, we show that the half-gates scheme has a *tight* security reduction (namely, requires time $O(2^k)$ to attack) if the hash function H being used is modeled as a random oracle. (See Appendix B.) As noted earlier, however, in existing implementations H is instantiated using (fixed-key) AES for better performance; this instantiation is *not* indifferentiable from a random oracle and our attack can be viewed as exploiting that gap.

The fact that the proposed instantiation of H is not indifferentiable from a random oracle was also observed by Guo et al. [18]. They define a property called tweakable circular correlation robustness (TCCR) for hash functions, show that using a TCCR hash function suffices for security of the half-gates scheme, and give a provably secure construction of a TCCR hash function based on fixed-key AES. They did not focus on obtaining better concrete security, and indeed, in Appendix C we show that using their hash function in the half-gates scheme would admit an attack similar to the one described above.

We thus turn to constructing a TCCR hash function with tight concrete security. In this context, the hash function H is evaluated on both a tweak and an input and, with our eventual application to garbling in mind, we use a fine-grained notion of concrete security that separately bounds the total number of calls the adversary makes to H as well as the maximum number of times μ the adversary repeats any particular tweak. As our main result, we show a construction of a TCCR hash function based on AES (modeled as an ideal cipher) that has tight concrete security when μ is small.

Importantly, $\mu = 1$ when a single circuit is garbled using the half-gates scheme, and so when our new hash function is used to instantiate the half-gates approach we immediately obtain a garbling scheme with tight security in the single-instance setting. In the multi-instance setting, however, μ can potentially be as large the number of circuits being garbled; thus, absent any changes, we would obtain a poor concrete-security bound, even when using our hash function, when many circuits are independently garbled. To address this, we show a simple way to randomize the tweaks used in the half-gates scheme in order to avoid significant degradation in the concrete security.

In contrast to the prior work of Guo et al. [18], the hash function we propose involves re-keying AES (and modeling AES as an ideal cipher) rather than relying on fixed-key AES (and modeling the result as a random permutation). Nevertheless, we show in Section 6 that by incorporating state-of-the-art optimizations for AES key scheduling [17], our hash function is almost as efficient as the one proposed by Guo et al. when used for circuit garbling.

1.2 Overview of the Paper

In Section 2 we establish notation and review relevant definitions for garbling schemes, including *concrete* security definitions for garbling in a *multi-instance* setting. We also describe the half-gates garbling scheme based on an abstract hash function H. In Section 3 we describe the instantiation

2^{37} gates have been garbled for some applications [14].

of H based on fixed-key AES that was proposed by Zahur et al. and that is used in existing implementations; we then show an attack with running time $O(2^k/C)$ that completely violates the privacy of that instantiation. We define the notion of multi-instance tweakable circular correlation robustness (miTCCR) for hash functions in Section 4.1, and show that the concrete security of the half-gates scheme when instantiated with a hash function H can be reduced to the concrete security of H in the sense of miTCCR. As our main positive result, we then show in Section 4.2 how to construct a hash function from an ideal cipher with tight security in that sense. In Section 5 we show how to slightly modify the half-gates scheme so as to also achieve good concrete security in the multi-instance setting. We discuss the performance of the resulting garbling scheme in Section 6.

2 Circuit Garbling

We adapt the definitions of garbling by Bellare et al. [7] to our setting. We consider boolean circuits containing AND and XOR gates with fan-in 2. (NOT gates can be handled by XORing with 1.) We represent any such circuit by a tuple $f = (n, m, \ell, \mathsf{Gates})$, where $n \geq 2$ denotes the number of input wires, $m \geq 1$ is the number of output wires, and ℓ is the number of gates. Such a circuit has exactly $n + \ell$ wires that we number starting from 1; we let $\mathsf{Inputs} = \{1, \ldots, n\}$ and $\mathsf{Outputs} = \{n + \ell - m + 1, \ldots, n + \ell\}$. The set $\mathsf{Gates} = \{(a, b, c, T)\}$, containing ℓ tuples, specifies the wiring of the circuit; a tuple $(a, b, c, T) \in \mathsf{Gates}$ with $a, b, c \in \{1, \ldots, n + \ell\}$ represents a gate of type $T \in \{\mathsf{XOR}, \mathsf{AND}\}$ with input wires a, b and output wire c. For a circuit f we let $|f| = C$ denote the number of AND gates in f. With slight abuse of notation, we let f also denote the function $f : \{0,1\}^n \to \{0,1\}^m$ computed by the circuit.

We consider a restricted class of garbling schemes in which garbling involves assigning two k-bit labels to each wire of the circuit, and evaluation involves computing one label for each output wire. (Our definition is thus similar to the one considered by Katz and Ostrovsky [23].) While this is less general than the class of garbling schemes considered by Bellare et al., this formulation suffices for analyzing the half-gates construction that is the focus of this paper.

Definition 1. *A circuit-garbling scheme* $\mathcal{G} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Decode})$ *is a tuple of algorithms where:*

- Garble *takes as input a circuit* f*, and returns* $(\mathsf{GC}, \{W_i^0, W_i^1\}_{i \in \mathsf{Inputs}}, d)$*, where* GC *denotes a garbled circuit,* $W_i^0, W_i^1 \in \{0,1\}^k$ *are the labels for the* i*-th input wire, and* d *represents decoding information.*
- Eval *takes as input a garbled circuit* GC *and input-wire labels* $\{W_i\}_{i \in \mathsf{Inputs}}$*. It returns output-wire labels* $\{W_i\}_{i \in \mathsf{Outputs}}$*.*
- Decode *takes as input output-wire labels* $\{W_i\}_{i \in \mathsf{Outputs}}$ *and decoding information* d*, and returns either* $\perp$ *or a string* $y \in \{0,1\}^m$*.*

Correctness requires that for any circuit f *and any* $x \in \{0,1\}^n$*, if we compute*

$$(\mathsf{GC}, \{W_i^0, W_i^1\}_{i \in \mathsf{Inputs}}, d) \leftarrow \mathsf{Garble}(f)$$

and $\{W_i\}_{i \in \mathsf{Outputs}} \leftarrow \mathsf{Eval}(\mathsf{GC}, \{W_i^{x_i}\})$*, then* $\mathsf{Decode}(\{W_i\}_{i \in \mathsf{Outputs}}, d) = f(x)$*.*

When we work in the ideal-cipher model (ICM), all algorithms (including any adversary) are given access to a random keyed permutation $E : \{0,1\}^L \times \{0,1\}^L \to \{0,1\}^L$ as well as its inverse E^{-1}; i.e., for every $\mathsf{key} \in \{0,1\}^L$ and input $x \in \{0,1\}^L$ it holds that $E^{-1}(\mathsf{key}, E(\mathsf{key}, x)) = x$.

We require correctness to hold for all such E. We sometimes also consider the random-permutation model (RPM) in which all parties have access to a random permutation π and its inverse. The RPM can be obtained from the ICM by setting $\pi(x) = E(0^L, x)$.

Security notions for garbling are considered in the following section.

2.1 (Multi-Instance) Security of Garbling

The canonical security definition for garbling schemes, which suffices for semi-honest secure computation, is *privacy*. As Bellare et al. [7] note, however, some more-recent applications of garbling require other definitions. For completeness, we thus also consider the notions of *obliviousness* and *authenticity* in Appendix A. In contrast to prior work, here we provide concrete-security definitions in a *multi-instance* setting in which an attacker may be given values produced by the (independent) garbling of $u \geq 1$ circuits.

Roughly speaking, privacy requires that the information needed to evaluate a garbled circuit (namely, GC, $\{W_i^{x_i}\}$, and d) reveals nothing other than $y = f(x)$. This is formalized by requiring the existence of a simulator Sim that takes the circuit f and a value y as input, and outputs values that are indistinguishable from GC, $\{W_i^{x_i}\}$, d. In the multi-instance setting, we compare the output of Sim to the outputs obtained from independently garbling u circuits. The following definition is specialized to the ICM for concreteness and since our main construction is in that model; it can be naturally adapted to the RPM.

Definition 2. *Garbling scheme $\mathcal{G}$ is (p, u, C, ε)-*$\mathsf{private}$ *if there is a simulator Sim so that for any distinguisher D making p queries to E and any $\{(f^i, x^i)\}_{i\in[u]}$ with $\sum_i |f^i| = C$, we have*

$$\left| \Pr_{\{(\mathsf{GC}^i, \{W_j^{i,0}, W_j^{i,1}\}, d^i) \leftarrow \mathsf{Garble}^E(f^i)\}_{i\in[u]}} \left[D^E\left(\{(\mathsf{GC}^i, \{W_j^{i,x_j^i}\}, d^i)\}_{i\in[u]} \right) = 1 \right] \right.$$
$$\left. - \Pr_{\{(\mathsf{GC}^i, \{W_j^i\}, d^i) \leftarrow \mathsf{Sim}^E(f^i, f^i(x^i))\}_{i\in[u]}} \left[D^E\left(\{(\mathsf{GC}^i, \{W_j^i\}, d^i)\}_{i\in[u]} \right) = 1 \right] \right| \leq \varepsilon,$$

where both probabilities are also over choice of E.

In the definition above, D may be unbounded so long as the number of queries it makes to E is bounded. The definition does not explicitly consider the running time of the simulator Sim, but one can verify that the running time of the simulator for our construction is $O(C)$. We remark further that while the distinguisher is given the circuits/inputs used in *all* the instances, we require the simulator to simulate each instance *independently*. (That is, when simulating the i-th instance the simulator is only given f^i, $f^i(x^i)$.)

2.2 The Half-Gates Garbling Scheme

The half-gates scheme $\mathsf{HalfGates}$ [46] is an approach for garbling that is compatible with the free-XOR technique, and only requires communicating $2k$ bits per AND gate. As the most efficient garbling scheme currently known, it is widely used in existing implementations of secure two-party computation in both the semi-honest and malicious settings. $\mathsf{HalfGates}$ is based on an abstract hash function $H : \{0,1\}^k \times \{0,1\}^L \to \{0,1\}^k$. We describe the scheme generically here, and discuss specific instantiations of H later.

$\mathsf{HalfGates}^H$

function $\mathsf{Garble}(f)$
  $R \leftarrow \{0,1\}^{k-1}\|1$
  $\mathsf{gid} := 1$
  for $i \in \mathsf{Inputs}$ **do**
    $W_i^0 \leftarrow \{0,1\}^k$
    $W_i^1 := W_i^0 \oplus R$
  for $(a,b,c,T) \in \mathsf{Gates}$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c^0 := W_a^0 \oplus W_b^0$
    else
      $(\mathsf{GC}[\mathsf{gid}], W_c^0) := \mathsf{GbAnd}(W_a^0, W_b^0, R, \mathsf{gid})$
      $\mathsf{gid} := \mathsf{gid} + 1$
  for $i \in \mathsf{Outputs}$ **do**
    $d_i := \mathsf{lsb}(W_i^0)$
  return $(\mathsf{GC}, \{(W_i^0, W_i^1)\}_{i\in\mathsf{Inputs}}, d)$

function $\mathsf{GbAnd}(W_a^0, W_b^0, R, \mathsf{gid})$
  $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
  $p_a := \mathsf{lsb}(W_a^0)$, $p_b := \mathsf{lsb}(W_b^0)$
  $T_G := H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$
  $T_E := H(W_b^0, j') \oplus H(W_b^1, j') \oplus W_a^0$
  $W_G^0 := H(W_a^0, j) \oplus p_a T_G$
  $W_E^0 := H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  $W_c^0 := W_G^0 \oplus W_E^0$
  return $((T_G, T_E), W_c^0)$

function $\mathsf{Eval}(\mathsf{GC}, \{W_i\}_{i\in\mathsf{Inputs}})$
  $\mathsf{gid} := 1$
  for $(a,b,c,T) \in \mathsf{Gates}$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c := W_a \oplus W_b$
    else
      $(T_G, T_E) := \mathsf{GC}[\mathsf{gid}]$
      $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
      $s_a := \mathsf{lsb}(W_a)$, $s_b := \mathsf{lsb}(W_b)$
      $W_G := H(W_a, j) \oplus s_a T_G$
      $W_E := H(W_b, j') \oplus s_b \cdot (T_E \oplus W_a)$
      $W_c := W_G \oplus W_E$
      $\mathsf{gid} := \mathsf{gid} + 1$
  return $\{W_i\}_{i\in\mathsf{Outputs}}$

function $\mathsf{Decode}(\{W_i\}_{i\in\mathsf{Outputs}}, d)$
  for $i \in \mathsf{Outputs}$ **do**
    $y_i := \mathsf{lsb}(W_i) \oplus d_i$
  return y

Figure 1: The half-gates scheme based on a hash function H.

We provide a high-level description, and refer to Figure 1 for details. To garble a circuit, the half-gates scheme begins by choosing a k-bit string R that is uniform subject to its least-significant bit being 1. For the i-th wire of the circuit with associated labels W_i^0, W_i^1, it will always be the case that $W_i^0 \oplus W_i^1 = R$. The garbler next chooses uniform 0-labels $\{W_i^0\}_{i\in\mathsf{Inputs}}$ for each input wire of the circuit. (This defines the 1-labels $\{W_i^1 = W_i^0 \oplus R\}$ for the input wires as well.) The garbled circuit is then generated gate-by-gate in topological order. For each XOR gate in the circuit, with ingoing wires a, b and outgoing wire c, the garbler simply sets $W_c^0 := W_a^0 \oplus W_b^0$ (and nothing is included in the garbled circuit for this gate). Each AND gate in the circuit is numbered topologically with a unique *gate identifier* gid ranging from 1 to C. For each AND gate in the circuit, with ingoing wires a, b and outgoing wire c, the garbler uses W_a^0, W_b^0, R, and the gate's identifier gid to compute the garbled table (T_G, T_E) as well as the 0-label W_c^0. This is done using a complicated procedure GbAnd that is defined in Figure 1. The garbled circuit consists of all the garbled AND gates.

To evaluate a garbled circuit, starting with labels $\{W_i\}_{i\in\mathsf{Inputs}}$ (where the evaluator does not necessarily know if $W_i = W_i^0$ or $W_i = W_i^1$), the evaluator proceeds as follows. For an XOR gate

with ingoing wires a, b and outgoing wire c, the evaluator computes $W_c := W_a \oplus W_b$. For an AND gate with ingoing wires a, b and outgoing wire c, the evaluator computes W_c from W_a, W_b, and the gate's identifier gid using the corresponding garbled table (see Figure 1). The final output is obtained using the least-significant bits of the output-wire labels.

3 Attacking Implementations of the Half-Gates Scheme

Inspired by earlier work of Bellare et al. [6], Zahur et al. [46] proposed to instantiate the hash function H in the half-gates schemes with a construction based on fixed-key AES (modeled as a random permutation π). Namely, they suggested to implement H as $H(x, i) = \pi(2x \oplus i) \oplus 2x \oplus i$.

Here, we show an attack that violates privacy when H is implemented in this way. Our attack succeeds with probability $O(p \cdot C/2^k)$, where p denotes the number of queries the attacker makes to π, and C denotes the number of AND gates garbled. Importantly, the attack also extends to the multi-instance setting, where C then denotes the total number of AND gates garbled. Our attack does not contradict the security proof by Zahur et al. (or the later proof of Guo et al. [18]), who only claim that an attacker's success probability cannot exceed this bound. We stress, however, that the existence of an attack meeting the bound was not previously known.

Guo et al. [18] have previously shown an attack on the above instantiation of H that violates (tweakable) correlation robustness with probability $O(pC/2^k)$ using p queries to π and C queries to a keyed version of H. However, their attack explicitly relies on the attacker's ability to make arbitrary queries to the keyed hash function and to obtain the full response, whereas in our setting the queries to the keyed hash function are made by the honest garbler (and so are outside the adversary's control) and the adversary is not given the output directly.

3.1 Attack Details

We describe the intuition behind the attack here, and give the details in Figure 2. The attack works by recovering the hidden global shift R used by the circuit garbler; note that once this value is obtained, the evaluator can use R along with the rest of the garbled circuit to learn, for each wire of the circuit, which labels are associated with which bits and thus, using the input labels it was sent, determine the actual input of the garbler. We focus on showing how to learn R. Observe that for each AND gate in the circuit with ingoing wires a, b and outgoing wire c, the circuit evaluator learns one of the two wire labels $W_a \in \{W_a^0, W_a^1\}$ as well as the value

$$T_G = H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$$

from the garbled gate. (Note that j depends on the gate identifier gid of the gate but we leave that implicit.) Recall further that $W_a^1 = W_a^0 \oplus R$. The circuit evaluator can thus compute

$$H_a \stackrel{\text{def}}{=} T_G \oplus W_a = H(W_a \oplus R, j) \oplus p_b R.$$

A key observation is that $p_b = 0$ with probability $1/2$! Thus, the circuit evaluator obtains, in expectation, $C/2$ values of the form $H_a^+ = H(W_a \oplus R, j)$ with W_a, j known. (We use H_a^+ to refer to an H_a-value for which $p_b = 0$.)

We now rely on the specific details of how H is implemented. When $H(x, i) \stackrel{\text{def}}{=} \pi(2x \oplus i) \oplus 2x \oplus i$ we have

$$H_a^+ = H(W_a \oplus R, j) = \pi(2(W_a \oplus R) \oplus j) \oplus 2(W_a \oplus R) \oplus j.$$

Inputs: A garbled circuit, along with input-wire labels $\{W_i\}_{i \in \mathsf{Inputs}}$.

Main algorithm:

1. Initialize $\mathcal{T} := \emptyset$. Evaluate the garbled circuit honestly and obtain a label W_a for each wire a.
2. For each AND gate (a, b, c, AND) with gate identifier gid and garbled table (T_G, T_E), set $j := 2 \times \mathsf{gid}$, compute $H_a := H(W_a, j) \oplus T_G$, and insert (H_a, j, W_a) into $\mathcal{T}$.
3. Choose uniform $W^* \in \{0,1\}^k$ until there exists (j, W_a) such that $(H(W^*, 0), j, W_a) \in \mathcal{T}$.
4. Given W^*, j, W_a from the previous step, compute $R := W_a \oplus W^* \oplus 2^{-1}j$. If $\mathsf{Check}(R) = 1$, output R; otherwise go to step 3.

$\mathsf{Check}(R)$**:**

1. For a gate (a, b, c, AND) with gate identifier gid and garbled table (T_G, T_E), let W_a, W_b, W_c be the labels computed on the respective wires.
2. If $((T_G, T_E), W_c \oplus wR) \stackrel{?}{=} \mathsf{GbAnd}(W_a^0 \oplus uR, W_b^0 \oplus vR, R, \mathsf{gid})$ for some values $u, v, w \in \{0,1\}$, output 1; else output 0.

Figure 2: Attack on the proposed implementation of the half-gates scheme.

If the circuit evaluator chooses a uniform W_i^*, it can check whether

$$H(W_i^*, 0) = H_a \tag{1}$$

for some a. If so, then (as we justify below in our discussion of false positives) with constant probability it will be the case that

$$2W_i^* = 2(W_a \oplus R) \oplus j. \tag{2}$$

Once the evaluator finds a W_i^* for which Eq. (2) holds, it can then easily solve for R. (Note also that it is easy to verify a candidate value R; see the Check routine in Figure 2.) The time to carry out the attack is therefore dominated by the time to find a solution to Eq. (2). Assume for simplicity we have exactly $C/2$ values $\{H_a^+\}$. Then if p uniform values $W_1^*, \ldots, W_p^*$ are chosen, the probability that Eq. (2) holds for some i, a is $p \cdot (C/2) \cdot 2^{-k} = p \cdot C/2^{k+1}$, as claimed.

Extension to the multi-instance setting. The above attack readily extends to the case when multiple circuits are (independently) garbled. In this case, C is simply the total number of AND gates garbled across all the circuits, and the attack recovers the shift R used for one of them.

False positives. We now more carefully account for the number of queries to π made during the course of the attack. We argued above that after p evaluations of H (which requires p evaluations of π) the attack finds R with probability $pC/2^{k+1}$. This analysis, however, does not account for false positives (i.e., a W^* for which Eq. (1) holds but Eq. (2) does not); note that every false positive incurs additional π-queries because it causes the Check routine to be executed. We now show that we expect only ≈ 2 false positives for every true positive.

To see this, fix some particular a and associated $H_a = H(W_a \oplus R, j) \oplus p_b R$, and consider a uniform W^*. There are three cases in which $H(W^*, 0) = H_a$:

- Case 1: $p_b = 0$ and $2W^* = 2(W_a \oplus R) \oplus j$. This occurs with probability $1/2^{k+1}$, and is a true positive.

- Case 2: $p_b = 0$ and $2W^* \neq 2(W_a \oplus R) \oplus j$, yet $H(W^*, 0) = H_a$. The probability of the first event is $1/2$, and the probability of the second is slightly less than 1. But conditioned on these events, the third event occurs only if

$$H(W^*, 0) = \pi(2W^*) = \pi(2(W_a \oplus R) \oplus j) \oplus 2(W_a \oplus R) \oplus j = H_a,$$

  which occurs with probability roughly $1/2^k$ since π is a random permutation. Overall, then, the probability of this case is also $1/2^{k+1}$.

- Case 3: $p_b = 1$, yet $H(W, 0) = H_a$. The probability of the first event is $1/2$. But conditioned on this, the second event occurs only if

$$H(W^*, 0) = \pi(2W^*) = \pi(2(W_a \oplus R) \oplus j) \oplus 2(W_a \oplus R) \oplus j \oplus R = H_a.$$

  There are now two sub-cases. If $2W^* = 2(W_a \oplus R) \oplus j$ (which occurs with probability 2^{-k}), then since R has min-entropy $k - 1$ the probability that the above equality holds is at most 2^{-k+1}. If $2W^* \neq 2(W_a \oplus R) \oplus j$, then because π is a uniform permutation the probability that the above equality holds is at most $1/(2^k - 1)$. Overall, then, the probability of this case is $1/2^{k+1} + 1/2^{2k} \approx 1/2^{k+1}$.

Summarizing: if the attack chooses p values $W_1^*, \ldots, W_p^*$, we expect a true positive with probability $pC/2^{k+1}$ and a false positive with probability $pC/2^k$. Put differently, if we set p such that $pC/2^{k+1} \approx 1$ then we expect to obtain R with probability ≈ 1 while incurring only ≈ 2 false positives. (Note that only $O(1)$ queries to π are made during calls to Check, so the net result is only a small number of additional queries to π.)

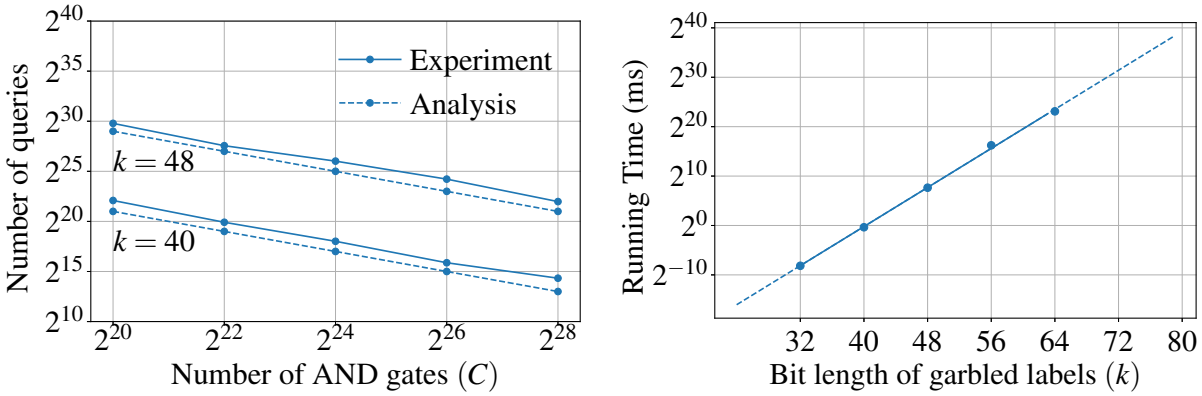
3.2 Attack Implementation

Here we describe our implementation of the attack described above.

Implementation optimizations. Above, we focused on the complexity of the attack in terms of the number of queries to π. In practice, though, the lookups in $\mathcal{T}$ also incur significant cost. For example, when $C = 2^{30}$ then $\mathcal{T}$ requires roughly 24 GB to store; this impacts both the running time of the attack (due to cache misses on memory accesses) and its dollar cost (since more-powerful machines are needed). To mitigate this, we made the following optimizations:

1. We first observe that it suffices to search for matches on H_a-values, and we thus store (only) those values in a hash table $\mathcal{H}$. Once a match on H_a is found, we can do a lookup in $\mathcal{T}$ to find the corresponding j, W_a values. Moreover, we store only 64 bits of each H_a value in $\mathcal{H}$ rather than the entire value. (This has only a small impact on the false-positive rate.) We store $\mathcal{H}$ in memory, but can store $\mathcal{T}$ on disk since it will be accessed only $O(1)$ times during the course of the attack.

2. We implement the hash table $\mathcal{H}$ using the "power of two choices" scheme [34]. In this construction, every element is mapped to two random buckets (each capable of holding eight 64-bit strings); an element is inserted in the bucket with lower occupancy, and lookups simply access both buckets. To further reduce the cost of memory accesses, we modified the way hashing is done to make sure that elements are always mapped to buckets within 16kB of each other in memory. In this way, both buckets for a given element will likely lie on the same page of memory, in which case both will be brought into the CPU cache when the memory access for the first bucket is made. This reduces the overall number of cache misses.



(a) Number of π-queries for the attack to succeed, on a log/log scale.

(b) The running time of our attack with $C = 2^{30}$ and different values of k.

Figure 3: Experimental complexity of the attack.

Verifying the attack complexity. We implemented our attack (with the above optimizations) to verify its correctness and complexity. We ran the attack with label lengths $k \in \{40, 48\}$ and number of gates C ranging from 2^{20}–2^{28} until the true value R was found; the attack was run 100 times for each set of parameters. We found that the average number of false positives (which cause lookups in $\mathcal{T}$ and invocations of the Check routine) was less than 5 in all cases. We plot the number of π-queries and the bound of $2^{k+1}/C$ given by our analysis in Figure 3a; our analysis is always within a factor of 2–3 of the experimental results. We believe our use of a hash table (which can cause additional false positives) partially contributes to the additional overhead.

Real-world running time and cost. We estimate the time and cost of implementing our attack when $k = 80$ and $C = 2^{30}$. For the purposes of this estimate, we assume customized preemptive instances with one Skylake CPU and 9 GB memory, each of which can be rented for $13.17 per month. By extrapolating experimental results for smaller values of k (see Figure 3b), we find that we can approximate the running time T of the attack (in milliseconds) as a function of k by the equation $T(k) = 2^{0.98877k - 39.77}$. For $k = 80$, this gives $T = 2^{39.3}$ ms or 267 machine-months. Such an attack would cost about $3500 to carry out. Since our attack can be fully parallelized, the wall-clock running time can be made arbitrarily small using multiple instances, without increasing the cost. For example, by opening 267 instances, the attack would finish in about a month.

4 Better Concrete Security for the Half-Gates Scheme

The attack in the previous section does not exploit any weakness in the half-gates scheme *per se*, but rather exploits a weakness in the way the underlying hash function is implemented. Building on the work of Guo et al. [18], we introduce here a security notion for hash functions called *multi-instance tweakable circular correlation robustness* (miTCCR) and show that this is an appropriate definition for analyzing the concrete security of the half-gates scheme.

4.1 Multi-Instance TCCR

Our definition of miTCCR differs from the related notion formalized by Guo et al. in two respects. First, we consider an attacker who is given access to *multiple* (independently keyed) functions,

rather than just one. Second, we explicitly allow the concrete security bound to depend on the maximum number of times μ an attacker repeats any particular tweak.

Given a function $H : \mathcal{W} \times \mathcal{T} \to \mathcal{W}$ (that depends on an ideal cipher E), define $\mathcal{O}_R^{\mathsf{miTCCR}}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$. Let $\mathsf{Func}_{\mathcal{W}\times\mathcal{T}\times\{0,1\},\mathcal{W}}$ denote the set of functions from $\mathcal{W} \times \mathcal{T} \times \{0,1\}$ to $\mathcal{W}$.

Definition 3. *Given a function $H^E : \mathcal{W} \times \mathcal{T} \to \mathcal{W}$, a distribution $\mathcal{R}$ on $\mathcal{W}$, and a distinguisher D, define*

$$\mathbf{Adv}^{\mathsf{miTCCR}}_{H,\mathcal{R}}(D, u, \mu) \stackrel{\text{def}}{=} \left| \Pr_{R_1,\ldots,R_u \leftarrow \mathcal{R}} \left[D^{E, \mathcal{O}^{\mathsf{miTCCR}}_{R_1}(\cdot),\ldots,\mathcal{O}^{\mathsf{miTCCR}}_{R_u}(\cdot)} = 1 \right] - \Pr_{f_1,\ldots,f_u \leftarrow \mathsf{Func}_{\mathcal{W}\times\mathcal{T}\times\{0,1\},\mathcal{W}}} \left[D^{E, f_1(\cdot),\ldots,f_u(\cdot)} = 1 \right] \right|,$$

where both probabilities are also over choice of E and we require that

1. *D never queries both $(w, i, 0)$ and $(w, i, 1)$ to the same oracle (for any w, i).*
2. *For all $i \in \mathcal{T}$, the number of queries (across all oracles) of the form $(\star, i, \star)$ is at most μ.*

We say H is $(p, q, u, \mu, \rho, \varepsilon)$-miTCCR, if for all distinguishers D making at most p queries to E and at most q queries (in total) to its other oracles, and all distributions $\mathcal{R}$ with min-entropy at least ρ, we have $\mathbf{Adv}^{\mathsf{miTCCR}}_{H,\mathcal{R}}(D, u, \mu) \leq \varepsilon$.

Note that we recover the definition from Guo et al. if we set $u = 1$ and $\mu = |\mathcal{T}|$.

The concrete security of the half-gates scheme is directly related to the concrete security (in the sense of miTCCR) of the underlying hash function used.

Theorem 1. *Let H be $(p, 2C, u, u, k-1, \varepsilon)$-miTCCR. Then the garbling scheme $\mathsf{HalfGates}^H$ is (p, u, C, ε)-private.*

A proof of the above follows along the same lines as the proof of the more general result we show later (cf. Theorem 3), so we omit it.

The challenge is thus to design a hash function with good concrete security in the sense of miTCCR. We remark that, as one might expect, a random oracle is one such candidate; see Appendix B. However, as discussed extensively by Guo et al. [18], it is not trivial to use a random oracle when implementing the half-gates scheme: there is a significant performance penalty when instantiating H using a cryptographic hash function like SHA-256 or SHA-3 (see also Table 1), and indifferentiable constructions of a random oracle from an ideal cipher E that are both efficient and have good concrete security are not known. (In particular, work of Gauravaram et al. [16] shows a construction using two calls to E with birthday-bound security; the construction we show in the next section is both more efficient and has better concrete security in the sense of miTCCR.)

4.2 Designing a Hash Function with Better Concrete Security

We construct (from an ideal cipher $E : \{0,1\}^L \times \{0,1\}^L \to \{0,1\}^L$) a hash function with good concrete security in the sense of miTCCR. Specifically, define $\widehat{\mathsf{MMO}}^E : \{0,1\}^L \times \{0,1\}^L \to \{0,1\}^L$ as

$$\widehat{\mathsf{MMO}}^E(x, i) \stackrel{\text{def}}{=} E(i, \sigma(x)) \oplus \sigma(x),$$

where σ is a linear orthomorphism. (We say $\sigma : \{0,1\}^L \to \{0,1\}^L$ is *linear* if $\sigma(x \oplus y) = \sigma(x) \oplus \sigma(y)$ for all $x, y \in \{0,1\}^L$. It is an *orthomorphism* if it is a permutation, and the function σ' given by $\sigma'(x) \stackrel{\text{def}}{=} \sigma(x) \oplus x$ is also a permutation.) We have:

Theorem 2. *If σ is a linear orthomorphism and E is modeled as an ideal cipher then $\widehat{\mathsf{MMO}}^E$ is $(p, q, u, \mu, \rho, \varepsilon)$-miTCCR, where*

$$\varepsilon = \frac{2\mu p}{2^\rho} + \frac{(\mu - 1) \cdot q}{2^\rho}.$$

Proof. Our proof uses the H-coefficient technique [36, 11], which we review in Appendix D (specialized for our proof). Fix a deterministic distinguisher D making queries to $u + 1$ oracles. The first is the ideal cipher (and its inverse); in the real world, the remaining oracles are of the form

$$\mathcal{O}_R^{\mathsf{miTCCR}}(w, i, b) = \widehat{\mathsf{MMO}}^E(R \oplus w, i) \oplus bR = E(i, \sigma(R \oplus w)) \oplus \sigma(R \oplus w) \oplus bR$$

(for u independent keys $R_1, \ldots, R_u$ sampled from $\mathcal{R}$), but in the ideal world they are u independent random functions from $\{0,1\}^{2L+1}$ to $\{0,1\}^L$. Following the notation from Appendix D, denote the transcript of D's interaction by $\mathcal{Q} = (\mathcal{Q}_E, \mathcal{Q}_\mathcal{O}, \mathbf{R})$. We only consider attainable transcripts. For $i \in \{0,1\}^L$ define $\mathcal{Q}_E[i] \stackrel{\text{def}}{=} \{(x, y) : (i, x, y) \in \mathcal{Q}_E\}$. Clearly, $\sum_{i \in \{0,1\}^L} \left|\mathcal{Q}_E[i]\right| = \left|\mathcal{Q}_E\right| = p$.

We say a transcript $(\mathcal{Q}_E, \mathcal{Q}_\mathcal{O}, \mathbf{R})$ is *bad* if:

- (B-1) There is a query $(\mathsf{idx}, w, i, b, z) \in \mathcal{Q}_\mathcal{O}$ and a query of the form $(i, \sigma(R_{\mathsf{idx}} \oplus w), \star)$ or of the form $(i, \star, \sigma(R_{\mathsf{idx}} \oplus w) \oplus b \cdot R_{\mathsf{idx}} \oplus z)$ in $\mathcal{Q}_E$.
- (B-2) There are distinct queries $(\mathsf{idx}, w, i, b, z)$, $(\mathsf{idx}', w', i, b', z') \in \mathcal{Q}_\mathcal{O}$ using the same "tweak" i such that $\sigma(R_{\mathsf{idx}} \oplus w) = \sigma(R_{\mathsf{idx}'} \oplus w')$ or $\sigma(R_{\mathsf{idx}} \oplus w) \oplus b \cdot R_{\mathsf{idx}} \oplus z = \sigma(R_{\mathsf{idx}'} \oplus w') \oplus b' \cdot R_{\mathsf{idx}'} \oplus z'$.

We bound the probabilities of the above events in the ideal world. Consider (B-1). Imagine that first all the oracles are chosen (which defines $\mathcal{Q}_E, \mathcal{Q}_\mathcal{O}$) and then the keys $\mathcal{R}$ are chosen. Fix some $(\mathsf{idx}, w, i, b, z) \in \mathcal{Q}_\mathcal{O}$. It is immediate that

$$\Pr[(i, R_{\mathsf{idx}} \oplus w, \star) \in \mathcal{Q}_E] \le \frac{\left|\mathcal{Q}_E[i]\right|}{2^\rho}$$

since the min-entropy of $\mathcal{R}$ is ρ. Moreover,

$$\Pr[(i, \star, \sigma(R_{\mathsf{idx}} \oplus w) \oplus bR_{\mathsf{idx}} \oplus z) \in \mathcal{Q}_E] = \Pr[(i, \star, \sigma(R_{\mathsf{idx}}) \oplus \sigma(w) \oplus bR_{\mathsf{idx}} \oplus z) \in \mathcal{Q}_E],$$

by linearity of σ. Now, note that:

- When $b = 0$, the above probability is at most $\left|\mathcal{Q}_E[i]\right| \cdot 2^{-\rho}$ since σ is a permutation and the min-entropy of $\mathcal{R}$ is ρ.
- When $b = 1$, the above probability is also at most $\left|\mathcal{Q}_E[i]\right| \cdot 2^{-\rho}$ since σ is an orthomorphism and the min-entropy of $\mathcal{R}$ is ρ.

Therefore,

$$\begin{aligned}\Pr[\text{(B-1)}] &\leq \sum_{(\mathsf{idx},w,i,b,z)\in\mathcal{Q}_{\mathcal{O}}} \frac{2\cdot\left|\mathcal{Q}_E[i]\right|}{2^\rho} \\ &= \sum_{i\in\{0,1\}^L} \underbrace{\sum_{(\mathsf{idx},w,i,b,z)\in\mathcal{Q}_{\mathcal{O}}}}_{\leq\mu} \frac{2\cdot\left|\mathcal{Q}_E[i]\right|}{2^\rho} \\ &\leq \mu\cdot\sum_{i\in\{0,1\}^L} \frac{2\cdot\left|\mathcal{Q}_E[i]\right|}{2^\rho} = \frac{2\mu p}{2^\rho}.\end{aligned}$$

We next consider (B-2). For fixed $i\in\{0,1\}^L$, consider a pair of distinct queries (idx,w,i,b,z), $(\mathsf{idx}',w',i,b',z')\in\mathcal{Q}_{\mathcal{O}}$. If $\mathsf{idx}\neq\mathsf{idx}'$, we have

$$\Pr\left[\sigma(R_{\mathsf{idx}}\oplus w)=\sigma(R_{\mathsf{idx}'}\oplus w')\right]\leq\frac{1}{2^\rho}$$

and

$$\Pr[\sigma(R_{\mathsf{idx}}\oplus w)\oplus bR_{\mathsf{idx}}\oplus z=\sigma(R_{\mathsf{idx}'}\oplus w')\oplus b'R_{\mathsf{idx}'}\oplus z']\leq\frac{1}{2^\rho}$$

as in the discussion of (B-1). If $\mathsf{idx}=\mathsf{idx}'$, then $\sigma(R_{\mathsf{idx}}\oplus w)=\sigma(R_{\mathsf{idx}'}\oplus w')\Rightarrow w=w'$ is not possible. Furthermore, with $b''=b\oplus b'$,

$$\begin{aligned}&\Pr[\sigma(R_{\mathsf{idx}}\oplus w)\oplus bR_{\mathsf{idx}}\oplus z=\sigma(R_{\mathsf{idx}'}\oplus w')\oplus b'R_{\mathsf{idx}'}\oplus z'] &(3)\\ &=\Pr[\sigma(w)\oplus z=\sigma(w')\oplus z'\oplus b''R_{\mathsf{idx}}]=\frac{1}{2^L}\leq\frac{1}{2^\rho}, &(4)\end{aligned}$$

using the fact that $z,z'\in\{0,1\}^L$ are uniform and independent. Thus, for any pair of queries in $\mathcal{Q}_{\mathcal{O}}$, the probability that (B-2) holds is at most $2/2^\rho$. If we let $C_i\leq\mu$ denote the number of queries in $\mathcal{Q}_{\mathcal{O}}$ using tweak i, then

$$\Pr[\text{(B-2)}]\leq\sum_{i\in\{0,1\}^L}\binom{C_i}{2}\cdot\frac{2}{2^\rho}\leq(\mu-1)\sum_{i\in\{0,1\}^L}\frac{C_i}{2^\rho}\leq\frac{(\mu-1)\cdot q}{2^\rho}.$$

Summarizing, the probability of a bad transcript in the ideal world is at most $\frac{2\mu p}{2^\rho}+\frac{(\mu-1)\cdot q}{2^\rho}$.

Fix a good transcript $\mathcal{Q}=(\mathcal{Q}_E,\mathcal{Q}_{\mathcal{O}},\mathbf{R})$. The probability that the ideal world is consistent with this transcript is given by Eq. (7). The probability that the real world is consistent with this transcript is

$$\frac{\Pr[\forall(\mathsf{idx},w,i,b,z)\in\mathcal{Q}_{\mathcal{O}}:\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}}}(w,i,b)=z\mid E\vdash\mathcal{Q}_E]}{\prod_{v\in\{0,1\}^L}(2^L)_{|\mathcal{Q}_E[v]|}}\cdot\Pr_{\mathcal{R}}[\mathbf{R}].$$

We can express the numerator of the above as

$$\prod_{j=1}^{q}\Pr[\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_j}}(w_j,i_j,b_j)=z_j\mid E\vdash\mathcal{Q}_E\wedge\forall\ell<j:\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_\ell}}(w_\ell,i_\ell,b_\ell)=z_\ell].$$

Note that $\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_j}}(w_j, i_j, b_j) = z_j$ iff $\widehat{\mathsf{MMO}}^E\big(R_{\mathsf{idx}_j} \oplus w_j, i_j\big) \oplus b_j R_{\mathsf{idx}_j} = z_j$, i.e.,

$$E\big(i_j, \sigma(R_{\mathsf{idx}_j} \oplus w_j)\big) = \sigma(R_{\mathsf{idx}_j} \oplus w_j) \oplus b_j R_{\mathsf{idx}_j} \oplus z_j.$$

Since the transcript is good, there is no query of the form $(i_j, \sigma(R_{\mathsf{idx}_j} \oplus w_j), \star)$ in $\mathcal{Q}_E$ (since (B-1) does not occur), nor is $E(i_j, \sigma(R_{\mathsf{idx}_j} \oplus w_j))$ determined by the fact that $\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_\ell}}(w_\ell, i_\ell, b_\ell) = z_\ell$ for all $\ell < j$ (since (B-2) does not occur). Similarly, there is no query of the form $(i_j, \star, \sigma(R_{\mathsf{idx}_j} \oplus w_j) \oplus b_j R_{\mathsf{idx}_j} \oplus z_j)$ in $\mathcal{Q}_E$ (since (B-1) does not occur), nor is $E^{-1}(i_j, \sigma(R_{\mathsf{idx}_j} \oplus w_j) \oplus b_j R_{\mathsf{idx}_j} \oplus z_j)$ determined by the fact that $\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_\ell}}(w_\ell, i_\ell, b_\ell) = z_\ell$ for all $\ell < j$ (since (B-2) does not occur). Thus, for all j we have

$$\Pr[\mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_j}}(w_j, i_j, b_j) = z_j \mid E \vdash \mathcal{Q}_E \wedge \forall \ell < j : \mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}_\ell}}(w_\ell, i_\ell, b_\ell) = z_\ell] \geq 1/2^L.$$

It follows that

$$\Pr[\forall (\mathsf{idx}, w, i, b, z) \in \mathcal{Q}_{\mathcal{O}} : \mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}}}(w, i, b) = z \mid E \vdash \mathcal{Q}_E] \geq 1/2^{Lq},$$

and so the probability that the real world is consistent with the transcript is at least the probability that the ideal world is consistent with the transcript. This completes the proof. □

Using shorter wire labels. Our construction above gives a hash function $H : \{0,1\}^L \times \{0,1\}^L \to \{0,1\}^L$, where L is the block length and key length of the underlying cipher E. In some applications of the half-gates scheme, one may prefer using wire labels of length $k < L$. This is easily done by defining $H' : \{0,1\}^k \times \{0,1\}^L \to \{0,1\}^k$ as

$$H'(x, i) = [H(x \| 0^{L-k}, i)]_k,$$

where $[z]_k$ denotes the k least-significant bits of z. It is not hard to see that if H is $(p, q, u, \mu, \rho, \varepsilon)$-miTCCR then so is H'. (Of course, for H' it must be the case that $\rho \leq k$.)

Putting everything together. Say $\widehat{\mathsf{MMO}}^E$ is used in the half-gates scheme with k-bit wire labels (as discussed above). Theorems 1 and 2 then imply that the resulting garbling scheme is (p, u, C, ε)-private with

$$\varepsilon = \frac{u \cdot p + (u-1) \cdot C}{2^{k-2}}.$$

Taking $u = 1$ (i.e., looking at the single-instance setting), we have $\varepsilon = p/2^{k-2}$, which is independent of the circuit size C and optimal up to a (small) constant. When $u > 1$, however, security degrades linearly in u; since u can be $\Theta(C)$, the security bound can be as bad as $O((pC + C^2)/2^k)$ in the multi-instance setting. We show in the next section how to rectify this.

5 Achieving Better Multi-Instance Security

As discussed at the end of the previous section, our new hash function gives an optimal concrete-security bound for the half-gates scheme in the single-instance setting. In the multi-instance setting, however, the security bound degrades as the number of instances increases.

Looking at our construction and the proof of miTCCR security (Theorem 2), we observe that the fundamental reason for the poor security bound in the multi-instance case is that μ (namely, the number of times a given "tweak" may be re-used; cf. Definition 3) can be as large as u (the number of circuits being garbled). Tracing back to the half-gates scheme, we see that this is because the scheme always assigns sequential gate identifiers (gids) starting at 1 to the AND gates in a circuit, and so in particular each circuit that is garbled will at least use the "tweak" $i = 1$. We fix this issue by modifying the scheme so that it instead numbers the gates sequentially *beginning at a random starting point* determined by the garbler (and sent to the evaluator along with the garbled circuit). That is, the only changes with respect to Figure 1 are that (1) in Garble, the initial value of gid is a uniform L-bit string, and (2) the initial value of gid is included in GC. We denote the modified scheme by $\widehat{\mathsf{HalfGates}}$. To analyze the resulting construction, we start with the following lemma.

Lemma 1. *Fix integers L, q, an integer $u \leq q$, and a sequence of positive integers $(q_1, \ldots, q_u)$ with $\sum_i q_i = q$. Consider the following experiment involving a set of 2^L bins and q balls: for each $i \in [u]$, q_i balls are placed in consecutive bins (wrapping around modulo 2^L), where the initial bin is uniform. If μ^* is the random variable denoting the maximum number of balls in any bin, then*

$$\Pr[\mu^* > \mu] \leq \frac{q^{\mu+1}}{(\mu+1)! \cdot 2^{\mu L}}\,.$$

Proof. Consider some μ sequences of balls, i.e., the i_1-th, …, i_μ-th, and consider the event that there is a $k \in \{0,1\}^L$ such that every one of those sequences hits the k-th bin. It can be seen that the probability is

$$2^L \times \frac{q_{i_1}}{2^L} \times \cdots \times \frac{q_{i_\mu}}{2^L} = \frac{q_{i_1} \times \cdots \times q_{i_\mu}}{2^{L\cdot(\mu-1)}}.$$

Since μ^* is the maximum number of balls in any of the 2^L bins, we have

$$\Pr[\mu^* \geq \mu] \quad \leq \sum_{0<i_1<i_2<\cdots<i_\mu\leq u} \frac{q_{i_1} \times \cdots \times q_{i_\mu}}{2^{L\cdot(\mu-1)}}$$

Observing that

$$\begin{aligned}(q_1 + q_2 + \cdots + q_u)^\mu \quad &\geq \sum_{i_1\neq i_2\neq\cdots\neq i_\mu} q_{i_1} \times \cdots \times q_{i_\mu} \\ &= \mu! \cdot \sum_{i_1<i_2<\cdots<i_\mu} q_{i_1} \times \cdots \times q_{i_\mu},\end{aligned}$$

we have

$$\sum_{i_1<i_2<\cdots<i_\mu} q_{i_1} \times \cdots \times q_{i_\mu} \leq \frac{(q_1 + q_2 + \cdots + q_u)^\mu}{\mu!}.$$

Therefore,

$$\Pr[\mu^* > \mu] = \Pr[\mu^* \geq \mu + 1] \quad \leq \frac{1}{2^{L\cdot\mu}} \times \frac{(q_1 + \cdots + q_u)^{\mu+1}}{(\mu+1)!} = \frac{q^{\mu+1}}{(\mu+1)! \cdot 2^{L\cdot\mu}}.$$

This complete the proof. □

With the above in place, we now prove:

Theorem 3. *Let H be $(p, 2C, u, \mu, k-1, \varepsilon)$-miTCCR. Then the garbling scheme $\widehat{\mathsf{HalfGates}}^{H}$ is (p, u, C, ε')-private, where*

$$\varepsilon' \leq \varepsilon + \frac{(2C)^{\mu+1}}{(\mu+1)! \cdot 2^{\mu L}}.$$

Proof. We describe a simulator Sim_1 that takes as input a circuit f and an output y, and generates a simulated garbled circuit, input-wire labels, and the decoding table. See below for details.

function $\mathsf{Sim}_1(f, y)$
  $\mathsf{gid}^* \leftarrow \{0,1\}^L$, $\mathsf{gid} := \mathsf{gid}^*$
  $\mathsf{Rand} \leftarrow \mathsf{Func}_{\{0,1\}^{k+L+1}, \{0,1\}^k}$
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^0 \leftarrow \{0,1\}^k$
  for $(a, b, c, T) \in \mathsf{Gates}(f)$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c^0 := W_a^0 \oplus W_b^0$
    else
      $(\mathsf{GC}[\mathsf{gid}], W_c^0) := \mathsf{SimAnd}_1(W_a^0, W_b^0, \mathsf{gid})$
      $\mathsf{gid} := \mathsf{gid} + 1$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i := \mathsf{lsb}(W_i^0) \oplus y_i$
  return $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^0\}_{i \in \mathsf{Inputs}}, d)$

function $\mathsf{SimAnd}_1(W_a^0, W_b^0, \mathsf{gid})$
  $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
  $p_a := \mathsf{lsb}(W_a^0)$, $p_b := \mathsf{lsb}(W_b^0)$
  $T_G := H(W_a^0, j) \oplus \mathsf{Rand}(W_a^0, j, p_b)$
  $T_E := H(W_b^0, j') \oplus \mathsf{Rand}(W_b^0, j', 0) \oplus W_a^0$
  $W_G^0 := H(W_a^0, j) \oplus p_a T_G$
  $W_E^0 := H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  $W_c^0 := W_G^0 \oplus W_E^0$
  return $((T_G, T_E), W_c^0)$

Fix some $\{(f^i, x^i)\}_{i \in [u]}$. We now show indistinguishability between the two distributions in Definition 2. To do so, we consider a sequence of hybrid distributions.

Ideal. Here, we run $\mathsf{Sim}_1(f^i, f^i(x^i))$ for $i \in [u]$.

Hybrid$_2$. Here, we run $\mathsf{Sim}_2(f^i, x^i)$ for $i \in [u]$, where Sim_2 is defined below. Intuitively, the description of Sim_1 is from the perspective of the garbler (who knows the $\{W_i^0\}$), while that of Sim_2 is from the perspective of the evaluator (who knows the $\{W_i^{v_i}\}$ only); the distribution of the outputs remains the same.

function $\mathsf{Sim}_2(f, x)$
  $\mathsf{gid}^* \leftarrow \{0,1\}^L$, $\mathsf{gid} := \mathsf{gid}^*$
  $\mathsf{Rand} \leftarrow \mathsf{Func}_{\{0,1\}^{k+L+1}, \{0,1\}^k}$
  Evaluate $f(x)$ and get all wire values v_i
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^{v_i} \leftarrow \{0,1\}^k$
  for $(a, b, c, T) \in \mathsf{Gates}(f)$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c^{v_c} := W_a^{v_a} \oplus W_b^{v_b}$
    else
      $(\mathsf{GC}[\mathsf{gid}], W_c^{v_c}) := \mathsf{SimAnd}_2(W_a^{v_a}, W_b^{v_b}, \mathsf{gid})$
      $\mathsf{gid} := \mathsf{gid} + 1$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i := \mathsf{lsb}(W_i^{v_i}) \oplus v_i$
  return $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^{v_i}\}_{i \in \mathsf{Inputs}}, d)$

function $\mathsf{SimAnd}_2(W_a^{v_a}, W_b^{v_b}, \mathsf{gid})$
  $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
  $s_a := \mathsf{lsb}(W_a^{v_a})$, $s_b := \mathsf{lsb}(W_b^{v_b})$
  $T_G := H(W_a^{v_a}, j) \oplus \mathsf{Rand}(W_a^{v_a}, j, v_b \oplus s_b)$
  $T_E := H(W_b^{v_b}, j') \oplus \mathsf{Rand}(W_b^{v_b}, j', v_a) \oplus W_a^{v_a}$
  $W_G^{v_a(v_b \oplus s_b)} := H(W_a^{v_a}, j) \oplus s_a T_G$
  $W_E^{v_a s_b} := H(W_b^{v_b}, j') \oplus s_b(T_E \oplus W_a^{v_a})$
  $W_c^{v_a v_b} := W_G^{v_a(v_b \oplus s_b)} \oplus W_E^{v_a s_b}$
  return $((T_G, T_E), W_c^{v_a v_b})$

We claim that distribution $\mathbf{Hybrid}_2$ is identical to distribution **Ideal**. This is because the values $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^0\}_{i\in\mathsf{Inputs}})$ in $\mathbf{Hybrid}_1$ and the corresponding values $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^{v_i}\}_{i\in\mathsf{Inputs}})$ in **Ideal** are all uniform, and in both distributions we have

$$d^i = \mathsf{lsb}(\mathsf{Eval}((\mathsf{gid}^*, \mathsf{GC}^i), \{W_j^i\}_{j\in\mathsf{Inputs}})) \oplus f^i(x^i),$$

where we slightly abuse notation and let $\mathsf{lsb}(W_1, \ldots, W_n) = \mathsf{lsb}(W_1), \ldots, \mathsf{lsb}(W_n)$.

$\mathbf{Hybrid}_3$. Here, we run $\mathsf{Sim}_3(f^i, x^i)$ for $i \in [u]$, where Sim_3 is defined below. Sim_3 is the same as Sim_2 except that it uses oracles $\mathcal{O}_R^{\mathsf{miTCCR}}$ in place of the random function Rand, and it computes values $\{W^{\bar{v}}\}$ that do not affect the output.

function $\mathsf{Sim}_3(f, x)$
  $\mathsf{gid}^* \leftarrow \{0,1\}^L$, $\mathsf{gid} := \mathsf{gid}^*$
  $R \leftarrow \{0,1\}^{k-1}\|1$
  Evaluate $f(x)$ and get all wire values v_i
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^{v_i} \leftarrow \{0,1\}^k$
    $W_i^{\bar{v}_i} := W_i^{v_i} \oplus R$
  for $(a, b, c, T) \in \mathsf{Gates}(f)$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c^{v_c} := W_a^{v_a} \oplus W_b^{v_b}$
    else
      $(\mathsf{GC}[\mathsf{gid}], W_c^{v_c}) := \mathsf{SimAnd}_3(W_a^{v_a}, W_b^{v_b}, \mathsf{gid})$
      $\mathsf{gid} := \mathsf{gid} + 1$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i := \mathsf{lsb}(W_i^{v_i}) \oplus v_i$
  return $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^{v_i}\}_{i\in\mathsf{Inputs}}, d)$

function $\mathsf{SimAnd}_3(W_a^{v_a}, W_b^{v_b}, \mathsf{gid})$
  $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
  $s_a := \mathsf{lsb}(W_a^{v_a})$, $s_b := \mathsf{lsb}(W_b^{v_b})$
  $T_G := H(W_a^{v_a}, j) \oplus \mathcal{O}_R^{\mathsf{miTCCR}}(W_a^{v_a}, j, v_b \oplus s_b)$
  $T_E := H(W_b^{v_b}, j') \oplus \mathcal{O}_R^{\mathsf{miTCCR}}(W_b^{v_b}, j', v_a) \oplus W_a^{v_a}$
  $W_G^{v_a(v_b\oplus s_b)} := H(W_a^{v_a}, j) \oplus s_a T_G$
  $W_E^{v_a s_b} := H(W_b^{v_b}, j') \oplus s_b(T_E \oplus W_a^{v_a})$
  $W_c^{v_a v_b} := W_G^{v_a(v_b\oplus s_b)} \oplus W_E^{v_a s_b}$
  return $((T_G, T_E), W_c^{v_a v_b})$

Let μ^* denote the maximum frequency of any tweak used as the input to $\mathcal{O}_R$, across all u executions of Sim_3. We claim that no distinguisher D making at most p queries to E can distinguish between $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ with probability better than $\varepsilon + \Pr[\mu^* > \mu]$. Indeed, we can easily reduce any such distinguisher to a distinguisher against H (in the sense of miTCCR) that respects the bound μ on the number of times a tweak may be repeated so long as $\mu^* \le \mu$. Note further that Lemma 1 implies $\Pr[\mu^* > \mu] \le \frac{(2C)^{\mu+1}}{(\mu+1)!\times 2^{\mu L}}$.

$\mathbf{Hybrid}_4$. Here, we run $\mathsf{Sim}_4(f^i, x^i)$ for $i \in [u]$, where Sim_4 is defined below. Sim_4 is identical to Sim_3 except that v_i is always set to 0 and $\mathcal{O}_R^{\mathsf{miTCCR}}(x, i, b)$ is expanded to $H(x \oplus R, i) \oplus bR$. It is immediate that distributions $\mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$ are identical.

function $\mathsf{Sim}_4(f, x)$
  $\mathsf{gid}^* \leftarrow \{0,1\}^L$, $\mathsf{gid} := \mathsf{gid}^*$
  $R \leftarrow \{0,1\}^{k-1} \| 1$
  for $i \in \mathsf{Inputs}(f)$ **do**
    $W_i^0 \leftarrow \{0,1\}^k$
    $W_i^1 := W_i^0 \oplus R$
  for $(a, b, c, T) \in \mathsf{Gates}(f)$ **do**
    if $T = \mathsf{XOR}$ **then**
      $W_c^0 := W_a^0 \oplus W_b^0$
    else
      $(\mathsf{GC}[\mathsf{gid}], W_c^0) := \mathsf{SimAnd}_3(W_a^0, W_b^0, R, \mathsf{gid})$
      $\mathsf{gid} := \mathsf{gid} + 1$
  for $i \in \mathsf{Outputs}(f)$ **do**
    $d_i := \mathsf{lsb}(W_i^0)$
  return $((\mathsf{gid}^*, \mathsf{GC}), \{W_i^{x_i}\}_{i \in \mathsf{Inputs}}, d)$

function $\mathsf{SimAnd}_4(W_a^0, W_b^0, R, \mathsf{gid})$
  $j := 2 \times \mathsf{gid}$, $j' := 2 \times \mathsf{gid} + 1$
  $p_a := \mathsf{lsb}(W_a^0)$, $p_b := \mathsf{lsb}(W_b^0)$
  $T_G := H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$
  $T_E := H(W_b^0, j') \oplus H(W_b^1, j') \oplus W_a^0$
  $W_G^0 := H(W_a^0, j) \oplus p_a T_G$
  $W_E^0 := H(W_b^0, j') \oplus p_b(T_E \oplus W_a^0)$
  $W_c^0 := W_G^0 \oplus W_E^0$
  return $((T_G, T_E), W_c^0)$

One may observe that $\mathbf{Hybrid}_4$ is identical to the real-world distribution that is obtained by running $\widehat{\mathsf{HalfGates}}^H(f^i)$ and then including the input-wire labels corresponding to x^i. This completes the proof. □

6 Concrete Security and Efficiency

Using Theorems 2 and 3 we see that when we instantiate $\widehat{\mathsf{HalfGates}}$ with $\widehat{\mathsf{MMO}}^E$, the overall garbling scheme is (p, u, C, ε)-private, with

$$\varepsilon = \frac{\mu p + (\mu - 1) \cdot C}{2^{k-2}} + \frac{(2C)^{\mu+1}}{(\mu+1)! \times 2^{\mu L}}. \tag{5}$$

Above, $k \leq L$ denotes the length of the wire labels and is chosen as part of the implementation, while μ is a free parameter that can be set to optimize the bound. The expression above can be separated into two terms: a term $\mu p / 2^{k-2}$ that represents the computational security (as it depends on the query complexity p of the attacker) and a term $\frac{(\mu-1)\cdot C}{2^{k-2}} + \frac{(2C)^{\mu+1}}{(\mu+1)! \times 2^{\mu L}}$ that corresponds to statistical security. To illustrate, we consider two particular options assuming $L = 128$ (to match the case where AES-128 is the cipher E):

1. $k = 80$, $C \leq 2^{43.5}$. The overall security bound here is optimized when $\mu = 1$, in which case

$$\varepsilon = \frac{p}{2^{78}} + \frac{2C^2}{2^{128}} \leq \frac{p}{2^{78}} + 2^{-40}.$$

   I.e., this gives 78-bit computational security and 40-bit statistical security.

2. $k = 128$, $C \leq 2^{61}$. Now the overall security bound is maximized when $\mu = 2$, in which case

$$\varepsilon \leq \frac{p}{2^{125}} + \frac{8 \cdot C^3}{3 \times 2^{256}} \leq \frac{p}{2^{125}} + 2^{-64}.$$

   I.e., this gives 125-bit computational security and 64-bit statistical security.

Hash function	NI support?	k	Comp. sec. (bits)	100 Mbps	2 Gbps	localhost
Zahur et al.	Y	128	89	0.4	7.8	23
SHA-3	N	128	125	0.27	0.27	0.28
SHA-256	N	128	125	0.4	1.1	1.2
SHA-256	Y	128	125	0.4	2.1	2.45
$\widehat{\mathsf{MMO}}^E$	Y	128	125	0.4	7.8	15
$\widehat{\mathsf{MMO}}^E$	Y	88	86	0.63	12	15

Table 1: **Performance of different hash functions in the half-gates scheme.** All reported numbers are in 10^6 AND gates per second. "NI support" indicates whether the implementation utilizes hardware-level instructions (i.e., AES-NI or SHA-NI), and "comp. sec." refers to the computational security bound assuming $C < 2^{40}$. The length of the wire labels is k.

Optimizations. Compared to the hash function proposed by Zahur et al. [46], which uses fixed-key AES, evaluation of our hash function involves re-keying AES each time it is called. In our implementation, we apply the optimizations introduced by Gueron et al. [17] that allow us to do key scheduling using AES-NI instructions with pipelining. In our current implementation, we batch two key-scheduling operations for each gate. In fact, since the AES key being used to garble a given gate (which depends on the gid) is entirely predictable, we could batch more than two key-scheduling operations to achieve even better efficiency. Our optimized implementation will be made publicly available in EMP [43].

Performance. In Table 1 we evaluate the performance of different hash functions in the half-gates scheme. "Zahur et al." refers to using $\mathsf{HalfGates}$ with their proposed hash function; the other rows refer to using $\widehat{\mathsf{HalfGates}}$ where we instantiate the hash function either with $\widehat{\mathsf{MMO}}^E$ (using AES-128 as the ideal cipher E), or with SHA-256 or SHA-3 (as random oracles).

We see that compared to the work of Zahur et al., when using wire labels of the same length $k = 128$ our scheme achieves better concrete security and is equally efficient as long as the network bandwidth is below 2 Gbps (so the network communication is the bottleneck). When the network is faster, the throughput (i.e., number of gates per second) of our scheme is lower but only by about 35%. Compared to instantiations using cryptographic hash functions, we see that garbling using SHA-256 without SHA-NI is up to 13× slower than our AES-based solution in a fast network; even with SHA-NI, garbling is up to 6× slower. Compared to the instantiation using SHA-3, our AES-based construction is up to 50× faster. We also provide the running time of our scheme using $k = 88$, which provides roughly the same security as the 128-bit scheme of Zahur et al.. We observe that in this case our scheme is about 1.5× faster in a 2 Gbps network, due to the shorter labels.

Acknowledgments

Work of Jonathan Katz was supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-1902070008. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government.

References

[1] Bar Ilan Cryptography Research Group. libscapi: The Secure Computation API. https://github.com/cryptobiu/libscapi, 2016.

[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 503–513. ACM Press, 1990.

[3] M. Bellare, D. J. Bernstein, and S. Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In *Advances in Cryptology—Eurocrypt 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, 2016.

[4] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology—Eurocrypt 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, 2000.

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 394–403. IEEE, 1997.

[6] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium on Security and Privacy (S&P) 2013*, pages 478–492, 2013.

[7] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *ACM Conf. on Computer and Communications Security (CCS) 2012*, pages 784–796. ACM Press, 2012.

[8] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology—Eurocrypt 1996*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996.

[9] M. Bellare and B. Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology—Crypto 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, 2016.

[10] P. Bose, V. T. Hoang, and S. Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In *Advances in Cryptology—Eurocrypt 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, 2018.

[11] S. Chen and J. P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology—Eurocrypt 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014.

[12] J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology—Eurocrypt 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, 2002.

[13] Data61. Multi-Protocol SPDZ. https://github.com/data61/MP-SPDZ, 2016.

[14] J. Doerner, D. Evans, and A. Shelat. Secure stable matching at scale. In *ACM Conf. on Computer and Communications Security (CCS) 2016*, pages 1602–1613. ACM Press, 2016.

[15] Engineering Cryptographic Protocols Group. ABY - A Framework for Efficient Mixed-protocol Secure Two-party Computation. https://github.com/encryptogroup/ABY, 2015.

[16] P. Gauravaram, N. Bagheri, and L. R. Knudsen. Building indifferentiable compression functions from the PGV compression functions. *Designs, Codes, and Cryptography*, 78(2):547–581, 2016.

[17] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas. Fast garbling of circuits under standard assumptions. *J. Cryptology*, 31(3):798–844, July 2018.

[18] C. Guo, J. Katz, X. Wang, and Y. Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *IEEE Symposium on Security and Privacy (S&P)*, 2020. Available at https://eprint.iacr.org/2019/074.

[19] V. T. Hoang and S. Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In *Advances in Cryptology—Crypto 2016, Part I*, volume 9814 of *LNCS*, pages 3–32. Springer, 2016.

[20] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium 2011*. USENIX Association, 2011.

[21] Y. Huang, J. Katz, and D. Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Advances in Cryptology—Crypto 2013, Part II*, volume 8043 of *LNCS*, pages 18–35. Springer, 2013.

[22] Y. Huang, J. Katz, V. Kolesnikov, R. Kumaresan, and A. J. Malozemoff. Amortizing garbled circuits. In *Advances in Cryptology—Crypto 2014, Part II*, volume 8617 of *LNCS*, pages 458–475. Springer, 2014.

[23] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology—Crypto 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, 2004.

[24] E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *Advances in Cryptology—Crypto 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, 2016.

[25] V. Kolesnikov, P. Mohassel, and M. Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In *Advances in Cryptology—Crypto 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, 2014.

[26] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *Intl. Colloquium on Automata, Languages, and Programming (ICALP)*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.

[27] B. Kreuter, A. Shelat, and C.-H. Shen. Billion-gate secure computation with malicious adversaries. In *USENIX Security Symposium 2012*, pages 285–300. USENIX Association, 2012.

[28] Y. Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *Advances in Cryptology—Crypto 2013, Part II*, volume 8043 of *LNCS*, pages 1–17. Springer, 2013.

[29] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology—Eurocrypt 2007*, volume 4515 of *LNCS*, pages 52–78. Springer, 2007.

[30] Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, Oct. 2012.

[31] Y. Lindell and B. Riva. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. In *Advances in Cryptology—Crypto 2014, Part II*, volume 8617 of *LNCS*, pages 476–494. Springer, 2014.

[32] Y. Lindell and B. Riva. Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In *ACM Conf. on Computer and Communications Security (CCS) 2015*, pages 579–590. ACM Press, 2015.

[33] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay—secure two-party computation system. In *USENIX Security Symposium 2004*, pages 287–302. USENIX Association, 2004.

[34] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[35] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conference on Electronic Commerce (EC)*, pages 129–139. ACM, 1999.

[36] J. Patarin. The "coefficients H" technique (invited talk). In *Annual International Workshop on Selected Areas in Cryptography (SAC) 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, 2009.

[37] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. In *Advances in Cryptology—Asiacrypt 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, 2009.

[38] P. Rindal. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.

[39] A. Shelat and C.-H. Shen. Two-output secure computation with malicious adversaries. In *Advances in Cryptology—Eurocrypt 2011*, volume 6632 of *LNCS*, pages 386–405. Springer, 2011.

[40] S. Tessaro. Optimally secure block ciphers from ideal primitives. In *Advances in Cryptology—Asiacrypt 2015, Part II*, volume 9453 of *LNCS*, pages 437–462. Springer, 2015.

[41] Unbound Tech. Protecting cryptographic signing keys and seed secrets with Multi-Party Computation. https://github.com/unbound-tech/blockchain-crypto-mpc, 2018.

[42] University of Bristol. APRICOT: Advanced protocols for real-world implementation of computational oblivious transfers. https://github.com/bristolcrypto/apricot, 2016.

[43] X. Wang, A. J. Malozemoff, and J. Katz. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit, 2016.

[44] A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.

[45] S. Zahur and D. Evans. Obliv-C: A language for extensible data-oblivious computation. Cryptology ePrint Archive, Report 2015/1153, 2015. http://eprint.iacr.org/2015/1153.

[46] S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole—reducing data transfer in garbled circuits using half gates. In *Advances in Cryptology—Eurocrypt 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, 2015.

[47] R. Zhu, Y. Huang, J. Katz, and A. Shelat. The cut-and-choose game and its application to cryptographic protocols. In *USENIX Security Symposium 2016*, pages 1085–1100. USENIX Association, 2016.

A (Multi-Instance) Obliviousness and Authenticity

We recall definitions of obliviousness and authenticity for garbling schemes, adapted for the multi-instance setting. We then argue why our construction (with small modifications) achieves optimal security bounds for these notions as well.

A.1 Obliviousness

Obliviousness, informally, requires that $(\mathsf{GC}, \{W_i^{x_i}\})$ reveal nothing about x, including $f(x)$. This is formalized by requiring the existence of a simulator Sim that takes the circuit f and outputs values that are indistinguishable from $(\mathsf{GC}, \{W_i^{x_i}\})$. In the multi-instance setting, we compare the output of Sim to the outputs obtained when independently garbling u circuits.

Definition 4. *Garbling scheme $\mathcal{G}$ is (p, u, C, ε)-*oblivious *if there is a simulator Sim such that for any distinguisher D making p queries to E and any $\{(f^i, x^i)\}_{i\in[u]}$ with $\sum_i |f^i| = C$, we have*

$$\left| \Pr_{\{(\mathsf{GC}^i, \{W_j^{i,0}, W_j^{i,1}\}, d^i) \leftarrow \mathsf{Garble}^E(f^i)\}_{i\in[u]}} \left[D^E\left(\{(\mathsf{GC}^i, \{W_j^{i,x_j^i}\})\}_{i\in[u]} \right) = 1 \right] \right.$$
$$\left. - \Pr_{\{(\mathsf{GC}^i, \{W_j^i\}) \leftarrow \mathsf{Sim}^E(f^i)\}_{i\in[u]}} \left[D^E\left(\{(\mathsf{GC}^i, \{W_j^i\})\}_{i\in[u]} \right) = 1 \right] \right| \leq \varepsilon,$$

where both probabilities are also over choice of E.

Theorem 4. *Let H be $(p, 2C, u, \mu, k-1, \varepsilon)$-miTCCR. Then the garbling scheme $\widehat{\mathsf{HalfGates}}^H$ is (p, u, C, ε')-oblivious, where*

$$\varepsilon' \leq \varepsilon + \frac{(2C)^{\mu+1}}{(\mu+1)! \times 2^{\mu L}}.$$

We briefly justify the above bound. Observe that the simulator (for privacy) in the proof of Theorem 3 only uses the output y when computing the decoding information d. The simulator for obliviousness is identical except that it omits that step. The concrete security bound we obtain for obliviousness is thus (at most) the bound we prove for privacy.

A.2 Authenticity

Informally, authenticity requires that an attacker given $(\mathsf{GC}, \{W_i^{x_i}\})$ should be unable to generate output-wire labels that cause the decoding algorithm to produce a valid output other than $f(x)$. In the multi-instance generalization, the adversary succeeds if it can do this for any of u instances.

Definition 5. *Garbling scheme $\mathcal{G}$ is $(p, u, C, m, \varepsilon)$-*authentic *if for any $\mathcal{A}$ making p queries to E and any $\{(f^i, x^i)\}_{i\in[u]}$ with $\sum_i |f^i| = C$ and $\sum_i m_i = m$, we have*

$$\Pr_{\substack{\{(\mathsf{GC}^i, \{W_j^{i,0}, W_j^{i,1}\}, d^i)\\ \leftarrow \mathsf{Garble}^E(f^i)\}_{i\in[u]}}} \left[\begin{array}{c} \mathcal{A}^E\left(\{(\mathsf{GC}^i, \{W_j^{i,x_j^i}\})\}_{i\in[u]}\right) = (i', \{W_j'\}), \\ \mathsf{Eval}(\mathsf{GC}^{i'}, \{W_j^{i,x_j^i}\}) \neq \{W_j'\} \wedge \mathsf{Decode}(d^{i'}, \{W_j'\}) \neq \bot \end{array} \right] \leq \varepsilon,$$

where the probability is also over choice of E.

In order to obtain authenticity, we need to modify $\widehat{\mathsf{HalfGates}}$ slightly. Specifically, the decoding information d_i for the i-th output wire will now be

$$(d_i[0], d_i[1]) = (H(W_i^0, \mathsf{gid}), H(W_i^1, \mathsf{gid})).$$

Decoding of a label W_i on the i-th output wire involves checking whether $H(W_i, \mathsf{gid})$ is equal to $d_i[0]$ or $d_i[1]$ (and returning $\bot$ if neither holds). We refer to the resulting scheme as $\overline{\mathsf{HalfGates}}$.

Theorem 5. *Let H be $(p, 2C + m, u, \mu, k - 1, \varepsilon)$-miTCCR. Then the garbling scheme $\overline{\mathsf{HalfGates}}^H$ is $(p, u, C, m, \varepsilon')$-authentic, where*

$$\varepsilon' \leq \varepsilon + \frac{(2C + m)^{\mu+1}}{(\mu + 1)! \times 2^{\mu L}} + 2^{-k}.$$

Our proof of the above proceeds in two steps: (1) we construct a simulator for the garbling scheme and show that the simulated garbled circuits are indistinguishable from real garbled circuits; (2) we show that the adversary cannot break authenticity for simulated garbled circuits. For(1), the simulator is almost identical to the privacy simulator we show in the proof of Theorem 3, except that it chooses uniform $d_i[1] \in \{0,1\}^k$ and sets $d_i = (H(W_i^0, \mathsf{gid}), d_i[1])$ if $y_i = 0$, and chooses uniform $d_i[0] \in \{0,1\}^k$ and sets $d_i = (d_i[0], H(W_i^0, \mathsf{gid}))$ if $y_i = 1$. By an argument similar to that used in the proof of Theorem 3, any distinguisher making at most p queries to E can distinguish between simulated garbled circuits and real garbled circuits with probability at most $\varepsilon + \frac{(2C+m)^{\mu+1}}{(\mu+1)!\times 2^{\mu L}}$. (The only difference is that we need to also count the oracle queries needed to decode.) For (2), since output labels are uniform and independent, the probability an attacker can violate authenticity is 2^{-k}.

B A Random Oracle as an miTCCR Hash Function

We show that a random oracle $\mathsf{RO} : \{0,1\}^{2L} \to \{0,1\}^L$ (also) has good concrete security in the sense of miTCCR. For completeness, we give the relevant definition (obtained by suitable modifying Definition 3). Recall that $\mathcal{O}_R^{\mathsf{miTCCR}}(x, i, b) = \mathsf{RO}(x \oplus R, i) \oplus b \cdot R$.

Definition 6. *Given a distribution $\mathcal{R}$ on $\{0,1\}^L$ and a distinguisher D, define*

$$\mathbf{Adv}^{\mathsf{miTCCR}}_{\mathsf{RO},\mathcal{R}}(D,u,\mu) \stackrel{\text{def}}{=} \Bigg| \Pr_{R_1,\ldots,R_u \leftarrow \mathcal{R}} \left[D^{\mathsf{RO},\mathcal{O}^{\mathsf{miTCCR}}_{R_1}(\cdot),\ldots,\mathcal{O}^{\mathsf{miTCCR}}_{R_u}(\cdot)} = 1\right] - \Pr_{f_1,\ldots,f_u \leftarrow \mathsf{Func}_{\mathcal{W}\times\mathcal{T}\times\{0,1\},\mathcal{W}}} \left[D^{\mathsf{RO},f_1(\cdot),\ldots,f_u(\cdot)} = 1\right] \Bigg|,$$

where both probabilities are also over choice of RO *and we require that*

1. *D never queries both $(x,i,0)$ and $(x,i,1)$ to the same oracle (for any x,i).*
2. *For all $i \in \{0,1\}^L$, the number of queries (across all oracles) of the form $(\star,i,\star)$ is at most μ.*

We say RO *is $(p,q,u,\mu,\rho,\varepsilon)$-miTCCR, if for all distinguishers D making at most p queries to* RO *and at most q queries (in total) to its other oracles, and all distributions $\mathcal{R}$ with min-entropy at least ρ, we have $\mathbf{Adv}^{\mathsf{miTCCR}}_{\mathsf{RO},\mathcal{R}}(D,u,\mu) \le \varepsilon$.*

Theorem 6. RO *is $(p,q,u,\mu,\rho,\varepsilon)$-miTCCR, where*

$$\varepsilon = \frac{\mu p}{2^\rho} + \frac{(\mu-1)q}{2^{\rho+1}}.$$

Proof. Fix a deterministic distinguisher D making queries to $u+1$ oracles. The first is $\mathsf{RO} : \{0,1\}^{2L} \to \{0,1\}^L$; in the real world, the remaining u oracles are $\mathcal{O}^{\mathsf{miTCCR}}_{R_1}(\cdot),\ldots,\mathcal{O}^{\mathsf{miTCCR}}_{R_u}(\cdot)$, where the R_i are chosen independently from distribution $\mathcal{R}$, while in the ideal world they are u independent random functions with the correct domain and range. Denote the transcript of D's interaction by $\mathcal{Q} = (\mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathcal{O}}, \mathbf{R})$, where $\mathcal{Q}_{\mathsf{RO}} = \{(x_1,i_1,y_1),\ldots\}$ means that D queried $\mathsf{RO}(x,i)$ and received response y.

An attainable transcript $(\mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathcal{O}}, \mathbf{R})$ is *bad* if:

- (B-1) There is a query $(\mathsf{idx},w,i,z) \in \mathcal{Q}_{\mathcal{O}}$ and a query of the form $(R_{\mathsf{idx}} \oplus w, i, \star)$ in $\mathcal{Q}_{\mathsf{RO}}$.
- (B-2) There exist two distinct queries (idx,w,i,z), $(\mathsf{idx}',w',i,z') \in \mathcal{Q}_{\mathcal{O}}$ such that $R_{\mathsf{idx}} \oplus w = R_{\mathsf{idx}'} \oplus w'$.

It is easy to show that

$$\Pr[\text{(B-1)}] = \sum_{(\mathsf{idx},w,i,z)\in\mathcal{Q}_{\mathcal{O}}} \frac{\left|\mathcal{Q}_{\mathsf{RO}}[i]\right|}{2^\rho} \le \mu \cdot \sum_{i\in\{0,1\}^k} \frac{\left|\mathcal{Q}_{\mathsf{RO}}[i]\right|}{2^\rho} = \frac{\mu p}{2^\rho},$$

where $\mathcal{Q}_{\mathsf{RO}}[i] := \{(x,y) : (x,i,y) \in \mathcal{Q}_{\mathsf{RO}}\}$ is defined similarly to Eq. (6). On the other hand, with C_i denoting the number of queries under i in $\mathcal{Q}_{\mathcal{O}}$ we have

$$\Pr[\text{(B-2)}] \le \sum_{i\in\{0,1\}^L} \frac{C_i(C_i-1)}{2} \cdot \frac{1}{2^\rho} \le (\mu-1) \sum_{i\in\{0,1\}^L} \frac{C_i}{2^{\rho+1}} \le \frac{(\mu-1)q}{2^{\rho+1}}$$

(using $C_i \le \mu$). Hence, the probability of a bad transcript in the ideal world is at most $\frac{\mu p}{2^\rho} + \frac{(\mu-1)q}{2^{\rho+1}}$.

The remaining analysis resembles that of the proof of Theorem 2, giving

$$\Pr[\forall(\mathsf{idx},w,i,z) \in \mathcal{Q}_{\mathcal{O}} : \mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}}}(w,i) = z \mid \mathsf{RO} \vdash \mathcal{Q}_{\mathsf{RO}}] = 1/2^{Lq},$$

and so the probability that the real world is consistent with the transcript is the same as (7). □

Inputs: A garbled circuit, along with input-wire labels $\{W_i\}_{i \in \mathsf{Inputs}}$.

Main algorithm:

1. Initialize a hash table $\mathcal{T} := \emptyset$. Evaluate the garbled circuit honestly and obtain a label W_a for each wire a.
2. For each AND gate (a, b, c, AND) with gate identifier gid and garbled table (T_G, T_E), set $j := 2 \times \mathsf{gid}$, compute $H_a := H(W_a, j) \oplus T_G$, and insert $(H_a \oplus j, j, W_a)$ into $\mathcal{T}$;
3. Repeatedly choose uniform $X^* \in \{0,1\}^k$ until there exists (j, W_a) such that $(\pi(X^*) \oplus X^*, j, W_a) \in \mathcal{T}$.
4. Given X^*, j, W_a from the previous step, compute $W^* := \pi^{-1}(X^* \oplus j)$ and $R := W_a \oplus W^*$. If $\mathsf{Check}(R) = 1$, output R; otherwise go to step 3.

$\mathsf{Check}(R)$**:**

1. For a gate (a, b, c, AND) with gate identifier gid and garbled table (T_G, T_E), let W_a, W_b, W_c be the labels computed on the respective wires.
2. If $(T_G, T_E, W_c \oplus wR) \stackrel{?}{=} \mathsf{GbAnd}(W_a^0 \oplus uR, W_b^0 \oplus vR, R, \mathsf{gid})$ for some values $u, v, w \in \{0,1\}$, output 1; else output 0.

Figure 4: Attack on the TMMO^π-based implementation of the half-gates scheme.

C Concrete Security of Using TMMO^π in HalfGates

In this section, we show an attack on the half-gates scheme that runs in time $O(2^k/C)$ when the underlying hash function is instantiated using the hash function from Guo et al. [18]. We concentrate on the single-instance setting for simplicity.

The construction TMMO defined by Guo et al. is based on a single permutation π, and is defined as

$$\mathsf{TMMO}^\pi(x, i) = \pi(\pi(x) \oplus i) \oplus \pi(x).$$

The attack is given in Figure 4. Essentially, the attack looks for a query (H_a, j, W_a) and a permutation query $\pi(X^*) = Y^*$ such that the evaluation of the former internally calls the latter. By construction, this means $H_a \oplus j = X^* \oplus Y^*$, which justifies our checking condition in Step 3. Given a query (H_a, j, W_a), for a uniform permutation query $\pi(X^*) = Y^*$, the probability to have a match $H_a \oplus j = X^* \oplus Y^*$ is $1/2^n$: the analysis is thus similar to that of Section 3.1.

D The H-Coefficient Technique

We provide a brief review of the H-coefficient technique, adapted from [18]. Fix a deterministic distinguisher D that is given access to an ideal cipher $E : \{0,1\}^L \times \{0,1\}^L \to \{0,1\}^L$, as well as u oracles $\mathcal{O}_1, \ldots, \mathcal{O}_u$ that can be of two types: in the real world they are functions that depend on u keys $R_1, \ldots, R_u$ sampled according to a distribution $\mathcal{R}$, while in the ideal world they are u functions chosen independently from $\mathsf{Func}_{\mathcal{W} \times \mathcal{T} \times \{0,1\}, \mathcal{W}}$. We are interested in bounding the maximum difference between the probabilities that D outputs 1 in the real world vs. the ideal world, where the maximum is taken over all D making p queries to E and q queries to its other oracles.

A transcript of D's interaction takes the form $\mathcal{Q} = (\mathcal{Q}_E, \mathcal{Q}_\mathcal{O}, \mathbf{R})$, where $\mathcal{Q}_E = \{(k_1, x_1, y_1), \ldots\}$ records D's queries/answers to/from E or E^{-1} (with $(k, x, y) \in \mathcal{Q}_E$ meaning $E(k, x) = y$, regardless of whether the query was to E or E^{-1}) and where $\mathcal{Q}_\mathcal{O} = \{(\mathsf{idx}_1, w_1, i_1, b_1, z_1), \ldots\}$ records D's queries/answers to/from the remaining oracles (where the tuple $(\mathsf{idx}, w, i, b, z) \in \mathcal{Q}_\mathcal{O}$ means that $\mathcal{O}_{\mathsf{idx}}(w, i, b) = z$). The keys $\mathbf{R} = (R_1, \ldots, R_u)$ are appended to the transcript as well (even though

they are not part of D's view) to facilitate the analysis: in the real world, these are the actual keys used by the oracles, whereas in the ideal world they are simply "dummy" keys sampled independently from $\mathcal{R}$. A transcript $\mathcal{Q}$ is *attainable* for some fixed D if there exist some oracles such that the interaction of D with those oracles would lead to transcript $\mathcal{Q}$.

Fix some D. Let $\mathcal{T}$ denote the set of attainable transcripts, let $\Pr_{\mathsf{real}}[\cdot]$ and $\Pr_{\mathsf{ideal}}[\cdot]$ denote the probabilities of events in the real and ideal worlds, respectively, and let $\mathcal{Q}^*$ denote the random variable corresponding to D's transcript. The H-coefficient technique involves defining a partition of $\mathcal{T}$ into a "bad" set $\mathcal{T}_{\mathsf{bad}}$ and a "good" set $\mathcal{T}_{\mathsf{good}} = \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$, and then showing that

$$\Pr_{\mathsf{ideal}}[\mathcal{Q}^* \in \mathcal{T}_{\mathsf{bad}}] \leq \varepsilon_1$$

and

$$\forall \mathcal{Q} \in \mathcal{T}_{\mathsf{good}} : \frac{\Pr_{\mathsf{real}}[\mathcal{Q}^* = \mathcal{Q}]}{\Pr_{\mathsf{ideal}}[\mathcal{Q}^* = \mathcal{Q}]} \geq 1 - \varepsilon_2.$$

The distinguishing advantage of D is then at most $\varepsilon_1 + \varepsilon_2$.

A key insight of the H-coefficient technique is that $\Pr_{\mathsf{real}}[\mathcal{Q}^* = \mathcal{Q}] / \Pr_{\mathsf{ideal}}[\mathcal{Q}^* = \mathcal{Q}]$ is equal to the ratio between the probability that the real-world oracles are consistent with $\mathcal{Q}$ and the probability that the ideal-world oracles are consistent with $\mathcal{Q}$. For each $v \in \{0,1\}^L$, define $\mathcal{Q}_E[v] \subseteq \mathcal{Q}_E$ as

$$\mathcal{Q}_E[v] \stackrel{\text{def}}{=} \{(x, y) : (v, x, y) \in \mathcal{Q}_E\}. \tag{6}$$

The probability that an ideal cipher (with L-bit blocks and L-bit keys) is consistent with the p queries in $\mathcal{Q}_E$ is exactly

$$\left(\prod_{v \in \{0,1\}^L} (2^L)_{|\mathcal{Q}_E[v]|} \right)^{-1},$$

where for integers $1 \leq b \leq a$, we set $(a)_b = a \cdot (a-1) \cdots (a-b+1)$, with $(a)_0 = 1$ by convention. For any attainable transcript $\mathcal{Q} = (\mathcal{Q}_E, \mathcal{Q}_{\mathcal{O}}, \mathbf{R})$, the probability that the ideal world is consistent with $\mathcal{Q}$ is always exactly

$$\frac{\Pr[\mathbf{R}]}{\prod_{v \in \{0,1\}^L} (2^L)_{|\mathcal{Q}_E[v]|} \cdot 2^{Lq}}. \tag{7}$$

(We assume $|\mathcal{Q}_{\mathcal{O}}| = q$, i.e., D always makes exactly q queries to its other oracles.) Bounding the distinguishing advantage of D thus reduces to bounding the probability that the real world is consistent with transcripts $\mathcal{Q} \in \mathcal{T}_{\mathsf{good}}$.

Let $E \vdash \mathcal{Q}_E$ denote the event that block cipher E is consistent with the queries/answers in $\mathcal{Q}_E$, i.e., that $E(v, x) = y$ for all $(v, x, y) \in \mathcal{Q}_E$. Since, in the real world, the behavior of the second oracle is completely determined by E and $\mathbf{R}$, we can also write $(E, \mathbf{R}) \vdash \mathcal{Q}_{\mathcal{O}}$ to denote the event that cipher E and keys $\mathbf{R}$ are consistent with the queries/answers in $\mathcal{Q}_{\mathcal{O}}$. For a (good) transcript $\mathcal{Q} = (\mathcal{Q}_E, \mathcal{Q}_{\mathcal{O}}, \mathbf{R})$, the probability that the real world is consistent with $\mathcal{Q}$ is exactly

$$\Pr\left[(E, \mathbf{R}) \vdash \mathcal{Q}_{\mathcal{O}} \mid E \vdash \mathcal{Q}_E\right] \cdot \Pr[E \vdash \mathcal{Q}_E] \cdot \Pr[\mathbf{R}]$$

(using independence of $\mathbf{R}$ and E). We have $\Pr[E \vdash \mathcal{Q}_E] = 1 / \prod_{v \in \{0,1\}^L} (2^L)_{|\mathcal{Q}_E[v]|}$ exactly as before. The crux of the proof thus reduces to bounding $\Pr\left[(E, \mathbf{R}) \vdash \mathcal{Q}_{\mathcal{O}} \mid E \vdash \mathcal{Q}_E\right]$. We can equivalently write this as $\Pr[\forall (\mathsf{idx}, w, i, b, z) \in \mathcal{Q}_{\mathcal{O}} : \mathcal{O}^{\mathsf{miTCCR}}_{R_{\mathsf{idx}}}(w, i, b) = z \mid E \vdash \mathcal{Q}_E]$.