Securely Instantiating 'Half Gates' Garbling in the Standard Model

Anasuya Acharya[1], Karen Azari[2] *, Mirza Ahad Baig[3], Dennis Hofheinz[4], and Chethan Kamath[5]

[1] Bar-Ilan University, Israel
anasuyahirai@gmail.com

[2] University of Vienna, Faculty of Computer Science, Vienna, Austria
karen.azari@univie.ac.at

[3] ISTA, Austria
mirzaahad.baig@ist.ac.at

[4] ETH Zurich, Switzerland
hofheinz@inf.ethz.ch

[5] IIT Bombay, India
ckamath@cse.iitb.ac.in

Abstract. Garbling is a fundamental cryptographic primitive, with numerous theoretical and practical applications. Since the first construction by Yao (FOCS'82, '86), a line of work has concerned itself with reducing the communication and computational complexity of that construction. One of the most efficient garbling schemes presently is the 'Half Gates' scheme by Zahur, Rosulek, and Evans (Eurocrypt'15). Despite its widespread adoption, the provable security of this scheme has been based on assumptions whose only instantiations are in idealized models. For example, in their original paper, Zahur, Rosulek, and Evans showed that hash functions satisfying a notion called circular correlation robustness (CCR) suffice for this task, and then proved that CCR secure hash functions can be instantiated in the random permutation model.

In this work, we show how to securely instantiate the Half Gates scheme in the standard model. To this end, we first show how this scheme can be securely instantiated given a (family of) *weak* CCR hash function, a notion that we introduce. Furthermore, we show how a weak CCR hash function can be used to securely instantiate other efficient garbling schemes, namely the ones by Rosulek and Roy (Crypto'21) and Heath (Eurocrypt'24). Thus we believe this notion to be of independent interest. Finally, we construct such weak CCR hash functions using indistinguishability obfuscation and one-way functions. The security proof of this construction constitutes our main technical contribution. While our construction is not practical, it serves as a proof of concept supporting the soundness of these garbling schemes, which we regard to be particularly important given the recent initiative by NIST to standardize garbling, and the optimizations in Half Gates being potentially adopted.

Keywords: Garbling Schemes · Circular Correlation Robust (CCR) Hashing · Indistinguishability Obfuscation

* Parts of the work were done while the author was affiliated with ETH Zurich, Switzerland.

1 Introduction

Garbling allows encoding a circuit $\mathbf{C}$ and an input $\mathbf{x}$ into a garbled circuit $\mathbf{F}$ and garbled input $\mathbf{X}$ such that given $\mathbf{F}$ and $\mathbf{X}$ one can "evaluate" $\mathbf{F}$ on $\mathbf{X}$ to learn $\mathbf{C}(\mathbf{x})$ without leaking anything else about the input. Since its introduction by Yao [56], the primitive has turned out to be useful not only in theory (e.g., [5,6,26,54]), but also in practice (e.g., [15,48,51]). Yao's original construction, based on symmetric-key encryption (SKE), roughly costs four ciphertexts per gate in terms of communication and four SKE operations per gate in terms of computation, for both garbling and evaluation, each. While this is already quite efficient, a line of work [12, 29, 44, 45, 49, 51, 52, 57] (see Section 1.3 for an in-depth overview) has further boosted efficiency, resulting in extremely efficient constructions. One of the most efficient is the hash-based scheme by Zahur, Rosulek, and Evans [57], henceforth referred to as the 'Half Gates' scheme. The Half Gates scheme reduces the cost of garbling drastically:

- garbling XOR gates is essentially free of charge,
- the communication cost for AND gates is reduced to two "ciphertexts", and
- the number of hash operations required for evaluations is just two (garbling still requires four hash operations).

We have, by now, a reasonably good understanding of the landscape of security of Yao's construction [36,38,39,41,46] – in particular, its security (in a strong sense) can be based on the existence of one-way functions. However, the same is not quite the case for its optimizations. With the aim of proving the security of schemes that employ the so-called "free-XOR" optimizations [4,45,50], Choi et al. [24] introduced a new security notion for hash functions, called circular correlation robustness (CCR). Informally, CCR security of hash functions is defined with respect to a random, secret "offset" to which the adversary is given oracle access to: it can adaptively query the oracle on any input of its choice and gets as response the offset hash (circularity) of the offset input (correlation). It is required that these responses appear pseudo-random to the adversary. This notion turned out to be quite powerful and the security of several other constructions, including Half Gates, was later based on it. However, despite an extensive body of work on this topic, all instantiations of CCR hash functions are in idealized models like random-oracle or random-permutation model [23,30,31,57]. Thus, in this work, we ask the following question:

> *Are optimization techniques (as used in Half Gates) instantiable in the standard model, or do they rely inherently on idealized models?*

We consider this question particularly important in light of the recent effort by NIST to standardize multi-party threshold schemes, as part of which garbled circuits have been listed as a 'Category 2' primitive [20]. Once standardized, the scheme can be expected to be used extensively, and it is, therefore, crucial to better understand the precise security guarantees of the prominent proposals.

To highlight the difficulty of the task at hand, consider plain correlation robustness (CR), i.e., CCR without the circularity requirement [35] where the adversary can query an oracle and receive only a hash of the offset input. Even this simpler notion seems hard to achieve: we are only aware of two works that attempt to instantiate CR in the standard model. One is the work of Applebaum, Harnik, and Ishai [4], who construct CR hash functions over groups that support a certain q-type Diffie-Hellman assumption. However, it does not support the XOR operations (over binary strings) necessary for the Half Gates scheme. The other is the construction of Abdalla, Benhamouda, and Passelègue [1] which requires non-standard assumptions on multi-linear maps.

1.1 Our Results

We answer the above question positively by proving the following theorem for the already-mentioned Half Gates scheme.

Theorem 1 (Informal, see Corollary 1). *Assuming indistinguishability obfuscation (iO) for circuits and one-way functions (OWFs) exist, the Half Gates scheme from [57] can be securely instantiated in the standard model.*

Here, by "securely" we refer to the *selective simulatability* model of security for garbling schemes [14]. As a consequence of Theorem 1, the Half Gates scheme can be instantiated under the same hardness assumptions that underlie recent constructions of iO [40, 55].

Moreover, a result similar to Theorem 1 can be shown for two more optimized garbling schemes. The first is the so-called "Three Halves" scheme of Rosulek and Roy [52]. This scheme supports the free-XOR optimization and can garble AND gates using roughly 1.5κ bits (where κ denotes the security parameter). The second scheme is by Heath [33], who presented a garbling scheme for an arithmetic model of computation called oblivious switch systems. In his scheme, garbling for addition gates is free and the garbling table for each multiplication gate has size $O(\ell\kappa)$ bits, where ℓ is the size of the bit-representation for the underlying arithmetic field.

Theorem 2 (Informal, see Theorems 6 and 7). *Assuming indistinguishability obfuscation (iO) for circuits and one-way functions (OWFs) exist, the garbling schemes from [33, 52] can be securely instantiated in the standard model.*

We prove Theorems 1 and 2 in two steps. First, we show that a weaker form of CCR suffices to instantiate this scheme in the standard model. That is:

Theorem 3 (Informal, see Theorems 5 to 7). *Assuming a family of* weakly *CCR (wCCR) hash functions exists, the schemes from [33, 52, 57] can be securely instantiated in the standard model.*

On a high level, wCCR requires CCR to hold only with respect to *semi-adaptive* query strategies (defined soon in Section 1.2). Such a query strategy is weaker than the (fully) adaptive query strategies that CCR allows (which is why CCR seems hard to instantiate), but stronger than a selective query strategy where the adversary must submit its queries in advance (which turns out to be too weak to prove security). Formalizing this notion, we believe, is our main conceptual contribution.

In the second step, we propose a family of wCCR hash functions based on iO and OWF. This renders all the garbling schemes mentioned above secure in the standard model (as opposed to relying on ideal assumptions).

Theorem 4 (Informal, see Theorem 8). *Assuming iO and OWFs, a family of wCCR hash functions exists.*

While our construction of wCCR hash family is simple, proving it wCCR turned out to be involved and is our main technical contribution. We discuss the construction and outline the proof in Section 1.2.

Discussion of our results.

- *Efficiency vs. feasibility.* The wCCR hash function from Theorem 4 is not efficient at all. Firstly, it relies on iO. Secondly, as we will see in Section 1.2, its description grows proportionately to the description of the query strategy. This implies, e.g., that the description of the hash function in our application to Half Gates (and hence the communication complexity) grows with the size of the circuit to be garbled. Nevertheless, it does show that all optimization techniques used in the Half Gates scheme can be instantiated in the standard model. Thus our results serve as a proof of concept in the vein of Applebaum [3], who showed that the free XOR optimization can be instantiated from related-key key-dependant message (RK-KDM) secure SKE, a notion that is analogous to CCR. However, as mentioned in [3], the hash-based construction of Half Gates requires new techniques.
- *Comparing settings.* Prior works on security of Half Gates and similar constructions [23, 24, 30, 31, 57] consider security in idealised settings (e.g., ideal permutation model) where the hash function is modeled as an idealized object available to all parties [13]. This reflects practice where the hash function is instantiated, e.g., using fixed-key AES which supports fast hardware implementations. The setting we consider is slightly different. We assume that the party that garbles the circuit samples the hash function used in the Half Gates construction from a *family*, and then sends its description over to the party that evaluates, as part of the garbled circuit.
- *Weak CCR vs. collision resistance.* Note that Theorem 4 and [8] together imply that wCCR hash functions cannot imply collision-resistant hash functions in an (oracle-aided) black-box sense; we consider this observation interesting from the viewpoint of understanding the hierarchy of cryptographic primitives.

Open questions. Our work leaves several fascinating questions open, some of which are listed below.

- *Efficiency.* As already pointed out, the description of our weak CCR hash function in Theorem 4 grows proportionately to the complexity of the query strategy. Avoiding this blow-up altogether, or recognizing settings where it can be reduced is an interesting technical question, and resolving it would imply improved communication complexity.
- *De-obfuscation.* A natural follow-up question to our work is to replace iO with weaker primitives or weaker computational assumptions. Compare this with the analogous notion of RK-KDM security, which can in fact be instantiated under standard assumptions [3, 17].
- *Adaptive security.* Recall that we prove security only in the weaker selective model of garbling. Very recently, the adaptive security of Half Gates in idealized models was shown in [11, 32], but nothing else is known about its adaptive security. This is in contrast with the adaptive security of Yao's construction, which is a well-studied topic [7, 36–39, 41, 42]. Therefore, it is natural to ask whether our techniques can be lifted to prove adaptive security in the standard model. While this won't be possible by a black-box reduction to the weak CCR security of the hash function, our techniques could still be useful to base adaptive security on standard assumptions in a non-black-box manner. This however, could require significantly newer ideas.

1.2 Technical Overview

In this technical overview we focus on the Half Gates scheme [57], since this already allows us to present all relevant techniques. For the details on the garbling schemes in [52] and [33] we

refer the reader to Appendix C and D respectively. We start off with an informal description of the relevant parts of the Half Gates construction (see Algorithms 1 and 2 for a formal description), and then explain how (full) CCR is useful to prove its security. Next we describe wCCR, and explain why it is sufficient to prove security of the Half Gates scheme in the standard model. Finally, we describe our construction of a wCCR hash function family.

Half Gates and CCR. Recall that the Half Gates scheme is a hash-based optimisation of Yao's construction. In Yao's scheme, to garble a circuit $\mathbf{C}$, one first *labels* each wire in $\mathbf{C}$ with a pair of randomly-sampled keys, logically associated with bits 0 and 1, respectively. We refer to these as the 0- and 1-label, respectively. Then these wire labels are used to generate a *garbling table* for each gate in $\mathbf{C}$, which encodes its gate table – in Yao's construction a garbling table consists of four (double) ciphertexts of the underlying SKE. The garbled circuit $\mathbf{F}$ consists of all the garbling tables along with an *output map* from labels of output wires to bits, which is required later for evaluation (for now, let's ignore the specifics of evaluation). The garbled input $\mathbf{X}$ consists of those labels of input wires of $\mathbf{C}$ that correspond to the input $\mathbf{x}$. The Half Gates construction from [57] optimises the above construction in two ways:

- Firstly, it employs the so-called "free-XOR" optimisation [45], which means that the garbling table for XOR gates can be *empty*. To implement this, the wire labels are *correlated* via a global *offset* Δ. To be specific, the 1-label of a wire C, L_C^1, is always set to be an XOR of its 0 label, i.e., $\mathsf{L}_C^1 := \mathsf{L}_C^0 \oplus \Delta$. In addition, the 0-label of an XOR gate, with input wires (A, B) and output wire C is generated *deterministically* as $\mathsf{L}_C^0 := \mathsf{L}_A^0 \oplus \mathsf{L}_B^0$ (thus the output label for XOR gates during evaluation can be obtained by simply XORing the given input labels).
- However, the garbling table for AND gates in the above optimisation still has to be generated as in Yao's scheme, and thus requires four ciphertexts per gate. To reduce this, a clever "row reduction" for AND gates is additionally employed which results in a garbling table consisting of just two "ciphertexts". To implement this, the 0-label of an AND gate, with input wires (A, B) and output wire C, is also now generated deterministically using the underlying hash H as follows (for this overview, let's ignore the 'tweaks')

$$\mathsf{L}_C^0 = \mathsf{H}(\mathsf{L}_A^0 \oplus p_a\Delta) \oplus \mathsf{H}(\mathsf{L}_B^0 \oplus p_b\Delta) \oplus p_a p_b \Delta. \tag{1}$$

  Here, p_a (resp., p_b) is the so-called permutation bit for A (resp., B) that is multiplied to every bit in Δ. In garbling schemes like the half Gates construction where the evaluation steps depend on an 'apparent value' $\in \{0, 1\}$ of the labels available to the evaluator, these permutation bits work to mask the actual semantic values of the labels, making this masked apparent value information theoretically hiding. The garbling table for an AND gate with input wires (A, B) has the form

$$G_{\mathsf{AND}}^0 = \mathsf{H}(\mathsf{L}_A^0) \oplus \mathsf{H}(\mathsf{L}_A^0 \oplus \Delta) \oplus p_b\Delta \qquad G_{\mathsf{AND}}^1 = \mathsf{H}(\mathsf{L}_B^0) \oplus \mathsf{H}(\mathsf{L}_B^0 \oplus \Delta) \oplus \mathsf{L}_A^0. \tag{2}$$

  The former ciphertext is a function of the hashes of labels of wire A only whereas the latter ciphertext contains hashes of labels of wire B only; hence the '2 halves'. For labels L_A and L_B available during evaluation, letting $s_a = \mathsf{lsb}(\mathsf{L}_A)$ and $s_b = \mathsf{lsb}(\mathsf{L}_B)$ be the 'apparent values' of these wires, the AND gate is evaluated as:

$$\mathsf{L}_C = \mathsf{H}(\mathsf{L}_A) \oplus \mathsf{H}(\mathsf{L}_B) \oplus s_a G_{\mathsf{AND}}^0 \oplus s_b(G_{\mathsf{AND}}^1 \oplus \mathsf{L}_A)$$

  This effectively cancels the appropriate hash values within the ciphertexts using those of the active labels and applies Δ only when the inputs values are $(1, 1)$.

We now outline how CCR can be used to show security of Half Gates. Recall that, informally, security requires $\mathbf{F}$ and $\mathbf{X}$ not to leak anything about the input except for $\mathbf{C}(\mathbf{x})$. This is modelled by requiring the distribution of $(\mathbf{F}, \mathbf{X})$ to be efficiently *simulatable* given just $\mathbf{C}$ and $\mathbf{C}(\mathbf{x})$. The simulation for the Half Gates scheme, on a high level, proceeds similar to that for Yao's scheme [46]. That is, the garbling tables are generated assuming that the input is the all-0 string (which implies that all internal wires evaluate to 0), and the output map is then "programmed" so that the 0-labels of the output wires map to $\mathbf{C}(\mathbf{x})$ (which ensures that evaluation is consistent). Thus all the 1-labels – and therefore the offset Δ – is unused and can be replaced by random values.

The definition of CCR is "tailored" to the specific goal of showing that the distribution of the simulation above is indistinguishable from the real distribution $(\mathbf{F}, \mathbf{X})$. To elaborate, a hash function H is CCR if the 'CCR oracle'

$$\mathcal{C}_{\mathsf{H},\Delta}(x, b) := \mathsf{H}(x \oplus \Delta) \oplus b\Delta, \tag{3}$$

for a (secret) random string Δ, appears pseudorandom to any bounded "legal" distinguisher given adaptive access to the oracle.[6] To see why CCR suffices for showing indistinguishability of simulation, observe that the garbling table from Eq. (2) can be rewritten in terms of the 0-labels as

$$\mathsf{H}(\mathsf{L}_A^0) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_A^0, p_b) \qquad \mathsf{H}(\mathsf{L}_B^0) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_B^0, 0) \oplus \mathsf{L}_A^0. \tag{4}$$

As a result, indistinguishability of $(\mathbf{F}, \mathbf{X})$ and the above-described simulated distribution can be shown based on pseudo-randomness of CCR. Several issues have been brushed under the rug here – we refer the readers to Section 3.1 for a more formal treatment.

Formulating weak CCR. Our starting point is the observation that allowing the adversary to access the CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ (Eq. (3)) at *arbitrary* input points, as in CCR, is an overkill when proving security in the standard model. In particular, for the case of security of Half Gates, it suffices to provide restricted access to $\mathcal{C}_{\mathsf{H},\Delta}$, at the correlated input points from Eqs. (1) and (2), determined by the construction. The only way that the adversary can influence the queries is via the choice of circuit and input. To reflect this, we define a weak form of CCR, captured by the following game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

1. $\mathcal{A}$ first submits a *query strategy* $\mathsf{QueryStrategy}$ to $\mathcal{C}$. This can be any oracle-aided randomised (polynomial-sized) circuit ($\mathcal{A}$ cannot control the random coins, however).[7]
2. $\mathcal{C}$ randomly samples a hash function H, a bit b and the global offset Δ, and then executes $\mathsf{QueryStrategy}$, answering its queries either using $\mathcal{C}_{\mathsf{H},\Delta}$ if $b = 0$ or using a random oracle (via lazy sampling) if $b = 1$. Then it returns (the description of) H, along with all the query-answer pairs, $\mathcal{Q}$, to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess b' and wins if it is correct (i.e., $b' = b$).

Note that $\mathcal{A}$'s access to the CCR oracle is *partly adaptive* in the sense that it has some control over them via the choice of $\mathsf{QueryStrategy}$, but not full control as in the definition of

[6] Informally, a distinguisher is legal if it does not trivialise the task, e.g., by querying both $\mathcal{C}_{\mathsf{H},\Delta}(x, 0)$ and $\mathcal{C}_{\mathsf{H},\Delta}(x, 1)$ to learn Δ. There are two definitions of CCR in the literature, the original one from [24] and a slightly weaker one from [57]. The definition of "legal" depends on the exact definition in consideration. We refer the reader to Remarks 1 to 3 for a detailed discussion.

[7] That query sequences produces through $\mathsf{QueryStrategy}$ are indeed "legal", as discussed in Footnote 6, will be guaranteed by implementing the 'tweak' (which we ignored in this overview for simplicity of exposition) as a counter.

CCR. We show that this notion nevertheless suffices to prove selective security of Half Gates. Once $\mathsf{QueryStrategy}$ is tailored to the scheme, the proof is a straightforward adaptation of the proof using CCR outlined earlier. We refer the reader to Section 3.1 for more details.

Constructing wCCR hash functions. We construct wCCR hash functions using iO and OWF. The construction is described using *puncturable* pseudo-random functions (PPRFs), which are implied by OWFs [18, 19, 43]. Very briefly, a PRF $\{F_k : \{0,1\}^\kappa \to \{0,1\}^\kappa\}_{k\in\{0,1\}^\kappa}$ is a PPRF if it additionally supports *puncturing* of any key k at any input $x^* \in \{0,1\}^\kappa$, such that: 1) the punctured key k_{x^*} preserves functionality of k on inputs $x \neq x^*$; and 2) the evaluation of F_k is pseudorandom at x^* (even given k_{x^*}).[8] Given such a PPRF, our construction is simple: to sample a hash function from the family, first sample a random PPRF key k and output the circuit

$$\mathsf{H}(x) \leftarrow \mathsf{iO}(F_k(x)).$$

To prove that this construction is weakly CCR, we need to show that

$$\left\{\mathsf{iO}(F_k), \mathcal{Q} := ((x_i, i, b_i, y_i))_{i\in[Q]}\right\} \overset{c}{\approx} \left\{\mathsf{iO}(F_k), \mathcal{Q}^* := ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right\}, \tag{5}$$

where $\mathcal{Q}$ (resp., $\mathcal{Q}^*$) denotes the query-answer pairs obtained by executing $\mathsf{QueryStrategy}$ with the oracle instantiated using the CCR oracle (resp., random oracle), and $\overset{c}{\approx}$ denotes computational indistinguishability. Let's refer to these two distributions by 'correlated' and 'random', respectively. More formally, for $i \in [Q]$, (x_i, b_i) is the i-th query generated by $\mathsf{QueryStrategy}$ (which may depend on prior answers as well as the random coins), and

- in the correlated distribution, $y_i := \mathsf{H}(x_i \oplus \Delta) \oplus b_i\Delta$ for a global offset Δ; and
- in the random distribution, y_i^* is a uniformly random value.

The high level idea to prove Eq. (5) is to use a hybrid argument to iteratively replace the y_is with y_i^*s. The difficulty lies in dealing with the correlation introduced by Δ. To handle this, we proceed in two stages.

Stage I. First we show that the correlated distribution is indistinguishable from an 'intermediate' distribution

$$\left\{\mathsf{iO}(F_k^*), \mathcal{Q}^* := ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right\}$$

in which the answers have been switched to random y_i^*s (as in the random distribution) at the expense of altering the hash function to $\mathsf{iO}(F_k^*)$. Here, F_k^* is a function related to F_k, where the evaluation of F_k at points $(x_1 \oplus \Delta, \cdots, x_Q \oplus \Delta)$ is reprogrammed to $(y_1^* \oplus b_1\Delta, \cdots, y_Q^* \oplus b_Q\Delta)$, respectively. To carry this out, we roughly follow the *punctured programming* approach from [53]. To switch the first answer to random:

1. Program[9] F_k at $x_1 \oplus \Delta$. That is, if the input to H is $x_1 \oplus \Delta$ (where x_1 gets fixed once $\mathsf{QueryStrategy}$ and its coins are fixed) output a *hardcoded* value $F_k(x_1 \oplus \Delta)$; on all other inputs invoke $F_{k_{x_1\oplus\Delta}}$ where, if you recall, $k_{x_1\oplus\Delta}$ denotes a key punctured at $x_1 \oplus \Delta$. Since the functionality of F_k is not affected by this switch, the resulting distribution is indistinguishable from the correlated distribution due to security of iO.

[8] More generally, PPRFs support puncturing at an arbitrary (polynomial-sized) set $S \subseteq \{0,1\}^\kappa$. But for our work, puncturing at a single point suffices.

[9] In the context of punctured programming, we use 'programming' to refer to hard-coding of a input point to some output. We use 'reprogramming' when this hard-coding is changed to a random output.

2. Reprogram the hardcoded value from $F_k(x_1 \oplus \Delta)$ to $y_1^* \oplus b_1\Delta$, for a random y_1^*. This switch is indistinguishable thanks to pseudo-randomness of the PPRF.
3. Switch the punctured key back to a normal key. As in Item 1 and since we keep the programmed random point, iO security implies that this switch is indistinguishable.

The rest of answers can be switched similarly, by iterating the above process. Notice that at this point, Δ is used *solely* for the purpose of reprogramming in F_k^*. To be specific, it is used to check whether or not the input x equals $x_i \oplus \Delta$ for some $i \in [Q]$, and to compute the list of hardwired answers $(y_1^* \oplus b_1\Delta, \cdots, y_Q^* \oplus b_Q\Delta)$. This will be key to "forgetting" Δ in the second stage.

Stage II. Next we revert to $\mathsf{iO}(F_k)$ by undoing the reprogramming of F_k^*, and thus end up with the target random distribution. To this end, let's assume we are given an *injective* pseudorandom generator (iPRG) G. Recall that the only use of Δ is to check if

'$x = x_i \oplus \Delta$ for some $i \in [Q]$',

or equivalently

'$x \oplus x_i = \Delta$ for some $i \in [Q]$'.

As a first step to forgetting Δ, this check is replaced with an *indirect* check

'$G(x \oplus x_i) = G(\Delta)$ for some $i \in [Q]$'.

Since G is injective, the functionality remains unchanged and this switch is indistinguishable thanks to iO security. Now, we can exploit pseudo-randomness of G to switch the check to

'$G(x \oplus x_i) = R$ for some $i \in [Q]$'

for a random R in the co-domain of G. Since R is outside the image of G with overwhelming probability, the check fails with overwhelming probability, and therefore it can be dropped altogether. Since the resulting function is functionally equivalent to F_k, we can rely on iO security to switch back to $\mathsf{iO}(F_k)$, and thus end up with the random distribution from Eq. (5). We note that the overall approach in second stage is reminiscent of *hidden sparse triggers* from [53]. The main additional challenge in our case is that we need to change the functionality provided by F_k on *many* inputs $x_i \oplus \Delta$ for known x_i and unknown Δ. The crucial observation here is that it is sufficient to provide a check for Δ (resp. $G(\Delta)$) as above.

There are two points that are now left to be discussed. Firstly, we use an iPRG in the second stage of the construction, whereas to complete the proof of Theorem 4 we need to rely only on iO and OWF. In the main body of the paper (Section 4) we show how to relax the requirement from iPRG to *injective* OWFs. Since iOWFs are implied by iO and OWFs [16], Theorem 4 follows. Second, notice that the dependence of the description of our hash function on $\mathsf{QueryStrategy}$ stems from having to maintain the hardcoded list that tracks the reprogramming: to be able to invoke iO security later in the security proof, $\mathsf{iO}(F_k)$ must be initially padded with a dummy value proportional to the size of this list which, in turn, depends on $\mathsf{QueryStrategy}$. Avoiding this dependence is an interesting open question (see Section 1.1).

1.3 Related Work

Circuit garbling, in general. Circuit garbling was introduced by Yao in [56] as a technique for secure two-party computation. Later, in [14], Bellare Hoang, and Rogaway provided formal syntax and security definitions for garbling as a standalone cryptographic primitive. Yao's scheme (in fact, there exist several variants of this very prominent construction) was proven (selectively) secure by Lindell and Pinkas [46] in a quite strong sense: Not only the input to the circuit remains private, but also the circuit itself, apart from its topology which is leaked because Yao's scheme garbles each gate individually.

Optimized schemes. Towards improving the computational cost of circuit evaluation, Naor, Pinkas and Sumner [49] proposed the so-called point-and-permute technique, which allows evaluation of the garbled circuit at the cost of only a single decryption per gate (as opposed to decrypting all four ciphertexts associated to a gate, as in Yao's original construction). In [45], Kolesnikov and Schneider proposed a method that allows to garble XOR gates for free, i.e. evaluation of XOR gates does not require any additional information anymore. Clearly, this comes at the cost of losing circuit privacy. However, most prominent applications do not require circuit privacy. A series of works [29, 49, 51] focused on reducing the number of ciphertexts per gate, independent of the gate's functionality. Unfortunately only the first of these is compatible with the free-XOR technique.

The scheme used in most practical applications [21, 22, 25, 30, 31, 47] is the Half Gates scheme by Zahur, Rosulek and Evans [57]. It allows garbling XOR gates for free and reduces the complexity of AND gates to just two ciphertexts. Several further constructions improve over Half Gates for specific types of computation [9] or in the number of ciphertexts per gate [52], or suggest a new approach for garbling that supports free-XOR [2]. Finally, the very recent work of Heath, Kolesnikov, and Ostrovsky [34] introduces a novel model of computation, called tri-state circuits, that allows efficient evaluation of RAM programs.

Security of optimized schemes. All the optimized garbling constructions above only consider selective security, i.e. a setting where circuit and input are garbled at the same time (as opposed to stronger adaptive security, where the circuit is garbled in an offline pre-computation phase before the input is even known). Furthermore, they rely either on idealized models such as the random oracle/ permutation model or on novel and very strong assumptions on the underlying building block they are based on (i.e. symmetric-key encryption or hash functions). In particular, the security of Half Gates relies on circular correlation robust (CCR) hash functions.

The notion of CCR was introduced by Choi et al. [24] and is strictly stronger than the notion of correlation robustness that was introduced by Ishai et al. in [35]. There exists very efficient constructions of CCR secure hash functions in the random permutation model [23, 30, 31], however in the standard model even the simpler notion of correlation robustness seems hard to achieve: The relatively simple, Diffie-Hellman based construction of PRGs that are secure under related-key attacks by Applebaum, Harnik, and Ishai [4] does not support does not support the use of keys which are related by an additive offset. The only known standard-model construction of (plain) correlation robust hash functions was proposed by Abdalla, Benhamouda and Passelègue [1] and is based on non-standard assumptions on multi-linear maps.

There exists a single work addressing the security of optimized garbling techniques in the standard model: in [3], Applebaum shows that the free-XOR technique can be securely

implemented if one uses a symmetric-key encryption scheme that is secure under binary linear related-key key-dependent message (RK-KDM) attacks. He also shows that this security notion is a strict strengthening of related key attacks and key-dependent message attacks and provides a first construction based on the Learning Parity with Noise assumption. Böhl, Davies, and Hofheinz propose further constructions of RK-KDM secure encryption from various assumptions. Unfortunately, Applebaum's approach is not compatible with row-reduction techniques as used in Half Gates. Furthermore, while RK-KDM security for encryption shares strong similarity with CCR security for hash functions, the techniques used to construct RK-KDM secure encryption crucially rely on the use of randomness and therefore don't seem applicable in the setting of (deterministic) hash functions.

2 Preliminaries

Notation. Throughout, we denote the computational security parameter by $\kappa \in \mathbb{N}$. Going forward, all function ensembles are indexed by $\kappa \in \mathbb{N}$ and all functions within them are indexed by $i \in \mathbb{N}$, though this notation may be omitted where this is implicit. Let bold-face letters represent vectors, e.g. $\mathbf{v}$, where $\mathbf{v}[k]$ refers to the element in the k^{th} index of the vector $\mathbf{v}$ and $|\mathbf{v}|$ refers to the number of elements in $\mathbf{v}$. For an integer $n \in \mathbb{N}$ let $[n] := \{1, 2, \ldots, n\}$. Further for an $m < n, m \in \mathbb{N}$ let $[m : n] = [m, m+1, \cdots, n]$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if for all constants $c > 0$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$. We sometimes use $\epsilon(\cdot)$ to denote negligible functions.

Circuit Notation. We represent a Boolean circuit $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ as a tuple of the form $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{q}, \{A, B, C, \mathsf{f}_g\}_{g \in [\mathsf{q}]})$ with $\mathsf{f}_g \in \{\mathsf{AND}, \mathsf{XOR}\}$ and $A, B, C \in [\mathsf{n} + \mathsf{q}]$. Here q represents the number of gates, n and m the input and output lengths, respectively. The circuit wires are numbered in topological order where those numbered in the first n are also the circuit input wires. Every wire beyond that is an output from some gate in the circuit. For each (binary) gate indexed $g \in [\mathsf{q}]$, A and B represent the indices of its left and right input wires respectively, while $C = \mathsf{n} + g$ represents the output wire index. f_g represents the functionality of the gate. Note that, in particular, the representation of $\mathbf{C}$ exceeds the number of gates. Looking ahead, this is important for garbling to work in polynomial time.

2.1 Cryptographic Primitives

In this section, we define the cryptographic primitives that we rely on in the paper, namely hardcore predicates, puncturable PRFs, indistinguishability obfuscator, CCR hash function and garbling. We refer the reader to Appendix A for additional preliminaries.

Hardcore Predicates. We introduce the notion of hard-core predicates, which relates one-wayness with pseudo-randomness.

Definition 1 (Hardcore Predicate). *For polynomial functions $\mathsf{n}(\cdot)$ and $\mathsf{m}(\cdot)$, let $f = \{f_\kappa : \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}^{\mathsf{m}(\kappa)}\}_{\kappa \in \mathbb{N}}$ be an ensemble of one-way functions. An ensemble of polynomial-time computable predicates $h = \{h_\kappa : \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}\}_{\kappa \in \mathbb{N}}$ is* hard-core *for f if for any PPT algorithm A, there exists a negligible function $\epsilon(\cdot)$ such that for all $\kappa \in \mathbb{N}$*

$$\Pr_{x \leftarrow \{0,1\}^{\mathsf{n}(\kappa)}}[\mathsf{A}(1^\kappa, f_\kappa(x)) = h_\kappa(x)] \leq \frac{1}{2} + \epsilon(\kappa).$$

Goldreich and Levin [28] showed that every one-way function has a hard-core predicate.

Puncturable pseudo-random functions (PPRFs). A PPRF [53] is a PRF that additionally supports *puncturing* of any key k at any input x^*, such that: 1) the punctured key k^* preserves functionality on non-punctured inputs $x \neq x^*$; and 2) the evaluation of the PRF at x^* is pseudorandom even given k^*. A formal definition is given below.

Definition 2 (PPRF). *Let $\mathsf{n}(\cdot)$ and $\mathsf{m}(\cdot)$ be polynomial functions. A family of puncturable PRFs $\mathcal{F} = \{F_\kappa : \{0,1\}^\kappa \times \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}^{\mathsf{m}(\kappa)}\}_{\kappa \in \mathbb{N}}$ is defined through a triple of polynomial-time algorithms $(\mathsf{Key}_F, \mathsf{Puncture}_F, \mathsf{Eval}_F)$, where the latter two are deterministic and the former is randomized, satisfying the following syntax:*

- $\mathsf{Key}_F(1^\kappa)$ *on input 1^κ samples a key $k \leftarrow \{0,1\}^\kappa$ uniformly at random.*
- $\mathsf{Puncture}_F(k, x)$ *on input a key $k \in \{0,1\}^\kappa$ and a point $x \in \{0,1\}^{\mathsf{n}(\kappa)}$ outputs a punctured key k_x.*
- $\mathsf{Eval}_F(k, x)$ *on input a (possibly punctured) key k and an input $x \in \{0,1\}^{\mathsf{n}(\kappa)}$ outputs a value $y \in \{0,1\}^{\mathsf{m}(\kappa)}$.*

We require the following correctness and security properties:

- Functionality Preserved under Puncturing: *For all $\kappa \in \mathbb{N}$, all $x^* \in \{0,1\}^{\mathsf{n}(\kappa)}$, all $x \in \{0,1\}^{\mathsf{n}(\kappa)} \setminus \{x^*\}$, all keys $k \in \{0,1\}^\kappa$, and punctured key $k_{x^*} := \mathsf{Puncture}_F(k, x^*)$, we have that $F_\kappa(k, x) = \mathsf{Eval}_F(k, x) = \mathsf{Eval}_F(k_{x^*}, x)$.*
- Pseudorandomness at Punctured Point: *For every pair $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ of PPT adversaries, consider the following setup:*
  1. *$(x^*, \sigma) \leftarrow \mathsf{A}_1(1^\kappa)$ (where σ is a state to be passed from A_1 to A_2),*
  2. *$k \leftarrow \mathsf{Key}_F(1^\kappa)$ and $k_{x^*} = \mathsf{Puncture}_F(k, x^*)$,*
  3. *$y_0 = \mathsf{Eval}_F(k, x^*)$ and $y_1 \leftarrow \{0,1\}^{\mathsf{m}(\kappa)}$.*

  Given this setup, we require that the advantage

$$Adv_{\mathsf{A}}^{\mathsf{PPRF}}(\kappa) = \frac{1}{2}\left|\Pr[\mathsf{A}_2(\sigma, k_{x^*}, y_0) = 1] - \Pr[\mathsf{A}_2(\sigma, k_{x^*}, y_1) = 1]\right|$$

  is negligible, where the probability is taken over the randomness of Key_F and the randomness of A

For ease of notation we write $F_k(x)$ to represent $\mathsf{Eval}_F(k, x) = F_\kappa(k, x)$.

Note that the Goldreich-Goldwasser-Micali PRF [27] is known to be puncturable in the above sense [18, 19, 43]. Hence, PPRFs exist if and only if one-way functions exist.

Indistinguishability Obfuscation. Indistinguishability Obfuscation (iO) [10, 53] guarantees that the obfuscations of two circuits are computationally indistinguishable as long as they are functionally equivalent, i.e., the output of both circuits are the same on every input.

Definition 3 (Indistinguishability obfuscator (iO) for Circuits). *A uniform PPT algorithm iO is called an obfuscator for polynomial-sized circuits if,*

- Completeness: *Let $\mathsf{n}(\cdot)$ and $\mathsf{m}(\cdot)$ be polynomial functions. For every $\kappa \in \mathbb{N}$, every polynomial-sized circuit $\mathbf{C}_\kappa : \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}^{\mathsf{m}(\kappa)}$, and every input $x \in \{0,1\}^{\mathsf{n}(\kappa)}$, we have that*

$$\Pr_{\tilde{\mathbf{C}} \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_\kappa)}[\tilde{\mathbf{C}}(x) = \mathbf{C}_\kappa(x)] = 1.$$

– Security: *For an adversary* A*, consider the following interactive game:*
  1. A *outputs (the description of) two circuits* $(\mathbf{C}_0, \mathbf{C}_1)$*,*
  2. A *receives an obfuscation* $\mathsf{iO}(\mathbf{C}_b)$ *of* $\mathbf{C}_b$ *(for uniformly chosen* $b \leftarrow \{0,1\}$*),*
  3. A *outputs a guess bit* b'*.*

  We say that A *is valid if it initially only outputs pairs* $(\mathbf{C}_0, \mathbf{C}_1)$ *such that* $\mathbf{C}_0$ *and* $\mathbf{C}_1$ *are functionally equivalent, and have the same size, input length, and output length. For security, we require that for every valid PPT (in* κ*)* A*, the advantage*

$$\mathit{Adv}^{\mathsf{iO}}_{\mathsf{A}}(\kappa) := \left| \Pr[b = b' \text{ in the above game with } \mathsf{A}] - 1/2 \right|$$

  is negligible, where the probability is taken over b*, the randomness of* iO*, and the random coins of* A*.*

Circular Correlation Robust Hash Function. We recall the definition of circular correlation robust (CCR) hash functions. For a hash function $\mathsf{H}_\kappa : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$, consider the following two oracles:

– The *random oracle* $\mathcal{R} : \{0,1\}^\kappa \times \mathbb{N} \times \{0,1\} \to \{0,1\}^\kappa$ represents a random function. Note that this oracle can be simulated efficiently by lazy sampling: For each new call to $\mathcal{R}$ with input (x, i, b) a κ-bit string is sampled uniformly at random and assigned as the oracle output; each time $\mathcal{R}$ is queried with a previously queried input, the same output is delivered.
– The *CCR oracle* $\mathcal{C}_{\mathsf{H},\Delta} : \{0,1\}^\kappa \times \mathbb{N} \times \{0,1\} \to \{0,1\}^\kappa$ represents a function parametrized by H_κ and a κ-bit value $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ sampled uniformly at random. For each input (x, i, b), this oracle outputs the value $\mathsf{H}_\kappa(x \oplus \Delta, i) \oplus b\Delta$.

Given these oracles, CCR secure hash functions are defined as given below.

Definition 4 (Circular Correlation Robust Hash Function). *Let* $\mathcal{H} = \{\mathsf{H}_\kappa : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$ *be an ensemble of hash functions. Let a 'legal' sequence of oracle queries, each of the form* (x, i, b)*, be one in which the same value* (x, i) *is never queried with different values of* b*. Then* $\mathcal{H}$ *is circular correlation robust (CCR) if for any PPT adversary* A*,*

$$\mathit{Adv}^{\mathit{CCR}}_{\mathsf{A}}(\kappa) = \frac{1}{2} \left| \Pr_{\Delta} \left[\mathsf{A}^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa) = 1 \right] - \Pr_{\mathcal{R}} \left[\mathsf{A}^{\mathcal{R}}(1^\kappa) = 1 \right] \right|$$

is negligible.

Garbling Schemes. Bellare et al. [14] abstract a garbling scheme as a cryptographic object containing four algorithms as given in Definition 5. For a Boolean circuit $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{q}, \{A, B, C, \mathsf{f}_g\}_{g \in [\mathsf{q}]})$, we denote the public information about $\mathbf{C}$ by $\phi(\mathbf{C})$, where ϕ is called a leakage function. In particular, $\phi_{\mathsf{circ}}(\mathbf{C}) = \mathbf{C}$ reveals the complete structure of the circuit, while $\phi_{\mathsf{topo}}(\mathbf{C}) = (\mathsf{n}, \mathsf{m}, \mathsf{q}, \{A, B, C\}_{g \in [\mathsf{q}]})$ reveals only the topology of the circuit, i.e. it does not leak the type of gates. Since the scope of this work is limited to schemes that garble gates in $\{\mathsf{AND}, \mathsf{XOR}\}$ and support Free-XOR, going forward, we use $\phi = \phi_{\mathsf{circ}}$.

Definition 5 (Garbling Scheme [14]). *A garbling scheme is a tuple* $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *of four polynomial-time algorithms with the following syntax:*

– $(\mathbf{F}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C})$*: on input a security parameter (in unary) and a circuit* $\mathbf{C}$*, returns a garbling* $\mathbf{F}$*, input encoding function* $\mathbf{e}$*, and output decoding function* $\mathbf{d}$*.*

– $\mathbf{X} = \mathsf{En}(\mathbf{e}, \mathbf{x})$*: returns the encoding* $\mathbf{X}$ *for function input* $\mathbf{x}$*.*
– $\mathbf{Y} = \mathsf{Ev}(\mathbf{F}, \mathbf{X})$*: returns the output labels* $\mathbf{Y}$ *by evaluating* $\mathbf{F}$ *on* $\mathbf{X}$*.*
– $\{\perp, \mathbf{y}\} = \mathsf{De}(\mathbf{Y}, \mathbf{d})$*: returns either the failure symbol* $\perp$ *or a value* $\mathbf{y} = f(\mathbf{x})$*.*

These algorithms must satisfy the following properties:

– Correctness*: For every circuit* $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ *and input* $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$*,*

$$\Pr[\mathbf{y} = \mathbf{C}(\mathbf{x}) : (\mathbf{F}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Gb}(\mathbf{C}),\ \mathbf{X} = \mathsf{En}(\mathbf{e}, \mathbf{x}),\ \mathbf{Y} = \mathsf{Ev}(\mathbf{F}, \mathbf{X}),\ \mathbf{y} = \mathsf{De}(\mathbf{d}, \mathbf{Y})] = 1$$

– Non-degeneracy*: For any pair* $\mathbf{C}_0, \mathbf{C}_1 : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ *with* q *gates,*

$$\{\mathbf{e}_0, \mathbf{d}_0\}_{(\mathbf{F}_0, \mathbf{e}_0, \mathbf{d}_0) \leftarrow \mathsf{Gb}(\mathbf{C}_0)} \equiv \{\mathbf{e}_1, \mathbf{d}_1\}_{(\mathbf{F}_1, \mathbf{e}_1, \mathbf{d}_1) \leftarrow \mathsf{Gb}(\mathbf{C}_1)}$$

Definition 6 (Selective Privacy by Simulation (PRIV-SIM)). *A garbling scheme* $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is PRIV-SIM-secure if there exists a PPT algorithm* Sim *such that for every two-stage PPT adversary* $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ *the following value is negligible:*

$$\Bigg| \Pr_{\substack{(\mathbf{C}, \mathbf{x}, \sigma) \leftarrow \mathsf{A}_0(1^\kappa) \\ (\mathbf{F}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{GS.Gb}(1^\kappa, \mathbf{C})}} [\mathsf{A}_1(\sigma, (\mathbf{F}, \mathsf{GS.En}(\mathbf{e}, \mathbf{x}), \mathbf{d})) = 1] - \Pr_{\substack{(\mathbf{C}, \mathbf{x}, \sigma) \leftarrow \mathsf{A}_0(1^\kappa) \\ (\mathbf{F}, \mathbf{X}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))}} [\mathsf{A}_1(\sigma, (\mathbf{F}, \mathbf{X}, \mathbf{d})) = 1] \Bigg|.$$

The Half Gates Garbling Scheme. The garbling algorithm takes as input a Boolean circuit composed of binary AND and XOR gates. Then starting from the input wires, labels are assigned to represent the 0 and 1 semantic values and then for each gate, the output wire labels are derived as a function of the input wire labels. Each gate is garbled differently depending on the gate functionality:

– This garbling scheme supports 'free-XOR', that is, XOR gates require no garbled representation [45]. This is done by sampling a global offset value Δ and ensuring that for all wires in the circuit, the bitwise-XOR of its wire labels is Δ. For the circuit input wires, the 0-label is sampled uniformly at random while the 1-label is set as the XOR of this and Δ. The output labels of an XOR gate are calculated by computing the bitwise-XOR of two input labels, one from each wire:

$$\mathsf{L}_A^0 \oplus \mathsf{L}_B^0 = \mathsf{L}_A^1 \oplus \mathsf{L}_B^1 = \mathsf{L}_C^0$$
$$\mathsf{L}_A^0 \oplus \mathsf{L}_B^1 = \mathsf{L}_A^1 \oplus \mathsf{L}_B^0 = \mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$$

– The garbling of each AND gate is represented using just 2 'ciphertexts' (2κ-bits) G_{AND}^0 and G_{AND}^1 of κ bits each. Intuitively, this needs to transfer enough information that the evaluation algorithm can derive an output label L_C (according to the gate functionality) as some linear combination of $(G_{\mathsf{AND}}^0, G_{\mathsf{AND}}^1, \mathsf{H}(\mathsf{L}_A), \mathsf{H}(\mathsf{L}_B), \mathsf{L}_A, \mathsf{L}_B)$ from one of the four cases in the AND-truth-table, all while being oblivious to which case is being evaluated. To implement this, first each input wire A and B is associated with a random 'permutation bit' p_a resp. p_b that allows to decouple the actions of the evaluation algorithm (that depend on the apparent value of the input labels) from the actual semantic value of the labels. The input labels are set such that $p_a = \mathsf{lsb}(\mathsf{L}_A^0)$ (resp. $p_b = \mathsf{lsb}(\mathsf{L}_B^0)$) and

$\mathsf{lsb}(\mathsf{L}_A^1) = \neg\mathsf{lsb}(\mathsf{L}_A^0)$ (resp. $\mathsf{lsb}(\mathsf{L}_B^1) = \neg\mathsf{lsb}(\mathsf{L}_B^0)$), which is guaranteed by choosing Δ with $\mathsf{lsb}(\Delta) = 1$. The output labels are then assigned as:

$$\mathsf{L}_C^0 = \mathsf{H}(\mathsf{L}_A^0 \oplus p_a\Delta) \oplus \mathsf{H}(\mathsf{L}_B^0 \oplus p_b\Delta) \oplus p_a p_b \Delta$$
$$\mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$$

The garbling must contain all the information required to evaluate, that is not derivable from L_A and L_B. It is generated as,

$$G_{\mathsf{AND}}^0 = \mathsf{H}(\mathsf{L}_A^0) \oplus \mathsf{H}(\mathsf{L}_A^1) \oplus p_b\Delta$$
$$G_{\mathsf{AND}}^1 = \mathsf{H}(\mathsf{L}_B^0) \oplus \mathsf{H}(\mathsf{L}_B^1) \oplus \mathsf{L}_A^0$$

During the evaluation, let $s_a = \mathsf{lsb}(\mathsf{L}_A)$ and $s_b = \mathsf{lsb}(\mathsf{L}_B)$ be the 'apparent values' of these wires – this is the actual wire value (known to the garbler) masked with the permutation bit. The AND gate is then evaluated as:

$$\mathsf{L}_C = \mathsf{H}(\mathsf{L}_A) \oplus \mathsf{H}(\mathsf{L}_B) \oplus s_a G_{\mathsf{AND}}^0 \oplus s_b(G_{\mathsf{AND}}^1 \oplus \mathsf{L}_A)$$

allowing the appropriate hashes within G_{AND}^0 and G_{AND}^1 to cancel, leaving us with the required output label.

3 Weak Circular Correlation Robust Hash Functions

In this section, we introduce a notion of security for hash function families that relaxes CCR security from Definition 4. We go on to show that this weaker security notion suffices to securely instantiate several garbling schemes, notably the Half Gates scheme [57], and the garbling schemes in [33, 52]. That is, these schemes are selectively private when being instantiated with a family of *weakly* CCR secure hash functions. We then show later in Section 4 that this primitive can be realized in the standard model (i.e., without idealizations such as random oracles).

Outsourcing Hash Function Access Into a 'Query Strategy'. Recall that the security notion of CCR requires that a PPT adversary A – given a hash function H and access to either a random oracle $\mathcal{R}$ or a CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ – cannot distinguish between either case with non-negligible advantage. Its querying strategy to the given oracle is unrestricted (as long as the sequence of queries is 'legal') and can depend adaptively on the information it receives. In contrast, we define the security notion of a 'weakly CCR' secure hash function *family* to be one in which the distinguishing adversary A does not have direct access to the query oracle. Instead, it creates an efficient deterministic oracle algorithm $\mathsf{QueryStrategy}^{(\cdot)}$ and submits this to a challenger $\mathcal{C}$. Syntactically, $\mathsf{QueryStrategy}^{(\cdot)}$ expects the security parameter 1^κ and some string r as input, as well as adaptive oracle access to a function $\mathcal{O}$ (that takes as input a bit-string x, a running number i that is automatically chosen and incremented upon every oracle query, and a bit b). The output of $\mathsf{QueryStrategy}^{\mathcal{O}}(1^\kappa; r)$ (for a concrete oracle $\mathcal{O}$) consists of a transcript of oracle queries (x_i, i, b_i) and responses $y_i := \mathcal{O}(x_i, i, b_i)$. In other words, the output of $\mathsf{QueryStrategy}^{\mathcal{O}}$ on input $(1^\kappa, r)$ is of the form

$$\mathcal{Q} = \{(x_i, i, b_i, y_i)\}_{i\in[Q]}.$$

Algorithm 1 Garbling Scheme GS^{H} for circuits with XOR and AND gates

procedure $\mathsf{Gb}(1^\kappa, \mathbf{C})$
  initialize $\mathbf{F} = []$, $\mathbf{e} = []$ and $\mathbf{d} = []$
  sample $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$ and set $\mathsf{L}_i^1 = \mathsf{L}_i^0 \oplus \Delta$
    set $\mathbf{e}[i] = \mathsf{L}_i^0$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$
    if $f_g = \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^0 = \mathsf{L}_A^0 \oplus \mathsf{L}_B^0$ and $\mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$
    else
      $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^0)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^0)$
      $G_g^0 = \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus \mathsf{H}(\mathsf{L}_A^0 \oplus \Delta, k_g^0) \oplus p_b\Delta$
      $G_g^1 = \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus \mathsf{H}(\mathsf{L}_B^0 \oplus \Delta, k_g^1) \oplus \mathsf{L}_A^0$
      $\mathsf{L}_C^0 = \mathsf{H}(\mathsf{L}_A^0 \oplus p_a\Delta, k_g^0) \oplus \mathsf{H}(\mathsf{L}_B^0 \oplus p_b\Delta, k_g^1) \oplus p_a p_b \Delta$
      $\mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$
      set $\mathbf{F}[g] = (G_g^0, G_g^1)$
    end if
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^0)$
  end for
  return $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$
end procedure

procedure $\mathsf{En}((\Delta, \mathbf{e}), \mathbf{x})$
  initialize $\mathbf{X} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathbf{X}[i] = \mathbf{e}[i] \oplus \mathbf{x}[i]\Delta$
  end for
  Return $\mathbf{X}$
end procedure

We require that Q, the total number of oracle queries in a run of $\mathsf{QueryStrategy}^{\mathcal{O}}$, only depends on κ, and A's random coins and the concrete choice of QueryStrategy (but not on, say, r or $\mathcal{O}$ that is input to QueryStrategy). Note that by definition (and in particular since i is automatically increased upon each query), $\mathcal{Q}$ cannot contain two entries of the form (x, i, b, y) and $(x', i, 1-b, y')$. Hence, this characterization of $\mathcal{Q}$ already satisfies the 'legality' notion for CCR secure hashing (Definition 4).

How Query Strategies Can Model Garbling. Query strategies as above can be used to express all the hash function queries that occur during a garbling process. More specifically, for garbling schemes like the Half Gates scheme (Algorithm 1-2) whose security depends solely on the properties of the underlying hash function H, given a circuit $\mathbf{C}$ and an input $\mathbf{x}$, we can define a specific querying strategy $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}$ that models a garbling of $\mathbf{C}$ and encoding of input $\mathbf{x}$ as follows. First, starting from the input $\mathbf{x}$, we term as *active* the value that each wire of the circuit would take on an evaluation of $\mathbf{C}(\mathbf{x})$. This means

Algorithm 2 Algorithms to Evaluate the Garbling

procedure $\mathsf{Ev}(\mathbf{F}, \mathbf{X})$
  initialize $\mathbf{Y} = []$
  for each gate $g \in [\mathsf{q}]$ in a topological order **do**
    $\mathsf{L}_A, \mathsf{L}_B \leftarrow$ active labels associated with the input wires of gate g
    if $f_g = \mathsf{XOR}$ **then**
      $\mathsf{L}_C = \mathsf{L}_A \oplus \mathsf{L}_B$
    else
      $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $s_a = \mathsf{lsb}(\mathsf{L}_A)$, $s_b = \mathsf{lsb}(\mathsf{L}_B)$
      $\mathsf{L}_C = \mathsf{H}(\mathsf{L}_A, k_g^0) \oplus \mathsf{H}(\mathsf{L}_B, k_g^1) \oplus s_a G_g^0 \oplus s_b(G_g^1 \oplus \mathsf{L}_A)$
    end if
    if C is a circuit output wire **then**
      $\mathbf{Y}[C] = \mathsf{L}_C$
    end if
  end for
  return $\mathbf{Y}$
end procedure

procedure $\mathsf{De}(\mathbf{Y}, \mathbf{d})$
  initialize $\mathbf{y} = []$
  for $j \in [\mathsf{m}]$ **do**
    $\mathbf{y}[j] = \mathbf{d}[j] \oplus \mathsf{lsb}(\mathbf{Y}[j])$
  end for
  return $\mathbf{y}$
end procedure

that in the garbling, for each circuit wire one label, that the evaluation algorithm derives, would be the *active label* while the other is termed as the *inactive label*. Next, we denote with $(\mathbf{F}, \mathbf{e}, \mathbf{d}) := \mathsf{Gb}^{\mathcal{O}}(\mathbf{C}; r)$ the computation of the garbling algorithm from Definition 5 with random coins r, re-arranged with all executions of H that use the inactive label as input being replaced by queries to $\mathcal{O}$. This results in all the primitive queries (to either H or $\mathcal{O}$) being written as a function of the active wire labels only. This is followed by setting $\mathbf{X} = \mathsf{En}(\mathbf{e}, \mathbf{x})$ as the set of active input wire labels. The arising transcript $\mathcal{Q} = \mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{X}}(1^\kappa; r)$ would be the sequence of queries and responses from the oracle that were made by the garbling algorithm. We discuss this in more detail with an example in Section 3.1.

The Challenger. For security parameter κ, a challenger, on receiving a query strategy $\mathsf{QueryStrategy}^{(\cdot)}$ as above from A, first samples a hash function $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$ uniformly at random from the function family $\mathcal{H}_\kappa$. It then samples a bit $c \in \{0,1\}$ according to which the oracle $\mathcal{O}$ is set. That is, if $c = 0$, $\mathcal{O} = \mathcal{R}$ (for an independent random oracle $\mathcal{R}$), and $\mathcal{O}(x, i, b) = \mathcal{C}_{\mathsf{H},\Delta}(x, i, b) = \mathsf{H}(x \oplus \Delta, i) \oplus b\Delta$ otherwise. Finally, the challenger samples random coins r, executes $\mathcal{Q}_c = \mathsf{QueryStrategy}^{\mathcal{O}}(1^\kappa; r)$ and hands $(\mathcal{Q}_c, \mathsf{H})$ to the adversary A. We say that the ensemble of hash function families $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa \in \mathbb{N}}$ is weakly CCR secure if there exists no PPT adversary A that can guess the bit c sampled by the challenger with non-negligible advantage. This notion is formalized in Definition 7.

Definition 7 (Weak Circular Correlation Robust (wCCR) Hash Functions). *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa \in \mathbb{N}}$ be an ensemble where each $\mathcal{H}_\kappa$ is a family of hash functions $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$. For $\mathsf{H} \in \mathcal{H}_\kappa$ and $\Delta \in \{0,1\}^{\kappa-1}||1$, we define oracles $\mathcal{C}_{\mathsf{H},\Delta}$ and $\mathcal{R}$ as*

– $\mathcal{C}_{\mathsf{H},\Delta}(x,i,b) = \mathsf{H}(x \oplus \Delta, i) \oplus b \cdot \Delta$ *(for* $x \in \{0,1\}^\kappa, i \in \mathbb{N}, b \in \{0,1\}$*),*
– $\mathcal{R}$ *is a random function with domain* $\{0,1\}^\kappa \times \mathbb{N} \times \{0,1\}$ *and range* $\{0,1\}^\kappa$*.*

For a polynomial $Q(\cdot)$*, we say* $\mathcal{H}$ *is a* Q-weak circular correlation robust *(*Q*-wCCR) hash function family if for any PPT adversary* $(\mathsf{A}_0, \mathsf{A}_1)$*, where* $\mathsf{A}_0(1^\kappa)$ *outputs* $\mathsf{QueryStrategy}^{(\cdot)}$ *that makes at most* $Q(\kappa)$ *oracle queries, the following value is negligible:*

$$\left| \Pr_{\substack{(\mathsf{QueryStrategy}^{(\cdot)},\sigma)\leftarrow\mathsf{A}_0(1^\kappa) \\ \mathsf{H}\leftarrow\mathcal{H}_\kappa, r\leftarrow\{0,1\}^{poly(\kappa)} \\ \Delta\leftarrow\{0,1\}^{\kappa-1}||1}} [\mathsf{A}_1(\sigma, \mathsf{QueryStrategy}^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa; r), \mathsf{H}) = 1] - \right.$$

$$\left. \Pr_{\substack{(\mathsf{QueryStrategy}^{(\cdot)},\sigma)\leftarrow\mathsf{A}_0(1^\kappa) \\ \mathsf{H}\leftarrow\mathcal{H}_\kappa, r\leftarrow\{0,1\}^{poly(\kappa)}, \mathcal{R}}} [\mathsf{A}_1(\sigma, \mathsf{QueryStrategy}^{\mathcal{R}}(1^\kappa; r), \mathsf{H}) = 1] \right|.$$

If $\mathcal{H}$ *is* Q*-wCCR for every polynomial* $Q(\cdot)$*, then we say that* $\mathcal{H}$ *is* weak circular correlation robust *(wCCR).*

In the definition, note that we sample Δ under the constraint that its last bit is always 1. Looking ahead, this is required for garbling applications to ensure that the apparent values of two labels of the same wire are distinct. Several remarks on our definition are in order.

Remark 1 (Comparison with the original definition [24]). The notion of CCR (Definition 4) first *fixes* a hash function H and then considers a PPT distinguisher that is given H and access to an oracle $\mathcal{O}$, which is either $\mathcal{C}_{\mathsf{H},\Delta}$ or $\mathcal{R}$. It is tasked with distinguishing between the two worlds, given adaptive access to $\mathcal{O}$ under the constraint that for any $x \in \{0,1\}^\kappa$ and $i \in \mathbb{N}$, $\mathcal{O}$ can be queried only on one of $(x,i,0)$ and $(x,i,1)$. The definition of wCCR is strictly weaker than CCR in two aspects. First, the query strategy of a wCCR distinguisher is more restricted, in that two inputs $x_0, x_1 \in \{0,1\}^\kappa$ cannot be queried to $\mathcal{O}$ for the same $i \in \mathbb{N}$. This renders the set of allowed query sets as being a strict subset of those that are allowed in the definition of CCR. Secondly, the degree of adaptivity in queries for wCCR is highly restricted as the distinguisher no longer has access to $\mathcal{O}$. Instead, it is required to submit its query strategy to a challenger at the beginning of the security game. Although the strategy itself can be an arbitrary PPT algorithm yielding a legal query set, the randomness used in this algorithm, if any, is sampled by the challenger. Furthermore, the query strategy has to be selected *before* the hash function H is known, hence queries cannot be made adaptively *depending* on H.

Remark 2 (Comparison with "Selective" CCR). In a selective variant of circular correlation robustness, again a fixed hash function H is considered, but in contrast to adaptive CCR the distinguisher first declares a set of queries that it will make to the oracle, under the restriction that for any $x \in \{0,1\}^\kappa$ and $i \in \mathbb{N}$ only one of $(x,i,0)$ and $(x,i,1)$ can be queried. It then receives all responses and can make no additional queries. This notion of CCR is insufficient for instantiating the garbling scheme in [57] where subsequent oracle queries need to be chosen as a function of preceding query responses. It is also incomparable to the wCCR notion (Definition 7) that we consider. On the one hand, wCCR allows for more restricted query sets where there cannot be two inputs x_0, x_1 to the oracle for the same $i \in \mathbb{N}$. This renders the set of allowed query sets as being a strict subset of those that are allowed for selective CCR. On the other hand, wCCR allows for more flexibility in the choice of queries since the query strategy need not select all queries independently of the responses of others.

Remark 3 (Comparison with CCR for Naturally Derived Keys [57]). Zahur et al. [57] define a notion of CCR where they restrict queries to 'naturally derived keys'. This is a definition tailored for application in their garbling scheme where the oracle queries x need to be 'naturally derived', in addition to satisfying that for any (x, i) only one of $(x, i, 0)$ and $(x, i, 1)$ can be queried. A naturally derived query is one that is either (1) sampled at random in $\{0,1\}^\kappa$; (2) the output of an oracle or hash query with a naturally derived key; or (3) a linear combination of the same. This notion is incomparable to wCCR as, on the one hand, this definition heavily restricts the permissible query strategies. On the other hand, the rules for allowed query sets in wCCR are more strict than those for the notion above. Furthermore, in wCCR the distinguisher does not have oracle access, and the randomness within the query strategy is sampled by the challenger.

Remark 4 (Comparison with RTCCR security [52]). Rosulek et al. [52] define a variant of CCR that they call *randomized tweakable circular correlation robust* security. This is formally defined in Section 3.1 (Definition 9) where we discuss its applicability to the garbling scheme in [52]. This security notion considers a two-part adversary: one with oracle access to either the CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ or the random oracle $\mathcal{R}$; and the other that receives all this information (and no further oracle access) and the description of the hash function itself, and needs to distinguish between the two oracles. Like weak CCR security, this notion considers a family of hash functions and the deciding adversary receives the hash function but without direct access to the oracles. However, RTCCR is more general in that hash function outputs can be a different length than its input. As a result, the CCR oracle also allows for different linear combinations of the bits of Δ to be applied to the hash output.

In Section 3.1, we go on to show that the definition of wCCR (Definition 7) suffices as the underlying assumption for proving that the garbling scheme in [57] is selectively secure.

3.1 Application to Secure Garbling

In this section, we show that the three garbling schemes presented in [33,52,57] are selectively secure when the underlying hash function H used in the garbling is sampled randomly from a wCCR-secure hash function family.

Selective Security of the Half Gates Garbling Scheme [57]. Zahur et al. in [57] present a garbling scheme GS^{H} that we detail in Algorithms 1 and 2. This scheme is proven selectively secure (PRIV-SIM) based on CCR security (Definition 4) of the underlying hash function H in [57].

Security under wCCR secure hash function families. For a family of wCCR secure hash functions $\mathcal{H}$, let $\mathsf{GS}^{\mathcal{H}}$ denote a garbling scheme where the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^\kappa; \mathbf{C})$ first samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ and then operates as in $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$ given in Algorithm 1. Finally, this garbling algorithm outputs $((\mathbf{F}, \mathsf{H}), (\Delta, \mathbf{e}), \mathbf{d})$, where $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$ are the outputs of $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$. The evaluation algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Ev}((\mathbf{F}, \mathsf{H}), \mathbf{X})$ works by using H to execute $\mathsf{GS}^{\mathsf{H}}.\mathsf{Ev}$ as given in Algorithm 2. All other algorithms in $\mathsf{GS}^{\mathcal{H}}$ operate exactly as in GS^{H}. Formally, we prove the following theorem:

Theorem 5. *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa \in \mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 1*

and 2 where, for security parameter κ, in the beginning of Gb *a hash function* H *is sampled uniformly at random from* $\mathcal{H}_\kappa$ *and output together with the garbled circuit* $\mathbf{F}$*. Then* $\mathsf{GS}^{\mathcal{H}}$ *satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

We refer the reader to Appendix B for a full proof of Theorem 5.

Proof Outline. The proof for this statement is adapted from [57] and considers the same description of the simulator and list of hybrids. The simulator for the garbling is a PPT algorithm $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))$ that needs to output the selective privacy challenge tuple created only using the circuit and the function output. Let $\mathbf{X}$ denote the set of *active input labels* for the garbling $\mathbf{F}, \mathbf{d}$. For each wire in the circuit $\mathbf{C}$, we refer to the label derived from evaluating $\mathbf{F}$ using $\mathbf{X}$ as the *active label*. In the simulation, for each wire, the label representing the 0-value is assigned as the active label. Then, note that for each AND gate, the garbling algorithm can be re-written as:

$$\begin{aligned}
k_g^0 &= 2g - 1, k_g^1 = 2g, p_a = \mathsf{lsb}(\mathsf{L}_A^0), p_b = \mathsf{lsb}(\mathsf{L}_B^0)\\
G_g^0 &= \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_A^0, k_g^0, p_b)\\
G_g^1 &= \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_B^0, k_g^1, 0) \oplus \mathsf{L}_A^0\\
\mathsf{L}_C^0 &= \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A^0)
\end{aligned}$$

in terms of the active labels and the weak CCR hash oracle $\mathcal{C}_{\mathsf{H},\Delta}(x, i, b) = \mathsf{H}(x \oplus \Delta, i) \oplus b\Delta$ only. Further, note that throughout the garbling algorithm, the indices $k_g^b \in \mathbb{N}$ act as a counter for the oracle queries made, rendering the set of garbling oracle queries and responses as a legal query strategy satisfying weak CCR (Definition 7).

Given this observation, the simulator Sim works as follows: first, it samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$. Then, for each input wire, an active label is sampled as in the real garbling. Then the garbling algorithm is executed with each instance of the weak CCR oracle output being replaced by a κ-bit value sampled uniformly at random. This process also derives the active wire label for all other wires in the circuit, including the output wires. Finally, the output decoding information is set such that the active output labels map to the function output as required.

Within the list of hybrid experiments to prove that the real and simulated distributions are computationally indistinguishable, all but one set of adjacent hybrids are identically distributed. Within the pair of adjacent hybrid distributions that are not identically distributed, in one hybrid experiment, the garbled circuit is generated in such a way that the oracle calls described above are made to the random oracle $\mathcal{R}$, in the other hybrid experiment, these are calls to the oracle $\mathcal{C}_{\mathsf{H},\Delta}$. Both experiments are otherwise identical.

This pair of adjacent hybrids is shown as computationally indistinguishable by reduction to the weak CCR property of the hash function family $\mathcal{H}$. Here a PPT adversary $\mathcal{A}_{\mathsf{wCCR}}$ for the weak CCR game (Definition 7) accepts from the distinguisher $\mathcal{A}_{\mathsf{GS}}$ for the hybrids, a circuit $\mathbf{C}$ with $\mathsf{q}_{\mathsf{AND}}$ AND gates, and input $\mathbf{x}$. $\mathcal{A}_{\mathsf{wCCR}}$ designs a PPT algorithm $\mathsf{QueryStrategy}$ with the following interface:

$$\mathcal{Q} = \{(x_j, j, b_j, y_j)\}_{j \in [2\mathsf{q}_{\mathsf{AND}}]} = \mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}(1^\kappa; r).$$

Letting $\mathsf{n} = |\mathbf{x}|$, this algorithm takes as input random coins $r \leftarrow \{0,1\}^{\mathsf{n}\kappa}$ from the challenger, that defines the active input labels corresponding to $\mathbf{x}$. This is detailed in Algorithm 3 with the lines indicating queries that populate $\mathcal{Q}$ marked in blue. This is a randomized algorithm

that is given to the challenger $\mathcal{C}$ for the weak CCR security game. The challenger $\mathcal{C}$ accepts this $\mathsf{QueryStrategy}^{(\cdot)}$, operates as implicit in Definition 7 and gives a tuple $(\mathcal{Q}, \mathsf{H})$ to $\mathcal{A}_{\mathsf{wCCR}}$. The adversary $\mathcal{A}_{\mathsf{wCCR}}$ can now execute the garbling using the active labels indicated in $\mathcal{Q}$ and using the items in $\mathcal{Q}$ wherever oracle queries are required. The output of this is a tuple $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ given to $\mathcal{A}_{\mathsf{GS}}$. Finally, $\mathcal{A}_{\mathsf{GS}}$ outputs a bit c' indicating its deduction for which hybrid distribution the tuple belongs to, and $\mathcal{A}_{\mathsf{wCCR}}$ outputs the same bit c' to complete the reduction. In this reduction, the advantage of $\mathcal{A}_{\mathsf{wCCR}}$ would be the same as that of $\mathcal{A}_{\mathsf{GS}}$, which is non-negligible, arriving at a contradiction since H was sampled from a family of weak CCR secure hash functions. This concludes the proof.

Algorithm 3 Query Strategy $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}$ for $\mathsf{GS}^{\mathcal{H}}$ adapted from [57]

procedure $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}(1^\kappa; r \leftarrow \{0,1\}^{n\kappa})$
  initialize $\mathcal{Q} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathsf{L}_i = r[(i-1)\kappa + 1 : i\kappa] \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$
    if $f_g = \mathsf{XOR}$ **then**
      set $\mathsf{L}_C = \mathsf{L}_A \oplus \mathsf{L}_B$
    else
      $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $p_a = \mathsf{lsb}(\mathsf{L}_A)$, $p_b = \mathsf{lsb}(\mathsf{L}_B)$
      query $M_A = \mathcal{O}(\mathsf{L}_A, k_g^0, p_b)$ and $M_B = \mathcal{O}(\mathsf{L}_B, k_g^1, 0)$
      set $\mathcal{Q}[k_g^0] = (\mathsf{L}_A, k_g^0, p_b, M_A)$ and $\mathcal{Q}[k_g^1] = (\mathsf{L}_B, k_g^1, 0, M_B)$
      $G_g^0 = \mathsf{H}(\mathsf{L}_A, k_g^0) \oplus M_A$
      $G_g^1 = \mathsf{H}(\mathsf{L}_B, k_g^1) \oplus M_B \oplus \mathsf{L}_A$
      $\mathsf{L}_C = \mathsf{H}(\mathsf{L}_A, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A)$
    end if
  end for
  return $\mathcal{Q}$
end procedure

Selective Security of the Three Halves Garbling Scheme [52]. Rosulek et al. in [52] present a garbling scheme GS^{H} that we detail in Algorithms 7 and 8 in Appendix C. This is a non-linear garbling scheme for Boolean circuits with AND and XOR gates that supports free-XOR and can garble AND gates using $1.5\kappa + 5$ bits (opposed to 2κ bits as in Half Gates), where κ is the computational security parameter. Their scheme is proven selectively secure (PRIV-SIM) when the underlying hash function H is RTCCR secure (randomized tweakable circular correlation robust – Definition 9).

Security under wCCR secure hash function families. The original garbling scheme in [52] contains a garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$, where $\mathcal{H}$ is an RTCCR hash function family. The

garbling algorithm internally samples a hash function (with $\frac{\kappa}{2}$-bit range) from this family and proceeds as given in $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$ indicated in Algorithm 7. In our case, for a family of wCCR secure hash functions $\mathcal{H}$, let $\mathsf{GS}^{\mathcal{H}}$ denote a garbling scheme where the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^{\kappa}; \mathbf{C})$ first samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_{\kappa}$ of the form $\mathsf{H} : \{0,1\}^{\kappa} \times \mathbb{N} \rightarrow \{0,1\}^{\kappa}$. It then operates as in $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$ given in Algorithm 7, but with the following exceptions:

- Algorithm 7 represents the garbling of an AND gate as a system of linear equations:

$$\mathbf{V}\begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = (\mathbf{R} \oplus \mathbf{T})\begin{bmatrix}\mathsf{L}_A^0\\ \mathsf{L}_B^0\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$$

  Here $\mathbf{H}$ is a vector containing all possible hash evaluations on input labels and $\mathbf{M}$ is an indicator matrix that describes which elements of $\mathbf{H}$ are used to compute each (half of) the output label in the truth-table of the AND gate. Further, $(\mathbf{R} \oplus \mathbf{T})$ is a similar indicator matrix that describes which (half of the) plaintext label is used (XOR-ed to the hash outputs) in the evaluation. We refer the reader to Appendix C for further details of the scheme.
- In our modified scheme, whenever calls to H (indicated by $\mathbf{M}$) are of the form $\mathsf{H}(\mathsf{L}, i)$ for $i \in \mathbb{N}$ and $\mathsf{L} \in \{\mathsf{L}_A^0, \mathsf{L}_A^1, \mathsf{L}_B^0, \mathsf{L}_B^1\}$, the $\frac{\kappa}{2}$-bit *prefix* of the response is used in the garbling. If the garbling indicates (within $(\mathbf{R} \oplus \mathbf{T})$) that (part of) Δ be XOR-ed to this value, then the $\frac{\kappa}{2}$-bit *prefix* of Δ is bitwise XOR-ed to this hash output.
- Whenever calls to H (indicated by $\mathbf{M}$) are of the form $\mathsf{H}(\mathsf{L} \oplus \mathsf{L}', i)$ for $i \in \mathbb{N}$ and $(\mathsf{L} \oplus \mathsf{L}') \in \{(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0), (\mathsf{L}_A^0 \oplus \mathsf{L}_B^1)\}$, the $\frac{\kappa}{2}$-bit *suffix* of the response is used in the garbling. If the garbling indicates (within $(\mathbf{R} \oplus \mathbf{T})$) that (part of) Δ be XOR-ed to this value, then the $\frac{\kappa}{2}$-bit *suffix* of Δ is bitwise XOR-ed to this hash output.

Finally, this garbling algorithm outputs $((\mathbf{F}, \mathsf{H}), (\Delta, \mathbf{e}), \mathbf{d})$, where $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$ are the outputs of $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$. The evaluation algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Ev}((\mathbf{F}, \mathsf{H}), \mathbf{X})$ works by using H to execute $\mathsf{GS}^{\mathsf{H}}.\mathsf{Ev}$ as given in Algorithm 8, where the outputs of the hash function H are converted to $\frac{\kappa}{2}$ bits as described above. All other algorithms in $\mathsf{GS}^{\mathcal{H}}$ operate exactly as in GS^{H}. Formally, we prove the following theorem:

Theorem 6. *Let $\mathcal{H} = \{\mathcal{H}_{\kappa}\}_{\kappa \in \mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 7 and 8 where, for security parameter κ, in the beginning of Gb a hash function H is sampled uniformly at random from $\mathcal{H}_{\kappa}$ and output together with the garbled circuit $\mathbf{F}$. Then $\mathsf{GS}^{\mathcal{H}}$ satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

We refer the reader to Appendix C for a full description of the garbling scheme [52] adapted to $\mathsf{GS}^{\mathcal{H}}$ and proof of Theorem 6.

Selective Security of the Arithmetic Garbling Scheme in [33]. The work in [33] presents a garbling scheme GS^{H} for circuits that are a $\mathbb{Z}_m$ arithmetic generalization of Boolean garbled circuits where addition is free and each multiplication gate has size $O(\ell\kappa)$ bits, where ℓ is the size of the bit representation of each field element in $\mathbb{Z}_m$. Their scheme is proven selectively secure (PRIV-SIM) when the underlying hash function H is CCR secure (Definition 4). [33] presents a formal proof for a version of their scheme where hash functions have the interface $\mathsf{H} : \{0,1\}^{\kappa} \times \mathbb{N} \rightarrow \{0,1\}^{\kappa}$ (as opposed to the more general form $\mathsf{H} : \mathbb{Z}_{2^k}^{\kappa} \times \mathbb{N} \rightarrow \mathbb{Z}_{2^k}^{\kappa}$ for which the complete scheme in [33] is defined). They then discuss how generalizing CCR security to hash functions with more general domains allows proving the garbling scheme, in its complete generality, selectively secure.

Security under wCCR secure hash function families. For a family of wCCR secure hash functions $\mathcal{H}$, let $\mathsf{GS}^{\mathcal{H}}$ denote a garbling scheme where the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^\kappa; \mathbf{C})$ first samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ of the form $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$ and then operates as in $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$ given in Algorithm 14. Finally, this garbling algorithm outputs $((\mathbf{F}, \mathsf{H}), (\Delta, \mathbf{e}), \mathbf{d})$, where $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$ are the outputs of $\mathsf{GS}^{\mathsf{H}}.\mathsf{Gb}$. The evaluation algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Ev}((\mathbf{F}, \mathsf{H}), \mathbf{X})$ works by using H to execute $\mathsf{GS}^{\mathsf{H}}.\mathsf{Ev}$ as given in Algorithm 15. All other algorithms in $\mathsf{GS}^{\mathcal{H}}$ operate exactly as in GS^{H}. Formally, we prove the following theorem:

Theorem 7. *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa \in \mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 14 and 15 where, for security parameter κ, in the beginning of Gb a hash function H is sampled uniformly at random from $\mathcal{H}_\kappa$ and output together with the garbled circuit $\mathbf{F}$. Then $\mathsf{GS}^{\mathcal{H}}$ satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

We refer the reader to Appendix D for a full description of the garbling scheme [33], adapted to using hash functions with interface $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$ (Algorithms 14 and 15) and a proof of Theorem 7. We refer the reader to Remark 5 for an informal discussion on how a generalization of weak CCR security, that applies to hash function families with more general domains, can realize the general garbling scheme in [33] and how this notion can be realized in the plain model.

4 Realizing Weak CCR

In this section, we present a family of wCCR hash functions. Our construction is simple: the hash function is the obfuscation of a PPRF whose key is sampled at random. A more formal description is given below, followed by the proof that it satisfies wCCR.

Construction 1 *Let iO be an indistinguishability obfuscator (Definition 3) and $F = (\mathsf{Key}_F, \mathsf{Puncture}_F, \mathsf{Eval}_F)$ be a puncturable PRF (Definition 2). Our candidate construction of wCCR hash function family ensemble is defined as*

$$\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa \in \mathbb{N}} := \{\{\mathsf{iO}(F_k(\cdot); r)\}_{k \in supp(\mathsf{Key}_F(1^\kappa)), r \in \{0,1\}^{poly(\kappa)}}\}_{\kappa \in \mathbb{N}},$$

where $supp(\cdot)$ denotes the support of a distribution. To sample a hash function H from this family, first sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$ and then output $\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot))$, where $\kappa \in \mathbb{N}$ is the security parameter.

Theorem 8. *Let iO be an indistinguishability obfuscator (Definition 3) and $F = (\mathsf{Key}_F, \mathsf{Puncture}_F, \mathsf{Eval}_F)$ be a puncturable PRF (Definition 2). Then Construction 1 is a Q-weak CCR hash function family for any fixed polynomial Q (Definition 7).*

As a corollary to Theorems 5 to 8, we obtain our main results:

Corollary 1. *Assuming iO and OWFs exist, the Half Gates construction (Algorithms 1 and 2), the Three Halves construction (Algorithms 7 and 8), and switch systems construction (Algorithms 14 and 15) instantiated with the hash function family from Construction 1 (and modified to sample a hash function from this family during the garbling procedure) are PRIV-SIM-secure.*

Notation. Before proving Theorem 8, we need to fix some notation. We start by defining a few auxiliary functions.

- We use δ_Δ to denote the point function at Δ, i.e.:

$$\delta_\Delta(x) := \begin{cases} 1 & \text{if } x = \Delta \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

- For an iOWF f and a hard-core predicate for f, h, we define

$$\begin{aligned} \pi(x) &:= (f(x), h(x)) \\ \overline{\pi}(x) &:= (f(x), \overline{h(x)}) = (f(x), 1 - h(x)). \end{aligned} \tag{7}$$

  Note that π and $\overline{\pi}$ are themselves injective.
- It follows from the above notation that

$$\begin{aligned} \delta_{\pi(\Delta)} \circ \pi(x) &= \begin{cases} 1 & \text{if } \pi(x) = (f(\Delta), h(\Delta)) \\ 0 & \text{otherwise} \end{cases} \\ \delta_{\overline{\pi}(\Delta)} \circ \pi(x) &= \begin{cases} 1 & \text{if } \pi(x) = (f(\Delta), 1 - h(\Delta)) \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

  Although $\delta_{\overline{\pi}(\Delta)} \circ \pi(x) = 0$ for all x, the definition will be useful since (for random x)

$$\pi(x) = (f(x), h(x)) \approx_c (f(x), 1 - h(x)) = \overline{\pi}(x)$$

  by the security of hard-core predicate. We will exploit this property in one of the steps of our proof.
- Since $\mathsf{QueryStrategy}$ makes its queries iteratively, we abuse notation and use

$$\mathsf{QueryStrategy}(1^\kappa; r, y_1, \ldots, y_{i-1})$$

  to denote (x_i, b_i), the i-th query made by $\mathsf{QueryStrategy}$ when it is initiated with random coins r, and its first $i-1$ queries were answered by $y_1, \cdots, y_{i-1}$, respectively. In particular, $\mathsf{QueryStrategy}(1^\kappa; r)$ denotes the first query (x_1, b_1). Looking ahead, we will abuse notation and use $\mathsf{QueryStrategy}(1^\kappa; r, y_1, \cdots, y_Q)$ to refer to $\mathsf{QueryStrategy}(1^\kappa; r)$.

Proof (of Theorem 8). Recall that our objective is to show indistinguishability of the 'correlated' and 'random' distributions, i.e.:

$$\begin{aligned} &\left(\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot)), \mathcal{Q} = ((x_i, i, b_i, y_i))_{i \in [Q]} \leftarrow \mathsf{QueryStrategy}^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa; r)\right) \\ \overset{c}{\approx} &\left(\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot)), \mathcal{Q} = ((x_i, i, b_i, y_i^*))_{i \in [Q]} \leftarrow \mathsf{QueryStrategy}^{\mathcal{R}}(1^\kappa; r)\right). \end{aligned} \tag{9}$$

Here, $\mathsf{QueryStrategy}^{(\cdot)}$ is a query strategy output by the wCCR adversary which was invoked on some security parameter κ, $k \leftarrow \mathsf{Key}_F(1^\kappa)$ and r is randomly-sampled coins for $\mathsf{QueryStrategy}$. We proceed via a hybrid argument. There are eight main hybrids, denoted $\mathbf{Hybrid}_0, \cdots, \mathbf{Hybrid}_7$, where

- the extreme hybrids correspond to the correlated and random distributions from Equation (9), respectively; and

– $\mathbf{Hybrid}_1$ corresponds to the 'intermediate' distribution we discussed in Section 1.2.

Recall that in the intermediate distribution, the query answers have been switched to random, but the hash function now contains a hard-coded list that keeps track of the programming. Additionally, in order to prove indistinguishability of $\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_1$, we introduce a sequence of $4(Q+1)$ sub-hybrids where Q is the total number of queries in an execution of $\mathsf{QueryStrategy}$. These are denoted by

$$\begin{aligned}\mathbf{Hybrid}_0 = \mathbf{Hybrid}_{0,0}, \cdots, \mathbf{Hybrid}_{0,3}, \mathbf{Hybrid}_{1,0}, \cdots \\ \cdots \mathbf{Hybrid}_{Q-1,3}, \mathbf{Hybrid}_{Q,0}, \cdots, \mathbf{Hybrid}_{Q,3} = \mathbf{Hybrid}_1.\end{aligned} \tag{10}$$

We now describe all the hybrids one by one (the diff from previous hybrids is highlighted in red), along with an informal argument for why the current step is indistinguishable. A formal proof can be found in Appendix E.

– $\mathbf{Hybrid}_0$ is the correlated distribution. Therefore we use H as in Construction 1, and the sequence of queries is then derived by running $\mathsf{QueryStrategy}$ instantiated with oracle $\mathcal{C}_{\mathsf{H},\Delta}$.

$\mathbf{Hybrid}_0(1^\kappa, \mathsf{QueryStrategy})$

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$ and $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$
2. Generate $\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot))$
3. For each $i \in [1, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1, \ldots, y_{i-1})$
   (b) $y_i := \mathcal{C}_{\mathsf{H},\Delta}(x_i, i, b_i) = \mathsf{H}(x_i \oplus \Delta, i) \oplus b_i\Delta$
4. Output $\left(\mathsf{H}, ((x_i, i, b_i, y_i))_{i\in[Q]}\right)$

– Recall that $\mathbf{Hybrid}_1$ corresponds to the intermediate hybrid from Section 1.2. Therefore, H is modified to H^*, which is the obfuscation of a program F^* that is similar to F_k, but has certain input-output pairs related to $\mathsf{QueryStrategy}$'s queries *programmed* in it At the same time, all the answers, y_i^*, have been switched to random. This switch will be carried out iteratively, one answer at a time, using the sequence of sub-hybrids from Equation (10), which we define later. The indistinguishability of $\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_1$ is proved using properties of iO and PPRF (Lemma 1 in Appendix E).

$\mathbf{Hybrid}_1(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$ and $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$, and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i\Delta & \text{if } x = x_i \oplus \Delta \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

– The only difference between $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_1$ is the way in which F^* carries out the programming. To be specific, on input (x, i) instead of checking if 'x equals $x_i \oplus \Delta$' (where x_i is hard-wired), F^* now checks if '$x \oplus x_i$ equals Δ'. Looking ahead, the point of this syntactic change is isolate Δ so that the the aforementioned check can be implemented using a point function. Since the program F^* in both $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ is functionally equivalent, we can use iO security to argue that the two distributions are indistinguishable.

$\mathbf{Hybrid}_2(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i(x \oplus x_i) & \text{if } x \oplus x_i = \Delta \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

– As already alluded to, the only change in $\mathbf{Hybrid}_3$ is that the 'if' check in F^* is now implemented using the point function δ_Δ from Equation (6). Again, the change is merely syntactical and functional equivalence of circuits in $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ implies their indistinguishability via iO security.

$\mathbf{Hybrid}_3(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i(x \oplus x_i) & \text{if } \delta_\Delta(x \oplus x_i) = 1 \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

– In $\mathbf{Hybrid}_4$, instead of carrying out the '$x \oplus x_i$ equals Δ' check directly using δ_Δ, it is carried out *indirectly* by applying the function π from Equation (8) first. Since π is injective, the outcome of these direct and indirect checks is the same.[10] Thus, by security of iO, $\mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$ are indistinguishable. At this point, notice that the only dependence of F^* on Δ is to implement $\delta_{\pi(\Delta)}$ and – more importantly – knowledge of $\pi(\Delta)$ suffices to this purpose. This will be crucial to invoking the hard-core security of π in the next step.

[10] In Section 1.2, the indirect check was carried out using an iPRG. There we relied on injectivity of the iPRG instead.

Hybrid$_4(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program:

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i(x \oplus x_i) & \text{if } \delta_{\pi(\Delta)} \circ \pi(x \oplus x_i) = 1 \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

– In **Hybrid**$_5$, we carry out a "computational diagonalization" to replace the indirect 'if' check in F^* from **Hybrid**$_4$ with another check, one that *never* passes.[11] To this end, we replace the point function $\delta_{\pi(\Delta)}$ with the complementary point function $\delta_{\overline{\pi}(\Delta)}$. This switch is indistinguishable thanks to security of hard-core predicates, and it is crucial that knowledge of $\pi(\Delta)$ suffices to generate both hybrids.

Hybrid$_5(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i(x \oplus x_i) & \text{if } \delta_{\overline{\pi}(\Delta)} \circ \pi(x \oplus x_i) = 1 \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

– Since the indirect 'if' check in F^* from **Hybrid**$_5$ never passes, it is possible to ignore that branch of the program altogether, and the resulting program is simply the PPRF F_k. Therefore, in **Hybrid**$_6$ we can revert to the hash function from Construction 1, and this is indistinguishable thanks to iO security. Note that the resulting distribution is independent of Δ.

Hybrid$_6(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \ldots, y_Q^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [1, Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot))$
4. Output $\left(\mathsf{H}, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

[11] In Section 1.2, this was accomplished by switching the indirect check to 'if $G(x \oplus x_i) = R$', for a random element R in G's co-domain. With overwhelming probability, R lies outside of G's image, and therefore the check is never triggered. Indistinguishability follows by G's pseudo-randomness.

– Finally, in $\mathbf{Hybrid}_7$, we replace the y_i^* values with the output of a random oracle $\mathcal{R}$. Since repeat queries are not allowed in $\mathsf{QueryStrategy}$ (via tweaks), the distributions in $\mathbf{Hybrid}_6$ and $\mathbf{Hybrid}_7$ are identical. Moreover, $\mathbf{Hybrid}_7$ is precisely the random distribution from Equation (9) we were aiming for.

$\mathbf{Hybrid}_7(1^\kappa, \mathsf{QueryStrategy})$:

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$ and $r \leftarrow \{0,1\}^{poly(\kappa)}$
2. Generate $\mathsf{H}(\cdot) \leftarrow \mathsf{iO}(F_k(\cdot))$
3. For each $i \in [1, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1, \ldots, y_{i-1})$
   (b) $y_i := \mathcal{R}(x_i, i, b_i)$
4. Output $\left(\mathsf{H}, ((x_i, i, b_i, y_i^*))_{i\in[Q]}\right)$

That concludes description of $\mathbf{Hybrid}_0, \cdots, \mathbf{Hybrid}_7$. We now describe the sequence of sub-hybrids from Equation (10), which is used to show indistinguishability of $\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_1$. Looking ahead, in $\mathbf{Hybrid}_{j,0}$, $j \in [0, Q]$, the first j answers will have been switched to random. We then use $\mathbf{Hybrid}_{j,0}, \ldots, \mathbf{Hybrid}_{j,3}$ to switch the $(j+1)$-th answer to random (thus $\mathbf{Hybrid}_{j,3}$ will be identical to $\mathbf{Hybrid}_{j+1,0}$). Consequently, the initial hybrid $\mathbf{Hybrid}_{0,0}$ is identical to $\mathbf{Hybrid}_0$, whereas the final hybrid $\mathbf{Hybrid}_{Q,3}$ will be identical to $\mathbf{Hybrid}_1$.

– In $\mathbf{Hybrid}_{j,0}$, the input-output pairs corresponding to the first j queries are reprogrammed in F^*. To be precise, for $i < j$, the input corresponding to the i-th query, i.e., $(x_i \oplus \Delta, i)$, is reprogrammed to the random value $y_i^* \oplus b_i\Delta$; on all other inputs (x, i), F^* evaluates to $F_k(x, i)$. Note that, as a result, $\mathbf{Hybrid}_{0,0}$ is identical to $\mathbf{Hybrid}_0$

$\mathbf{Hybrid}_{j,0}(1^\kappa, \mathsf{QueryStrategy})$

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \cdots, y_j^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i\Delta & \text{if } x = x_i \oplus \Delta \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with values k, Δ and $((x_i, b_i, y_i^*))_{i\in[Q]}$ hard-coded.
4. $(x_{j+1}, b_{j+1}) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_j^*)$
5. $y_{j+1} := \mathcal{C}_{\mathsf{H},\Delta}(x_{j+1}, j+1, b_{j+1}) = \mathsf{H}(x_{j+1} \oplus \Delta, j+1) \oplus b_{j+1}\Delta$
6. For each $i \in [j+2, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*, y_{j+1}, y_{j+2} \ldots, y_{i-1})$
   (b) $y_i := \mathcal{C}_{\mathsf{H}^*,\Delta}(x_i, i, b_i) = \mathsf{H}^*(x_i \oplus \Delta, i) \oplus b_i\Delta$
7. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}, ((x_i, i, b_i, y_i))_{i\in[j+1,Q]}\right)$

– In $\mathbf{Hybrid}_{j,1}$, we program F^* *additionally* at the hash input corresponding to the $j+1$-th query, i.e., $(x_{j+1} \oplus \Delta, j+1)$. To this end, F^* is hard-coded with a key that is

punctured at $(x_{j+1} \oplus \Delta, j+1)$. Since punctured keys preserve functionality at all non-punctured points, the functionality of F^*s in both $\mathbf{Hybrid}_{j,0}$ and $\mathbf{Hybrid}_{j,1}$ is same, and we can argue their indistinguishability using iO security.

$\mathbf{Hybrid}_{j,1}(1^\kappa, \mathsf{QueryStrategy})$

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \cdots, y_j^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. $(x_{j+1}, b_{j+1}) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_j^*)$
4. Set $y_{j+1} := F_k(x_{j+1} \oplus \Delta, j+1) \oplus b_{j+1}\Delta$
5. Generate punctured key $k_{(x_{j+1}\oplus\Delta,1)} \leftarrow \mathsf{Puncture}_F(k, (x_{j+1} \oplus \Delta, 1))$
6. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i\Delta & \text{if } x = x_i \oplus \Delta \\ y_{j+1} \oplus b_{j+1}\Delta & \text{else if } x = x_{j+1} \oplus \Delta \\ F_{k_{(x_{j+1}\oplus\Delta, j+1)}}((x,i)) & \text{otherwise} \end{cases}$$

   with $k_{(x_{j+1}\oplus\Delta,j+1)}$, Δ, $((x_i, b_i, y_i^*))_{i\in[Q]}$ and $(x_{j+1}, b_{j+1}, y_{j+1})$ hard-coded.
7. For each $i \in [j+2, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*, y_{j+1}, y_{j+2} \ldots, y_{i-1})$
   (b) $y_i := \mathcal{C}_{\mathsf{H}^*,\Delta}(x_i, i, b_i) = \mathsf{H}^*(x_i \oplus \Delta, i) \oplus b_i\Delta$
8. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[Q]}, ((x_i, i, b_i, y_i))_{i\in[j+1,Q]}\right)$

– In $\mathbf{Hybrid}_{j,2}$, the value of F^* at $(x_{j+1} \oplus \Delta, j+1)$ is reprogrammed to a random value $y_{j+1}^* \oplus b_{j+1}\Delta$. This switch is indistinguishable due to pseudo-randomness of PPRF at punctured point (which holds even given the punctured key).

$\mathbf{Hybrid}_{j,2}(1^\kappa, \mathsf{QueryStrategy})$

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \cdots, y_j^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. $(x_{j+1}, b_{j+1}) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_j^*)$
4. Sample $y_{j+1}^* \leftarrow \{0,1\}^\kappa$
5. Generate punctured key $k_{(x_{j+1}\oplus\Delta,1)} \leftarrow \mathsf{Puncture}_F(k, (x_{j+1} \oplus \Delta, 1))$
6. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i\Delta & \text{if } x = x_i \oplus \Delta \\ y_{j+1}^* \oplus b_{j+1}\Delta & \text{else if } x = x_{j+1} \oplus \Delta \\ F_{k_{(x_{j+1}\oplus\Delta, 1)}}((x,i)) & \text{otherwise} \end{cases}$$

   with $k_{(x_{j+1}\oplus\Delta,j+1)}$, Δ and $((x_i, b_i, y_i^*))_{i\in[j+1]}$ hard-coded.
7. For each $i \in [j+2, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*, y_{j+1}, y_{j+2} \ldots, y_{i-1})$
   (b) $y_i := \mathcal{C}_{\mathsf{H}^*,\Delta}(x_i, i, b_i) = \mathsf{H}^*(x_i \oplus \Delta, i) \oplus b_i\Delta$
8. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i\in[j+1]}, ((x_i, i, b_i, y_i))_{i\in[j+2,Q]}\right)$

– Finally, in $\mathbf{Hybrid}_{j,3}$, we revert the key in F^* back to a normal key. Since functionality is unchanged, thanks to iO security the resulting distribution is indistinguishable from $\mathbf{Hybrid}_{j,2}$. In addition, notice that $\mathbf{Hybrid}_{j,3}$ is distributed identically to $\mathbf{Hybrid}_{j+1,0}$ since the differences between them are only syntactical.

$\mathbf{Hybrid}_{j,3}(1^\kappa, \mathsf{QueryStrategy})$

1. Sample $k \leftarrow \mathsf{Key}_F(1^\kappa)$, $r \leftarrow \{0,1\}^{poly(\kappa)}$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and $y_1^*, \cdots, y_j^* \leftarrow \{0,1\}^\kappa$
2. For each $i \in [Q]$: $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_{i-1}^*)$
3. $(x_{j+1}, b_{j+1}) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \ldots, y_j^*)$
4. Sample $y_{j+1}^* \leftarrow \{0,1\}^\kappa$
5. Generate $\mathsf{H}^*(\cdot) \leftarrow \mathsf{iO}(F^*(\cdot))$, where F^* is the program

$$F^*((x,i)) = \begin{cases} y_i^* \oplus b_i \Delta & \text{if } x = x_i \oplus \Delta \\ F_k((x,i)) & \text{otherwise} \end{cases}$$

   with k, Δ, $((x_i, b_i, y_i^*))_{i \in [j+1]}$ hard-coded.
6. For each $i \in [j+2, Q]$:
   (a) $(x_i, b_i) := \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*, y_{j+1}, y_{j+2} \ldots, y_{i-1})$
   (b) $y_i := \mathcal{C}_{\mathsf{H}^*,\Delta}(x_i, i, b_i) = \mathsf{H}^*(x_i \oplus \Delta, i) \oplus b_i \Delta$
7. Output $\left(\mathsf{H}^*, ((x_i, i, b_i, y_i^*))_{i \in [j+1]}, ((x_i, i, b_i, y_i))_{i \in [j+2,Q]}\right)$

This concludes the description of our hybrids, and the proof outline. We refer the readers to Appendix E for a formal proof of their indistinguishability.

References

1. Abdalla, M., Benhamouda, F., Passelègue, A.: Algebraic XOR-RKA-secure pseudorandom functions from post-zeroizing multilinear maps. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 386–412. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_14
2. Acharya, A., Ashur, T., Cohen, E., Hazay, C., Yanai, A.: A new approach to garbled circuits. In: Tibouchi, M., Wang, X. (eds.) ACNS 23, Part II. LNCS, vol. 13906, pp. 611–641. Springer, Heidelberg (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_23
3. Applebaum, B.: Garbling XOR gates "for free" in the standard model. Journal of Cryptology **29**(3), 552–576 (Jul 2016). https://doi.org/10.1007/s00145-015-9201-9
4. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. In: Chazelle, B. (ed.) ICS 2011. pp. 45–60. Tsinghua University Press (Jan 2011)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press (Oct 2004). https://doi.org/10.1109/FOCS.2004.20
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (Jul 2010). https://doi.org/10.1007/978-3-642-14165-2_14
7. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate or how to compress garbled circuits keys. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 166–184. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_10

8. Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 191–209. IEEE Computer Society Press (Oct 2015). https://doi.org/10.1109/FOCS.2015.21
9. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for Boolean and arithmetic circuits. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 565–577. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978410
10. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59** (2012), https://doi.org/10.1145/2160158.2160159
11. Barnum, C., Heath, D., Kolesnikov, V., Ostrovsky, R.: Adaptive garbled circuits and garbled ram from non-programmable random oracles. Cryptology ePrint Archive, Paper 2023/1527 (2023), https://eprint.iacr.org/2023/1527
12. 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). https://doi.org/10.1145/100216.100287
13. 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). https://doi.org/10.1109/SP.2013.39
14. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382279
15. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008. pp. 257–266. ACM Press (Oct 2008). https://doi.org/10.1145/1455770.1455804
16. Bitansky, N., Paneth, O., Wichs, D.: Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 474–502. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49096-9_20
17. Böhl, F., Davies, G.T., Hofheinz, D.: Encryption schemes secure under related-key and key-dependent message attacks. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 483–500. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_28
18. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_15
19. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_29
20. Brandão, L.T.A.N., Peralta, R.: NIST first call for multi-party threshold schemes (2023), https://csrc.nist.gov/pubs/ir/8214/c/ipd
21. Chaudhari, H., Choudhury, A., Patra, A., Suresh, A.: ASTRA: high throughput 3pc over rings with application to secure prediction. In: ACM SIGSAC 2019. pp. 81–92 (2019), https://doi.org/10.1145/3338466.3358922
22. Chaudhari, H., Rachuri, R., Suresh, A.: Trident: Efficient 4pc framework for privacy preserving machine learning. In: NDSS 2020. The Internet Society (2020), https://www.ndss-symposium.org/ndss-paper/trident-efficient-4pc-framework-for-privacy-preserving-machine-learning/
23. Chen, Y.L., Tessaro, S.: Better security-efficiency trade-offs in permutation-based two-party computation. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 275–304. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_10
24. 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, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9_3

25. Cui, H., Wang, X., Yang, K., Yu, Y.: Actively secure half-gates with minimum overhead under duplex networks. In: EUROCRYPT 2023. pp. 35–67 (2023), https://doi.org/10.1007/978-3-031-30617-4_2
26. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_25
27. Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 276–288. Springer, Heidelberg (Aug 1984)
28. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC. pp. 25–32. ACM Press (May 1989). https://doi.org/10.1145/73007.73010
29. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast garbling of circuits under standard assumptions. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 567–578. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813619
30. Guo, C., Katz, J., Wang, X., Weng, C., Yu, Y.: Better concrete security for half-gates garbling (in the multi-instance setting). In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 793–822. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_28
31. Guo, C., Katz, J., Wang, X., Yu, Y.: Efficient and secure multiparty computation from fixed-key block ciphers. In: 2020 IEEE Symposium on Security and Privacy. pp. 825–841. IEEE Computer Society Press (May 2020). https://doi.org/10.1109/SP40000.2020.00016
32. Guo, X., Yang, K., Wang, X., Yu, Y., Liu, Z.: Unmodified half-gates is adaptively secure - so is unmodified three-halves. Cryptology ePrint Archive, Paper 2023/1528 (2023), https://eprint.iacr.org/2023/1528, https://eprint.iacr.org/2023/1528
33. Heath, D.: Efficient arithmetic in garbled circuits. In: EUROCRYPT 2024. pp. 3–31 (2024), https://doi.org/10.1007/978-3-031-58740-5_1
34. Heath, D., Kolesnikov, V., Ostrovsky, R.: Tri-state circuits - A circuit model that captures RAM. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 128–160. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_5
35. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_9
36. Jafargholi, Z., Kamath, C., Klein, K., Komargodski, I., Pietrzak, K., Wichs, D.: Be adaptive, avoid overcommitting. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 133–163. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_5
37. Jafargholi, Z., Oechsner, S.: Adaptive security of practical garbling schemes. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) INDOCRYPT 2020. LNCS, vol. 12578, pp. 741–762. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-65277-7_33
38. Jafargholi, Z., Scafuro, A., Wichs, D.: Adaptively indistinguishable garbled circuits. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 40–71. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_2
39. Jafargholi, Z., Wichs, D.: Adaptive security of Yao's garbled circuits. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part I. LNCS, vol. 9985, pp. 433–458. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53641-4_17
40. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V. (eds.) 53rd ACM STOC. pp. 60–73. ACM Press (Jun 2021). https://doi.org/10.1145/3406325.3451093
41. Kamath, C., Klein, K., Pietrzak, K.: On treewidth, separators and yao's garbling. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 486–517. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90453-1_17
42. Kamath, C., Klein, K., Pietrzak, K., Wichs, D.: Limits on the adaptive security of yao's garbling. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826,

pp. 486–515. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_17
43. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 669–684. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516668
44. Kolesnikov, V., Mohassel, P., Rosulek, M.: FleXOR: Flexible garbling for XOR gates that beats free-XOR. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 440–457. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_25
45. 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, Heidelberg (Jul 2008). https://doi.org/10.1007/978-3-540-70583-3_40
46. 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). https://doi.org/10.1007/s00145-008-9036-8
47. Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: Oblivm: A programming framework for secure computation. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 359–376 (2015), https://doi.org/10.1109/SP.2015.29
48. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Blaze, M. (ed.) USENIX Security 2004. pp. 287–302. USENIX Association (Aug 2004)
49. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce. p. 129–139 (1999), https://doi.org/10.1145/336992.337028
50. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_22
51. 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, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_15
52. Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 94–124. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_5
53. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014). https://doi.org/10.1145/2591796.2591825
54. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC1. In: 40th FOCS. pp. 554–567. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814630
55. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 127–156. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5_5
56. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25
57. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_8

Supplementary Material

A Additional Preliminaries

Definition 8. *Two ensembles of probability distributions $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every PPT distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $\kappa \in \mathbb{N}$,*

$$\left| \Pr_{x\leftarrow X_\kappa}[D(1^\kappa, x) = 1] - \Pr_{y\leftarrow Y_\kappa}[D(1^\kappa, y) = 1] \right| < \epsilon(\kappa).$$

B Proof of Theorem 5

Theorem 5 *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa\in\mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 1 and 2 where, for security parameter κ, in the beginning of Gb a hash function H is sampled uniformly at random from $\mathcal{H}_\kappa$ and output together with the garbled circuit $\mathbf{F}$. Then $\mathsf{GS}^{\mathcal{H}}$ satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

Proof. The proof of the theorem would follow from the fact that if there existed a PPT distinguisher D that has non-negligible advantage ϵ in the PRIV-SIM security game (Definition 6), this can be used in a black-box way by a PPT adversary $\mathcal{A}$ to gain non-negligible advantage in the security game for weak CCR (Definition 7). To show this, let us first describe the PPT simulator Sim that the challenger in PRIV-SIM invokes.

Simulator for $\mathsf{GS}^{\mathcal{H}}$. The simulator for the garbling is a PPT algorithm $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))$ that needs to output the selective privacy challenge tuple created only using the circuit and the function output. Note that the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ garbles a circuit $\mathbf{C}$ gate by gate in topological order and Sim needs to output a tuple that contains values that are indistinguishable from those derived from the real garbling scheme on the circuit and input.

Let $\mathbf{X}$ denote the set of *active input labels* for the garbling $\mathbf{F}, \mathbf{d}$. For each wire in the circuit $\mathbf{C}$, we refer to the label derived from evaluating $\mathbf{F}$ using $\mathbf{X}$ as the *active label*. In the simulation, for each wire we assign the label representing the 0-value as the active label. Then, note that for each AND gate, the garbling algorithm can be re-written as:

$$\begin{aligned}
k_g^0 &= 2g - 1, k_g^1 = 2g, p_a = \mathsf{lsb}(\mathsf{L}_A^0), p_b = \mathsf{lsb}(\mathsf{L}_B^0)\\
G_g^0 &= \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_A^0, k_g^0, p_b)\\
G_g^1 &= \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_B^0, k_g^1, 0) \oplus \mathsf{L}_A^0\\
\mathsf{L}_C^0 &= \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A^0)
\end{aligned}$$

in terms of the active labels and the weak CCR hash oracle $\mathsf{H}(x \oplus \Delta, i) \oplus b\Delta = \mathcal{C}_{\mathsf{H},\Delta}(x, i, b)$ only. Further, note that throughout the garbling algorithm, the indices $k_g^b \in \mathbb{N}$ act as a counter for the oracle queries made, rendering the set of garbling oracle queries and responses as a legal query strategy satisfying weak CCR (Definition 7).

Given this observation, the simulator Sim works as follows: first, it samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$. Then, for each input wire, an active label is sampled as in the real garbling.

Then the garbling algorithm is executed with each instance of weak CCR oracle output being replaced by a κ-bit value sampled uniformly at random, as required in the construction. This process also derives the active wire label for all other wires in the circuit, including the output wires. Finally, the output decoding information is set such that the active output labels map to the function output as required. Algorithm 4 formalizes the actions of the simulator.

Algorithm 4 Simulator Sim for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$ adapted from [57]

1: **procedure** $\mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))$
2: sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ and initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
3: **for** every $i \in [\mathsf{n}]$ **do**
4: sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$
5: set $\mathbf{X}[i] = \mathsf{L}_i^0$
6: **end for**
7: **for** each $g \in [\mathsf{q}]$ in topological order **do**
8: parse gate $g = (A, B, C, f_g)$
9: **if** $f_g == \mathsf{XOR}$ **then**
10: set $\mathsf{L}_C^0 = \mathsf{L}_A^0 \oplus \mathsf{L}_B^0$
11: **else**
12: $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^0)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^0)$
13: sample $M_A, M_B \leftarrow \{0,1\}^\kappa$
14: $G_g^0 = \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus M_A$
15: $G_g^1 = \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus M_B \oplus \mathsf{L}_A^0$
16: $\mathsf{L}_C^0 = \mathsf{H}(\mathsf{L}_A^0, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B^0, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A^0)$
17: set $\mathbf{F}[g] = (G_g^0, G_g^1)$
18: **end if**
19: **end for**
20: **for** each $j \in [\mathsf{m}]$ **do**
21: set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^0) \oplus \mathbf{C}(\mathbf{X})[j]$
22: **end for**
23: **return** $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
24: **end procedure**

Hybrid Experiments. To prove the theorem, it now remains to show that the distribution of the output of this simulator is computationally indistinguishable from a tuple derived by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ that follow Algorithm 1. We do so by considering the following list of hybrid experiments:

– $\mathbf{Hybrid}_0$: This is the distribution that is derived as the outcome of Sim (Algorithm 4) over all the internal randomness used by the simulator.

$$\mathbf{Hybrid}_0 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x})), \kappa \in \mathbb{N}}$$

– $\mathbf{Hybrid}_1$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a random oracle $\mathcal{R}$, a PPT algorithm $\mathsf{Sim}^{\mathcal{R}}$ operates exactly as (Algorithm 4) except Step 13. Instead, in this experiment, the values M_A and M_B are

derived as,

$$M_A = \mathcal{R}(\mathsf{L}_A^0, k_g^0, p_b)$$
$$M_B = \mathcal{R}(\mathsf{L}_B^0, k_g^1, 0)$$

$$\mathbf{Hybrid}_1 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{Sim}^{\mathcal{R}}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x})), \kappa \in \mathbb{N}}$$

Due to the nature of the outputs of the random oracle, we have that this distribution is identical to the former one $\mathbf{Hybrid}_0 \equiv \mathbf{Hybrid}_1$.

– $\mathbf{Hybrid}_2$: This is a distribution that is derived as an outcome of a hybrid experiment where, first a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled. Then given access to a random oracle $\mathcal{R}$, a PPT algorithm $\mathsf{G}^{\mathcal{R},\mathsf{H}}$ operates using H as given in Algorithm 5. Unlike in the previous experiment, G receives the circuit input $\mathbf{x}$ as input and uses this to derive the values of all the active labels. Overall, this hybrid experiment differs from the previous one only in that the labeling of the active labels is no longer designated as 0, but it carries the same values as in the real execution of the garbling evaluation.

$$\mathbf{Hybrid}_2 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}^{\mathcal{R},\mathsf{H}}(1^\kappa, \mathbf{C}, \mathbf{x}), \kappa \in \mathbb{N}}$$

As such, this distribution is identical to the former one since the (re-)labeling of the active wire labels is never visible to the adversary $\mathbf{Hybrid}_1 \equiv \mathbf{Hybrid}_2$.

– $\mathbf{Hybrid}_3$: Here first a weak CCR secure hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled along with $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ sampled uniformly at random. Then $\mathcal{C}_{\mathsf{H},\Delta}$ is the weak CCR oracle that outputs, $\mathcal{C}_{\mathsf{H},\Delta}(x, i, b) = \mathsf{H}(x \oplus \Delta, i) \oplus b\Delta$. The distribution $\mathbf{Hybrid}_3$ is derived as an outcome of a hybrid experiment where, given access to an oracle $\mathcal{C}_{\mathsf{H},\Delta}$ and function H, a PPT algorithm $\mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta},\mathsf{H}}$ operates as given in Algorithm 5. G receives the circuit input $\mathbf{x}$ as input and uses this to derive the values of all the active labels.

$$\mathbf{Hybrid}_3 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta},\mathsf{H}}(1^\kappa, \mathbf{C}, \mathbf{x}), \kappa \in \mathbb{N}}$$

This hybrid experiment differs from the previous only in Step 13 in that the oracle query used to derive M_A and M_b are no longer made to the random oracle $\mathcal{R}$, but to $\mathcal{C}_{\mathsf{H},\Delta}$. We show later that this hybrid distribution can be shown as computationally indistinguishable from the previous by reduction to the weak CCR security of the hash function family $\mathbf{Hybrid}_2 \overset{c}{\approx} \mathbf{Hybrid}_3$.

– $\mathbf{Hybrid}_4$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a weak CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ as above, a PPT algorithm $\mathsf{G}_*^{\mathcal{C}_{\mathsf{H},\Delta}}$ operates as given in Algorithm 6. This differs from the previous hybrid experiment only in that the lines marked in blue additionally exist in Algorithm 6, where they were not present in Algorithm 5.

$$\mathbf{Hybrid}_4 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}_*^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa, \mathbf{C}, \mathbf{x}), \kappa \in \mathbb{N}}$$

As the extra lines executed in this hybrid experiment do not affect the output distribution, it follows that this distribution is identical to the previous one $\mathbf{Hybrid}_3 \equiv \mathbf{Hybrid}_4$.

– $\mathbf{Hybrid}_5$: This is the final hybrid experiment where the distribution is derived exactly as a tuple derived by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ in Algorithm 1.

$$\mathbf{Hybrid}_5 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),(\Delta,\mathbf{e}),\mathbf{d}) \leftarrow \mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^\kappa,\mathbf{C}),\mathbf{X}=\mathsf{GS}^{\mathcal{H}}.\mathsf{En}((\Delta,\mathbf{e}),\mathbf{x}),\kappa\in\mathbb{N}}$$

Algorithm 5 Hybrid Experiment $\mathsf{G}^{\mathcal{O},\mathsf{H}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}^{\mathcal{O},\mathsf{H}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$ and let $a, b \in \{0,1\}$ be the active input values
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^{\mathsf{XOR}(a,b)} = \mathsf{L}_A^a \oplus \mathsf{L}_B^b$
    else
      $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^a)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^b)$
      query $M_A = \mathcal{O}(\mathsf{L}_A^a, k_g^0, p_b)$ and $M_B = \mathcal{O}(\mathsf{L}_B^b, k_g^1, 0)$
      $G_g^0 = \mathsf{H}(\mathsf{L}_A^a, k_g^0) \oplus M_A$
      $G_g^1 = \mathsf{H}(\mathsf{L}_B^b, k_g^1) \oplus M_B \oplus \mathsf{L}_A^a$
      $\mathsf{L}_C^{\mathsf{AND}(a,b)} = \mathsf{H}(\mathsf{L}_A^a, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B^b, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A^a)$
      set $\mathbf{F}[g] = (G_g^0, G_g^1)$
    end if
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^{\mathbf{C}(\mathbf{x})[j]}) \oplus \mathbf{C}(\mathbf{x})[j]$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

This differs from the previous experiment in that, here, for each wire, instead of first calculating the active label and then deriving the inactive label with respect to it, the 0-label is calculated (irrespective of whether it is the active label) and the 1-label is derived. Note that in this distribution and the previous, the algebraic relationships between each element derived remain the same. As such, this distribution is identical to the previous one $\mathbf{Hybrid}_4 \equiv \mathbf{Hybrid}_5$.

Security Reduction to Weak CCR. To prove the theorem, it only remains now to show that $\mathbf{Hybrid}_2 \overset{\mathrm{c}}{\approx} \mathbf{Hybrid}_3$. We do so by demonstrating a security reduction to the weak CCR security property (Definition 7) of the underlying hash function family $\mathcal{H}$ used in the garbling scheme. Let D_{GS} be a PPT distinguisher that can distinguish between the hybrid distributions $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ with non-negligible advantage ϵ. If such a distinguisher exists, we show that it can be used in a black-box way by a PPT adversary $\mathcal{A}$ to win the weak CCR security game (Definition 7) for the hash function family $\mathcal{H}$ with non-negligible advantage as follows:

- The adversary $\mathcal{A}$ receives a circuit $\mathbf{C}$ and input $\mathbf{x}$ from D_{GS} that expects to receive a tuple of the form $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ in return.
- **Formulating a Query Strategy:** Given $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ with $\mathsf{q}_{\mathsf{AND}}$ AND gates, and input $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$, the adversary $\mathcal{A}$ designs a PPT algorithm with the interface,

$$\mathcal{Q} = \{(x_j, j, b_j, y_j)\}_{j \in [2\mathsf{q}_{\mathsf{AND}}]} = \mathsf{QueryStrategy}_{\mathbf{C},\mathbf{x}}^{\mathcal{O}}(1^\kappa; r)$$

Algorithm 6 Hybrid Experiment $\mathsf{G}_*^{\mathcal{O}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}_*^{\mathcal{O}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and set $\mathcal{O} = \mathcal{C}_{\mathsf{H},\Delta}$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$
    set $\mathsf{L}_i^{\neg\mathbf{x}[i]} = \mathsf{L}_i^{\mathbf{x}[i]} \oplus \Delta$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$ and let $a, b \in \{0,1\}$ be the active input values
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^{c=\mathsf{XOR}(a,b)} = \mathsf{L}_A^a \oplus \mathsf{L}_B^b$
    else
      $k_g^0 = 2g - 1$, $k_g^1 = 2g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^a)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^b)$
      query $M_A = \mathcal{O}(\mathsf{L}_A^a, k_g^0, p_b)$ and $M_B = \mathcal{O}(\mathsf{L}_B^b, k_g^1, 0)$
      $G_g^0 = \mathsf{H}(\mathsf{L}_A^a, k_g^0) \oplus M_A$
      $G_g^1 = \mathsf{H}(\mathsf{L}_B^b, k_g^1) \oplus M_B \oplus \mathsf{L}_A^a$
      $\mathsf{L}_C^{c=\mathsf{AND}(a,b)} = \mathsf{H}(\mathsf{L}_A^a, k_g^0) \oplus (p_a \cdot G_g^0) \oplus \mathsf{H}(\mathsf{L}_B^b, k_g^1) \oplus p_b(G_g^1 \oplus \mathsf{L}_A^a)$
      set $\mathbf{F}[g] = (G_g^0, G_g^1)$
    end if
    set $\mathsf{L}_C^{\neg c} = \mathsf{L}_C^c \oplus \Delta$
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^{\mathbf{C}(\mathbf{x})[j]}) \oplus \mathbf{C}(\mathbf{x})[j]$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

where the strategy accepts randomness of the form $r \leftarrow \{0,1\}^{\mathsf{n}\kappa}$. This is detailed in Algorithm 3. This is the algorithm that is given to the challenger $\mathcal{C}$ for the weak CCR security game.

- The challenger $\mathcal{C}$ accepts this $\mathsf{QueryStrategy}^{\mathcal{O}}$ and operates as implicit in Definition 7. That is, it samples a bit $c \leftarrow \{0,1\}$ and if $c = 0$, it samples randomness r and executes $\mathsf{QueryStrategy}^{\mathcal{R}}_{\mathbf{C},\mathbf{x}}$. Otherwise, it samples r and Δ, and executes $\mathsf{QueryStrategy}^{\mathcal{C}_{\mathsf{H},\Delta}}_{\mathbf{C},\mathbf{x}}$. It gives a tuple $(\mathcal{Q}, \mathsf{H})$ to $\mathcal{A}$.
- Given the set of query responses $\mathcal{Q}$ and the hash algorithm H, the adversary $\mathcal{A}$ can now execute Algorithm 5 with active input labels as indicated in the queries in $\mathcal{Q}$ and using the items in $\mathcal{Q}$ where ever oracle queries are required. The output of this is a tuple $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ which is distributed as $\mathbf{Hybrid}_2$ if the bit chosen by the challenger is $c = 0$, and is distributed as $\mathbf{Hybrid}_3$ otherwise.
- $\mathcal{A}$ gives $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ to D_{GS} and outputs whatever it outputs.

Note that in the above game, $\mathcal{A}$ has the same advantage as that of D_{GS}, which is non-negligible. However, since H is sampled from a weak CCR secure function family $\mathcal{H}_\kappa$, it follows that no such $\mathcal{A}$ can exist and therefore no such D_{GS} can exist. This completes the proof.

C Proof of Theorem 6

In this section, we first begin with fully describing the garbling scheme from [52], adapted to using a hash function H that is sampled from a weak CCR secure hash function family $\mathcal{H}$ (Definition 7). This is a security definition that they introduce which relaxes CCR security (Definition 4) in the following ways:

- In the security game for RTCCR, the hash function H is drawn from a family of hash functions, and the adversary only receives the description of H *after* making all of its oracle queries.
- The function $\mathsf{H} : \{0,1\}^{\mathsf{n}} \times \mathbb{N} \to \{0,1\}^{\mathsf{m}}$ may have different input and output lengths. Further, the CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ in the RTCCR security game receives a linear function $L : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ (instead of a bit b) and XORs the output of the hash function with $L(\Delta)$ (opposed to $b \cdot \Delta$). For the application in the garbling scheme (Algorithms 7 and 8) the hash function $\mathsf{H} \in \mathcal{H}$ is s.t. $\mathsf{H} : \{0,1\}^{\kappa} \times \mathbb{N} \to \{0,1\}^{\frac{\kappa}{2}}$ and the linear functions required are of the form,

$$L_{a,b}(\Delta_L, \Delta_R) = a \cdot \Delta_L \oplus b \cdot \Delta_R$$

  where $(a,b) \in \{(0,0),(0,1),(1,0)\}$ and Δ_L and Δ_R are the $\frac{\kappa}{2}$-bit prefix and suffix of Δ respectively.

Definition 9. *For polynomial functions $\mathsf{n}(\cdot)$ and $\mathsf{m}(\cdot)$ and a computational security parameter κ, a family of hash functions $\mathcal{H}_\kappa$, where each $\mathsf{H} \in \mathcal{H}_\kappa$ maps $\{0,1\}^{\mathsf{n}(\kappa)} \times \mathcal{T} \to \{0,1\}^{\mathsf{m}(\kappa)}$ for some set of tweaks $\mathcal{T}$, is **randomized tweakable circular correlation robust** (RTCCR) for a set of linear functions $\mathcal{L}$ from $\{0,1\}^{\mathsf{n}(\kappa)}$ to $\{0,1\}^{\mathsf{m}(\kappa)}$ if, for any PPT adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ that never repeat an oracle query to $\mathcal{C}_{\mathsf{H},\Delta}$ on the same (X,τ),*

$$\left| \Pr_{\sigma \leftarrow \mathcal{A}_1^{\mathsf{H},\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa)} \left[\mathcal{A}_2(\sigma,\mathsf{H}) = 1\right] - \Pr_{\sigma \leftarrow \mathcal{A}_1^{\mathsf{H},\mathcal{R}}(1^\kappa)} \left[\mathcal{A}_2(\sigma,\mathsf{H}) = 1\right] \right|$$

is negligible, where the probabilities are taken over the choice of $\Delta \leftarrow \{0,1\}^{\mathsf{n}(\kappa)-1}||1$, the random oracle $\mathcal{R} : \{0,1\}^{\mathsf{n}(\kappa)} \times \mathcal{T} \times \mathcal{L} \to \{0,1\}^{\mathsf{m}(\kappa)}$ and the hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$. The oracle $\mathcal{C}_{\mathsf{H},\Delta}$ is defined as,

$$\mathcal{C}_{\mathsf{H},\Delta}(X \in \{0,1\}^{\mathsf{n}(\kappa)}, \tau \in \mathcal{T}, L \in \mathcal{L}) = \mathsf{H}(X \oplus \Delta, \tau) \oplus L(\Delta)$$

The Garbling Scheme. For a single AND gate with input wires denoted by A and B, and the output wire denoted by C, [52] abstracts the garbling in the form of the following system of linear equations:

$$\mathbf{V} \begin{bmatrix} \vec{\mathbf{C}} \\ \vec{\mathbf{G}} \end{bmatrix} = \mathbf{M}\vec{\mathbf{H}} \oplus (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix} \vec{\mathbf{A}} \\ \vec{\mathbf{B}} \\ \vec{\mathbf{\Delta}} \end{bmatrix} \tag{11}$$

where $\mathbf{V}, \mathbf{M}, \mathbf{R}, \mathbf{T}$ are binary matrices and $\vec{\mathbf{A}}, \vec{\mathbf{B}}, \vec{\mathbf{\Delta}}, \vec{\mathbf{C}}, \vec{\mathbf{G}}, \vec{\mathbf{H}}$ are vectors. Where it is clear from the context these may be denoted as $\mathbf{A}, \mathbf{B}, \mathbf{\Delta}, \mathbf{C}, \mathbf{G}, \mathbf{H}$, without the vector notation. . Informally, the optimization in the garbling gate size of [52] is derived from treating each κ-bit wire label as having two *slices* of $\frac{\kappa}{2}$ bits each. In this equation $\vec{\mathbf{C}}$ is the vector of the

output label slices (where $\mathsf{L}_C^0 = \mathbf{C}[0]||\mathbf{C}[1]$) and $\vec{\mathbf{G}}$ is the vector of ciphertexts that the garbled gate is composed of. The matrix $\mathbf{V}$ of binary values acts as indicators that dictate how $\mathbf{C}$ and $\mathbf{G}$ relate when evaluating a garbled row of the gate functionality.

$\vec{\mathbf{H}}$ is the vector of hash output combinations possible for given input wires and their labels. While in [52] this contains $\frac{\kappa}{2}$-bit elements as the hash outputs, we will explain in the ensuing text how one can use a hash function H with κ-bit outputs and truncate these to $\frac{\kappa}{2}$-bit in order to achieve the same results. The matrix $\mathbf{M}$ contains indicators that dictate how these $\frac{\kappa}{2}$-bit elements in $\mathbf{H}$ are used to create each ciphertext in $\mathbf{G}$.

$\vec{\mathbf{A}}, \vec{\mathbf{B}}$ are vectors of input label slices and $\vec{\mathbf{\Delta}}$ is the vector of the free-XOR constant slices. The above can be simplified to linear garbling if the vectors $\vec{\mathbf{A}}, \vec{\mathbf{B}}, \vec{\mathbf{C}}, \vec{\mathbf{\Delta}}$ are replaced with respective scalars $\mathsf{L}_A^0, \mathsf{L}_B^0, \mathsf{L}_C^0, \Delta$. The matrix $\mathbf{T}$ of binary values indicates a permutation of the truth table of the binary AND gate according to the colour bits of the input wires. $\mathbf{R}$ is termed the control matrix. It contains binary values that indicate how the plaintext slices of the labels are used (along with hash outputs) to create each ciphertext in $\mathbf{G}$.

We now explain the details of the garbling scheme [52], for garbling an AND gate, including specifying how the matrices $\mathbf{V}, \mathbf{M}, \mathbf{R}, \mathbf{T}$ are defined.

L.H.S. of Equation 11. The Boolean AND function over two inputs can be represented using a truth table containing 4 rows. Corresponding to each row, the garbling scheme in [52] separately defines how the $\frac{\kappa}{2}$-bit prefix $\mathbf{C}[0]$ and suffix $\mathbf{C}[1]$ of the output label is derived using the values in $\mathbf{G}$. The vector of ciphertexts $\mathbf{G}$ contains 3 'halves' – $\frac{\kappa}{2}$-bit strings. The matrix $\mathbf{V}$ contains 8 rows – one corresponding to each truth table row and slice in $\mathbf{C}$. It contains 5 columns, each indicating a linear combination of elements in $\mathbf{C}$ and $\mathbf{G}$ as given below.

$$\mathbf{V}\begin{bmatrix}\vec{\mathbf{C}}\\ \vec{\mathbf{G}}\end{bmatrix} = \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\1&0&0&0&1\\0&1&0&1&1\\1&0&1&0&1\\0&1&0&0&1\\1&0&1&0&0\\0&1&0&1&0\end{bmatrix}\begin{bmatrix}\mathbf{C}[0]\\ \mathbf{C}[1]\\ \mathbf{G}[0]\\ \mathbf{G}[1]\\ \mathbf{G}[2]\end{bmatrix}$$

From the above, we can deduce the linear combination of garbling ciphertexts that are involved in deriving each output label slice for each row of the truth table for AND gates. This is indicated in Table 1.
The R.H.S. of Equation 11 indicates which linear combinations of the input labels and the hashes thereof are involved in the garbling for each row in the AND gate truth table, for deriving each slice of the output label. Going forward, we first discuss the combination of hash outputs and then that of the plaintext input labels themselves.

Hash Outputs in Equation 11. The garbling scheme in [52] employs a hash function $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^{\frac{\kappa}{2}}$ that is RTCCR secure. Calls to this hash function are made using κ-bit inputs and, for a garbled AND gate, $\mathbf{H}$ given below is the vector containing all the different ways this function is used. This includes 4 possible calls with only one input wire label and 2 that can be made by combining two input labels, one from each input wire. The matrix $\mathbf{M}$ contains 8 rows – one corresponding to each truth table row and slice in $\mathbf{C}$. In

Inputs	Ciphertext
$(0,0,L)$	–
$(0,0,R)$	–
$(0,1,L)$	$\mathbf{G}[2]$
$(0,1,R)$	$\mathbf{G}[1] \oplus \mathbf{G}[2]$
$(1,0,L)$	$\mathbf{G}[0] \oplus \mathbf{G}[2]$
$(1,0,R)$	$\mathbf{G}[2]$
$(1,1,L)$	$\mathbf{G}[0]$
$(1,1,R)$	$\mathbf{G}[1]$

Table 1. Cipertexts for each row in GS^{H}. L indicates $\mathbf{C}[0]$ and R indicates $\mathbf{C}[1]$.

Algorithm 7 Three-halves Garbling GS^{H} for circuits with XOR and AND gates

procedure $\mathsf{Gb}(1^\kappa, \mathbf{C})$
  initialize $\mathbf{F} = []$, $\mathbf{e} = []$ and $\mathbf{d} = []$
  sample $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$ and set $\mathsf{L}_i^1 = \mathsf{L}_i^0 \oplus \Delta$
    $\mathbf{e}[i] = \mathsf{L}_i^0$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$
    if $f_g = \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^0 = \mathsf{L}_A^0 \oplus \mathsf{L}_B^0$ and $\mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$
    else
      derive $(\mathsf{L}_C^0, \mathbf{G}, \mathbf{z}) \leftarrow \mathsf{GbAND}(\mathsf{L}_A^0, \mathsf{L}_B^0)$
      set $\mathsf{L}_C^1 = \mathsf{L}_C^0 \oplus \Delta$
      $\mathbf{F}[g] = (\mathbf{G}, \mathbf{z})$
    end if
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^0)$
  end for
  return $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$
end procedure

procedure $\mathsf{GbAND}(\mathsf{L}_A^0, \mathsf{L}_B^0)$
  set $p_a = \mathsf{lsb}(\mathsf{L}_A^0)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^0)$
  derive $\mathbf{T} \leftarrow \mathsf{AND}(p_a, p_b)$
  sample $\mathbf{R}, \mathbf{r} \leftarrow \mathsf{SampleR}(\mathbf{T})$
  compute $\mathbf{z} \parallel \begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = \mathbf{V}^{-1}(\mathbf{r} \parallel (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathsf{L}_A^0\\ \mathsf{L}_B^0\\ \Delta\end{bmatrix} \oplus \mathbf{MH})$
  $\mathsf{L}_C^0 = \mathbf{C}[0]||\mathbf{C}[1]$
  return $(\mathsf{L}_C^0, \mathbf{G}, \mathbf{z})$
end procedure

our construction, replacing the hash function with one that is weak CCR secure, for the k^{th} AND gate, it contains 6 columns, each indicating a linear combination of elements in $\mathbf{H}$ as given below.

$$\mathbf{M}\vec{\mathbf{H}} = \begin{bmatrix} 1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,1\,0 \end{bmatrix} \begin{bmatrix} \frac{\kappa}{2} \text{ prefix of } & \mathsf{H}(\mathsf{L}_A^0, 3k-3) \\ \frac{\kappa}{2} \text{ prefix of } & \mathsf{H}(\mathsf{L}_A^1, 3k-3) \\ \frac{\kappa}{2} \text{ prefix of } & \mathsf{H}(\mathsf{L}_B^0, 3k-2) \\ \frac{\kappa}{2} \text{ prefix of } & \mathsf{H}(\mathsf{L}_B^1, 3k-2) \\ \frac{\kappa}{2} \text{ suffix of } & \mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, 3k-1) \\ \frac{\kappa}{2} \text{ suffix of } & \mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^1, 3k-1) \end{bmatrix} \tag{12}$$

Note that these linear combinations contain only hashes that an evaluator is able to compute for the particular truth table row and slice with the active input labels available to it.

Tweaks. For the k^{th} AND gate garbling, within $\mathbf{H}$, the tweak $3k-3$ is used for hash calls of the form $\mathsf{H}(\mathsf{L}_A^0)$ and $\mathsf{H}(\mathsf{L}_A^1)$ involving labels of wire A only; the tweak $3k-2$ is used in hash calls involving labels of wire B only; and $3k-1$ is used in hash calls involving the XOR of two labels. We omit this from the equation 12 for simplicity.

In order to replace this with a hash function sampled from a weak CCR secure function family, we require an algorithm that can take the κ-bit output of such a hash function and convert it to $\frac{\kappa}{2}$-bit values in such a way that the CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ is usable in a black-box way for garbling – replacing hash calls that involve Δ. Towards this, note that during evaluation, each slice of the output label is derived separately using the input labels and the ciphertexts. So during garbling, for each truth table row, we make calls to a hash function H sampled from a weak CCR secure function family $\mathcal{H}$ as indicated by $\mathbf{M}$. Then with the κ-bit output derived, for computing $\mathbf{C}[0]$ and $\mathbf{C}[1]$, we use the $\frac{\kappa}{2}$-bit prefix and suffix respectively. This preserves the correctness and privacy of the given system of equations.

Input Labels in Equation 11. It now remains to discuss how $\mathbf{T}$ and the control matrix $\mathbf{R}$ are sampled. $\mathbf{T} \in \{0,1\}^{8\times 6}$ is an indicator matrix that dictates whether (either the $\frac{\kappa}{2}$-bit prefix or suffix of) $\mathbf{\Delta}$ is XOR-ed in the equation. Choosing $\mathbf{T}$ is straightforward as the truth table of the AND gate and the colour bits chosen for the input wires in a particular instantiation of the scheme fixes $\mathbf{T}$. If we have the permute bits to be (π_A, π_B), then:

$$\mathbf{T} = \begin{bmatrix} \mathbf{0}\,\mathbf{0} & g(\pi_A, \pi_B) \\ \mathbf{0}\,\mathbf{0} & g(\pi_A, 1 \oplus \pi_B) \\ \mathbf{0}\,\mathbf{0} & g(1 \oplus \pi_A, \pi_B) \\ \mathbf{0}\,\mathbf{0} & g(1 \oplus \pi_A, 1 \oplus \pi_B) \end{bmatrix} \tag{13}$$

where $g(x,y) = \begin{cases} \mathbf{0}_{2\times 2} & \mathsf{AND}(x,y) = 0 \\ \mathbf{I}_{2\times 2} & \mathsf{AND}(x,y) = 1 \end{cases}$

It remains to describe $\mathbf{R}$ that indicates the combination of input labels required in the clear for evaluation. In [52] the input combinations for each value of the permute bits are chosen separately by the garbler. While this would reveal information about the permute bits itself, they are encrypted such that the evaluator gets access only to the part of the matrix $\mathbf{R}$ that is chosen for the active labels. Security is proven in context of an adversarial

evaluator that only receives this marginal view. [52] give a complete description of how this matrix is sampled, which we abstract here as the pair of algorithms $(\mathsf{SampleR}, \mathsf{DecodeR})$. This completes the description of the three-halves garbling scheme.

Algorithm 8 Three-halves Scheme GS^{H} Encoding, Decoding, and Evaluation Algorithms

procedure $\mathsf{En}((\Delta, \mathbf{e}), \mathbf{x})$
  initialize $\mathbf{X} = []$
  for each $i \in [\mathsf{n}]$ **do**
    set $\mathbf{X}[i] = \mathbf{e}[i] \oplus \mathbf{x}[i]\Delta$
  end for
  return $\mathbf{X}$
end procedure

procedure $\mathsf{De}(\mathbf{Y}, \mathbf{d})$
  initialize $\mathbf{y} = []$
  for $j \in [\mathsf{m}]$ **do**
    $\mathbf{y}[j] = \mathbf{d}[j] \oplus \mathsf{lsb}(\mathbf{Y}[j])$
  end for
  return $\mathbf{y}$
end procedure

procedure $\mathsf{Ev}(\mathbf{F}, \mathbf{X})$
  initialize $\mathbf{Y} = []$
  for each gate $g \in [\mathsf{q}]$ in topological order **do**
    $\mathsf{L}_A, \mathsf{L}_B \leftarrow$ active labels associated with input wires of gate g
    if $f_g = \mathsf{XOR}$ **then**
      $\mathsf{L}_C = \mathsf{L}_A \oplus \mathsf{L}_B$
    else
      $\mathbf{E}[g] \leftarrow \mathsf{EvalAND}(\mathbf{F}[g], \mathsf{L}_A, \mathsf{L}_B)$
      $\mathsf{L}_C = \mathbf{E}[g][0] || \mathbf{E}[g][1]$
    end if
    if C is a circuit output wire **then**
      $\mathbf{Y}[C] = \mathsf{L}_C$
    end if
  end for
  return $\mathbf{Y}$
end procedure

procedure $\mathsf{EvalAND}(\mathbf{F} = (\mathbf{G}, \mathbf{z}), \mathsf{L}_A, \mathsf{L}_B)$
  $s_a = \mathsf{lsb}(\mathsf{L}_A),\ s_b = \mathsf{lsb}(\mathsf{L}_B)$
  $\mathbf{H}_e^\top = [\mathsf{H}(\mathsf{L}_A, 3g-3) \quad \mathsf{H}(\mathsf{L}_B, 3g-2) \quad \mathsf{H}(\mathsf{L}_A \oplus \mathsf{L}_B, 3g-1)]$
  compute $\mathbf{r} || \mathbf{X}[s_a, s_b] = \mathbf{V}[s_a, s_b] \left(\mathbf{z} || \begin{bmatrix} \mathbf{0} \\ \mathbf{G} \end{bmatrix} \right) \oplus \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \mathbf{H}_e$
  $\mathbf{R}[i, j] = \mathsf{DecodeR}(\mathbf{r}, s_a, s_b)$
  set $\mathbf{E} = \mathbf{X}[s_a, s_b] \oplus \mathbf{R}[s_a, s_b] \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$
  return $\mathbf{E}$
end procedure

C.1 Proof

Theorem 6 *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa\in\mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 7 and 8 where, for security parameter κ, in the beginning of Gb a hash function H is sampled uniformly at random from $\mathcal{H}_\kappa$ and output together with the garbled circuit $\mathbf{F}$. Then $\mathsf{GS}^{\mathcal{H}}$ satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

Proof Outline. The proof for this statement is adapted from [52] and considers the same description of the simulator and list of hybrids. The simulator for the garbling is a PPT algorithm $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))$ that needs to output the selective privacy challenge tuple created only using the circuit and the function output. In the same way as in the proof for Theorem 5, in the simulation, for each wire the label representing the 0-value is assigned as the active label. Then the vector of hash evaluations $\mathbf{H}$ in the garbling algorithm for each k^{th} AND gate in the scheme from [52] can be re-written in terms of

$$\begin{aligned}
\tfrac{\kappa}{2}\text{-bit prefixes of} \quad & \mathsf{H}(\mathsf{L}_A^0, 3k-3) \quad \text{and} \quad \mathsf{H}(\mathsf{L}_A^1, 3k-3) \\
\tfrac{\kappa}{2}\text{-bit prefixes of} \quad & \mathsf{H}(\mathsf{L}_B^0, 3k-2) \quad \text{and} \quad \mathsf{H}(\mathsf{L}_B^1, 3k-2) \\
\tfrac{\kappa}{2}\text{-bit suffixes of} \quad & \mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, 3k-1) \quad \text{and} \quad \mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^1, 3k-1)
\end{aligned}$$

and is used in the algorithm as given by $\mathbf{MH}$, where the indicators in $\mathbf{M}$ dictate which of the above hash calls are activated. The R.H.S. of the system of garbling equations, written in terms of the labels L_A^0 and L_B^0, can be re-written as,

$$(\mathbf{R}\oplus\mathbf{T})\begin{bmatrix}\mathsf{L}_A^0\\ \mathsf{L}_B^0\\ \Delta\end{bmatrix} \oplus \mathbf{MH} = (\mathbf{R}\oplus\mathbf{T})\begin{bmatrix}\mathsf{L}_A^0\\ \mathsf{L}_B^0\\ \mathbf{0}\end{bmatrix} \oplus (\mathbf{R}\oplus\mathbf{T})\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$$

Within the part of this system indicated in blue, weak CCR hash evaluations using labels L_A^1 and L_B^1 can be re-written in terms of the following oracle calls:

$$\mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_A^0, 3k-3, b) \quad \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_B^0, 3k-2, b) \quad \mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, 3k-1, b)$$

using only the labels L_A^0 and L_B^0, where the last input bit $b \in \{0,1\}$ can be set according to the indicators in $(\mathbf{R}\oplus\mathbf{T})$ that dictate whether Δ needs to be XOR-ed to the output. Further, throughout the garbling algorithm, the indices $3k - c \in \mathbb{N}$ ($c \in \{3,2,1\}$) act as a counter for the oracle queries made, rendering the set of garbling oracle queries and responses as a legal query strategy satisfying weak CCR (Definition 7).

Given this observation, the simulator Sim works as follows: first, it samples a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$. Then, for each input wire, an active label is sampled as in the real garbling. Then the garbling algorithm is executed with each instance of weak CCR oracle output being replaced by a $\frac{\kappa}{2}$-bit value sampled uniformly at random, as required in the construction. This process also derives the active wire label for all other wires in the circuit, including the output wires. Finally, the output decoding information is set such that the active output labels map to the function output as required.

Like in the proof for Theorem 5, within the list of hybrid experiments to prove that the real and simulated distributions are computationally indistinguishable, all but one set of adjacent hybrids are identically distributed. Within the pair of adjacent hybrid distributions

that are not identically distributed, in one hybrid experiment, the garbled circuit is generated in such a way that the oracle calls described above are made to the random oracle $\mathcal{R}$. In the other hybrid experiment, these are calls to $\mathcal{C}_{\mathsf{H},\Delta}$. Both experiments are otherwise identical.

This pair of adjacent hybrids can be shown as computationally indistinguishable by reduction to the weak CCR property of the hash function family $\mathcal{H}$ where the adversary's query strategy this time is based on the garbling and encoding algorithm in GS^{H} (Algorithm 7– [52]). This concludes the proof.

Proof. To prove the theorem we show that if there existed a PPT distinguisher D that has non-negligible advantage ϵ in the PRIV-SIM security game (Definition 6), this can be used in a black-box way by a PPT adversary A to gain non-negligible advantage in the security game for weak CCR security of the hash function family $\mathcal{H}$ (Definition 7). To show this, let us first describe the PPT simulator Sim that the challenger in PRIV-SIM invokes.

Simulator for $\mathsf{GS}^{\mathcal{H}}$. The simulator for the garbling is a PPT algorithm $((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{Sim}(1^{\kappa},\mathbf{C},\mathbf{C}(\mathbf{x}))$ that needs to output the selective security challenge tuple created only using the circuit and the function output. Note that the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ garbles a circuit $\mathbf{C}$ gate by gate in topological order and Sim needs to output a tuple that contains values that are indistinguishable from those derived from the real garbling scheme on the circuit and input.

Let $\mathbf{X}$ denote the set of *active input labels* for the garbling $\mathbf{F},\mathbf{d}$. That is, for a real garbling $(\mathbf{F},\mathbf{d})$, this is the input encoding $\mathbf{X} = \mathsf{GS}^{\mathcal{H}}.\mathsf{En}((\boldsymbol{\Delta},\mathbf{e}),\mathbf{x})$. For each wire in the circuit $\mathbf{C}$, we refer to the label derived from evaluating $\mathbf{F}$ using $\mathbf{X}$ as the *active label*. In the simulation, for each wire we assign the label representing the 0-value as the active label. Then, note that for the k^{th} AND gate in the circuit, the system

$$(\mathbf{R}\oplus\mathbf{T})\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix}\oplus\mathbf{MH}$$

can be represented using calls to $\mathcal{C}_{\mathsf{H},\Delta}$ and H on 0-labels only. Here the array $\vec{\mathbf{H}}$ in the garbling algorithm can be written as:

$$\vec{\mathbf{H}} = \begin{bmatrix} \frac{\kappa}{2}\text{ prefix of} & \mathsf{H}(\mathsf{L}_A^0, 3k-3)\\ \frac{\kappa}{2}\text{ prefix of} & \mathsf{H}(\mathsf{L}_A^1, 3k-3)\\ \frac{\kappa}{2}\text{ prefix of} & \mathsf{H}(\mathsf{L}_B^0, 3k-2)\\ \frac{\kappa}{2}\text{ prefix of} & \mathsf{H}(\mathsf{L}_B^1, 3k-2)\\ \frac{\kappa}{2}\text{ suffix of} & \mathsf{H}(\mathsf{L}_A^0\oplus\mathsf{L}_B^0, 3k-1)\\ \frac{\kappa}{2}\text{ suffix of} & \mathsf{H}(\mathsf{L}_A^0\oplus\mathsf{L}_B^1, 3k-1)\end{bmatrix}$$

in terms of the 0-labels and Δ. These values are 'activated' depending on the entries in $\mathbf{M}$. The even entries in this vector can be implemented through calls to $\mathcal{C}_{\mathsf{H},\Delta}$ where the last entry $b\in\{0,1\}$ in $\mathcal{C}_{\mathsf{H},\Delta}(\cdot,\cdot,\cdot)$ is determined using the entries in $(\mathbf{R}\oplus\mathbf{T})$. This allows representing the garbling equation in terms of the active labels and the weak CCR hash oracle $\mathsf{H}(x\oplus\Delta,i)\oplus b\Delta = \mathcal{C}_{\mathsf{H},\Delta}(x,i,b)$ only (where either the $\frac{\kappa}{2}$-bit prefix or suffix is used). Further, note that throughout the garbling algorithm, the indices $k_g^c\in\mathbb{N}$ act as a counter for the oracle queries made, rendering the set of garbling oracle queries and responses as a legal query strategy satisfying weak CCR (Definition 7).

Given this observation, the simulator Sim works as follows: first, it samples a hash function $\mathsf{H}\leftarrow\mathcal{H}_{\kappa}$. Then, for each input wire, an active label is sampled as in the real garbling.

Then the garbling algorithm is executed with each instance of weak CCR oracle output being replaced by a $\frac{\kappa}{2}$-bit value sampled uniformly at random, as required in the construction. This process also derives the active wire label for all other wires in the circuit, including the output wires. Finally, the output decoding information is set such that the active output labels map to the function output as required. Algorithm 9 formalizes the actions of the simulator.

Algorithm 9 Simulator Sim for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$ adapted from [52]

procedure $\mathsf{Sim}(1^\kappa, \mathbf{C}, \mathbf{C}(\mathbf{x}))$
  sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ and initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$
    set $\mathbf{X}[i] = \mathsf{L}_i^0$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^0 = \mathsf{L}_A^0 \oplus \mathsf{L}_B^0$
    else
      $k_g^0 = 3g - 2$, $k_g^1 = 3g - 1$, $k_g^2 = 3g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^0)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^0)$
      sample $M_A, M'_A, M_B, M'_B, M_{AB}, M'_{AB} \leftarrow \{0,1\}^{\frac{\kappa}{2}}$
      derive $\mathbf{T} \leftarrow \mathsf{AND}(p_a, p_b)$
      sample $\mathbf{R}, \mathbf{r} \leftarrow \mathsf{SampleR}(\mathbf{T})$
      compute $\mathbf{z} \parallel \begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = \mathbf{V}^{-1}(\mathbf{r} \parallel (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathsf{L}_A^0\\ \mathsf{L}_B^0\\ \Delta\end{bmatrix} \oplus \mathbf{MH})$
        where $(\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$ is computed as:
      – use hash calls $\mathsf{H}(\mathsf{L}_A^0, k_g^0)$, $\mathsf{H}(\mathsf{L}_B^0, k_g^1)$ and $\mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, k_g^2)$ as indicated in $\mathbf{MH}$
      – for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^1, k_g^0)$,
        • use M_A if entry for $\boldsymbol{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$ is 0 and use M'_A if it is 1
      – for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_B^1, k_g^1)$,
        • use M_B if entry for $\boldsymbol{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$ is 0 and use M'_B if it is 1
      – for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^1, k_g^2)$,
        • use M_{AB} if entry for $\boldsymbol{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$ is 0 and use M'_{AB} if it is 1
      set $\mathsf{L}_C^0 = \mathbf{C}[0] || \mathbf{C}[1]$
      set $\mathbf{F}[g] = \mathbf{G}$
    end if
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^0) \oplus \mathbf{C}(\mathbf{X})[j]$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

Hybrid Experiments. To prove the theorem, it now remains to show that the distribution of the output of this simulator is computationally indistinguishable from a tuple derived

by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ that follow Algorithm 7. We do so by considering the following list of hybrid experiments:

- $\mathbf{Hybrid}_0$: This is the distribution that is derived as the outcome of Sim (Algorithm 9) over all the internal randomness used by the simulator.

$$\mathbf{Hybrid}_0 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d})\leftarrow\mathsf{Sim}(1^\kappa,\mathbf{C},\mathbf{C}(\mathbf{x})),\kappa\in\mathbb{N}}$$

- $\mathbf{Hybrid}_1$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a random oracle $\mathcal{R}$, a PPT algorithm $\mathsf{Sim}^{\mathcal{R}}$ operates exactly as Algorithm 9 except Step 13. Instead, in this experiment, the values $M_A, M'_A, M_B, M'_B, M_{AB}, M'_{AB}$ are derived as,

$$\begin{aligned}
M_A &= \frac{\kappa}{2}\text{-bit prefix of } \mathcal{R}(\mathsf{L}_A^0, k_g^0, 0)\\
M'_A &= \frac{\kappa}{2}\text{-bit prefix of } \mathcal{R}(\mathsf{L}_A^0, k_g^0, 1)\\
M_B &= \frac{\kappa}{2}\text{-bit prefix of } \mathcal{R}(\mathsf{L}_B^0, k_g^1, 0)\\
M'_B &= \frac{\kappa}{2}\text{-bit prefix of } \mathcal{R}(\mathsf{L}_B^0, k_g^1, 1)\\
M_{AB} &= \frac{\kappa}{2}\text{-bit suffix of } \mathcal{R}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, k_g^2, 0)\\
M'_{AB} &= \frac{\kappa}{2}\text{-bit suffix of } \mathcal{R}(\mathsf{L}_A^0 \oplus \mathsf{L}_B^0, k_g^2, 1)
\end{aligned}$$

$$\mathbf{Hybrid}_1 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d})\leftarrow\mathsf{Sim}^{\mathcal{R}}(1^\kappa,\mathbf{C},\mathbf{C}(\mathbf{x})),\kappa\in\mathbb{N}}$$

  Due to the nature of the outputs of the random oracle, we have that this distribution is identical to the former one $\mathbf{Hybrid}_0 \equiv \mathbf{Hybrid}_1$.
- $\mathbf{Hybrid}_2$: This is a distribution that is derived as an outcome of a hybrid experiment where, first a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled. Then given access to a random oracle $\mathcal{R}$, a PPT algorithm $\mathsf{G}^{\mathcal{R},\mathsf{H}}$ operates as given in Algorithm 10. Unlike in the previous experiment, G receives the circuit input $\mathbf{x}$ as input and uses this to derive the values of all the active labels. Overall, this hybrid experiment differs from the previous one only in that the labeling of the active labels is no longer designated as 0, but it carries the same values as in the real execution of the garbling evaluation.

$$\mathbf{Hybrid}_2 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d})\leftarrow\mathsf{G}^{\mathcal{R},\mathsf{H}}(1^\kappa,\mathbf{C},\mathbf{x}),\kappa\in\mathbb{N}}$$

  As such, this distribution is identical to the former one since the (re-)labeling of the active wire labels is never visible to the adversary $\mathbf{Hybrid}_1 \equiv \mathbf{Hybrid}_2$.
- $\mathbf{Hybrid}_3$: Here first a weak CCR secure hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled along with $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ sampled uniformly at random. Then $\mathcal{C}_{\mathsf{H},\Delta}$ is the weak CCR oracle that outputs, $\mathcal{C}_{\mathsf{H},\Delta}(x,i,b) = \mathsf{H}(x \oplus \Delta, i) \oplus b\Delta$. The distribution $\mathbf{Hybrid}_3$ is derived as an outcome of a hybrid experiment where, given access to an oracle $\mathcal{C}_{\mathsf{H},\Delta}$, a PPT algorithm $\mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta},\mathsf{H}}$ operates as given in Algorithm 10. G receives the circuit input $\mathbf{x}$ as input and uses this to derive the values of all the active labels.

$$\mathbf{Hybrid}_3 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d})\leftarrow\mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta},\mathsf{H}}(1^\kappa,\mathbf{C},\mathbf{x}),\kappa\in\mathbb{N}}$$

  This hybrid experiment differs from the previous only in Step 13 in that the oracle query used to derive the values $M_A, M'_A, M_B, M'_B, M_{AB}, M'_{AB}$ are no longer made to

the random oracle $\mathcal{R}$, but to $\mathcal{C}_{\mathsf{H},\Delta}$. We show later that this hybrid distribution can be shown as computationally indistinguishable from the previous by reduction to the weak CCR security of the hash function family $\mathbf{Hybrid}_2 \overset{c}{\approx} \mathbf{Hybrid}_3$.

Algorithm 10 Hybrid Experiment $\mathsf{G}^{\mathcal{O},\mathsf{H}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}^{\mathcal{O},\mathsf{H}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$ and let $a, b \in \{0,1\}$ be the active input values
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^{\mathsf{XOR}(a,b)} = \mathsf{L}_A^a \oplus \mathsf{L}_B^b$
    else
      $k_g^0 = 3g - 2$, $k_g^1 = 3g - 1$, $k_g^2 = 3g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^a)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^b)$
      derive $\mathbf{T} \leftarrow \mathsf{AND}(p_a, p_b)$ and sample $\mathbf{R}, \mathbf{r} \leftarrow \mathsf{SampleR}(\mathbf{T})$
      compute $\mathbf{z} \parallel \begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = \mathbf{V}^{-1}(\mathbf{r} \parallel (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathsf{L}_A^a\\ \mathsf{L}_B^b\\ \Delta\end{bmatrix} \oplus \mathbf{MH})$
        where $(\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$ is computed as:
- use hash calls $\mathsf{H}(\mathsf{L}_A^a, k_g^0)$, $\mathsf{H}(\mathsf{L}_B^b, k_g^1)$ and $\mathsf{H}(\mathsf{L}_A^a \oplus \mathsf{L}_B^b, k_g^2)$ as indicated in $\mathbf{MH}$
- for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^{\neg a}, k_g^0)$,
  - letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_A^a, k_g^0, x)$
- for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_B^{\neg b}, k_g^1)$,
  - letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_B^b, k_g^1, x)$
- for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^a \oplus \mathsf{L}_B^{\neg b}, k_g^2)$,
  - letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_A^a \oplus \mathsf{L}_B^b, k_g^2, x)$

    set $\mathsf{L}_C^c = \mathbf{C}[0]||\mathbf{C}[1]$
    set $\mathbf{F}[g] = \mathbf{G}$
  end if
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^{\mathbf{C}(\mathbf{x})[j]}) \oplus \mathbf{C}(\mathbf{x})[j]$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

– $\mathbf{Hybrid}_4$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a weak CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ as above, a PPT algorithm $\mathsf{G}_*^{\mathcal{C}_{\mathsf{H},\Delta}}$ operates as given in Algorithm 11. This differs from the previous hybrid experiment only in that the lines marked in blue additionally exist in Algorithm 11, where they were not present in Algorithm 10.

$$\mathbf{Hybrid}_4 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}_*^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa, \mathbf{C}, \mathbf{x}), \kappa \in \mathbb{N}}$$

As the extra lines executed in this hybrid experiment do not affect the output distribution, it follows that this distribution is identical to the previous one $\mathbf{Hybrid}_3 \equiv \mathbf{Hybrid}_4$.

Algorithm 11 Hybrid Experiment $\mathsf{G}_*^{\mathcal{O}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}_*^{\mathcal{O}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and set $\mathcal{O} = \mathcal{C}_{\mathsf{H},\Delta}$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$ and set $\mathsf{L}_i^{\neg\mathbf{x}[i]} = \mathsf{L}_i^{\mathbf{x}[i]} \oplus \Delta$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$ and let $a, b \in \{0,1\}$ be the active input values
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C^{c=\mathsf{XOR}(a,b)} = \mathsf{L}_A^a \oplus \mathsf{L}_B^b$
    else
      $k_g^0 = 3g - 2$, $k_g^1 = 3g - 1$, $k_g^2 = 3g$, $p_a = \mathsf{lsb}(\mathsf{L}_A^a)$, $p_b = \mathsf{lsb}(\mathsf{L}_B^b)$
      derive $\mathbf{T} \leftarrow \mathsf{AND}(p_a, p_b)$ and sample $\mathbf{R}, \mathbf{r} \leftarrow \mathsf{SampleR}(\mathbf{T})$
      compute $\mathbf{z} \parallel \begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = \mathbf{V}^{-1}(\mathbf{r} \parallel (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathsf{L}_A^a\\ \mathsf{L}_B^b\\ \Delta\end{bmatrix} \oplus \mathbf{MH})$
        where $(\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$ is computed as:
        - use hash calls $\mathsf{H}(\mathsf{L}_A^a, k_g^0)$, $\mathsf{H}(\mathsf{L}_B^b, k_g^1)$ and $\mathsf{H}(\mathsf{L}_A^a \oplus \mathsf{L}_B^b, k_g^2)$ as indicated in $\mathbf{MH}$
        - for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^{\neg a}, k_g^0)$,
          * letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_A^a, k_g^0, x)$
        - for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_B^{\neg b}, k_g^1)$,
          * letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_B^b, k_g^1, x)$
        - for each entry in $\mathbf{MH}$ activating $\mathsf{H}(\mathsf{L}_A^a \oplus \mathsf{L}_B^{\neg b}, k_g^2)$,
          * letting x be the entry for $\mathbf{\Delta}$ in $(\mathbf{R} \oplus \mathbf{T})$, use $\mathcal{O}(\mathsf{L}_A^a \oplus \mathsf{L}_B^b, k_g^2, x)$
      set $\mathsf{L}_C^c = \mathbf{C}[0]||\mathbf{C}[1]$ and $\mathbf{F}[g] = \mathbf{G}$
    end if
    set $\mathsf{L}_C^{\neg c} = \mathsf{L}_C^c \oplus \Delta$
  end for
  for each $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \mathsf{lsb}(\mathsf{L}_j^{\mathbf{C}(\mathbf{x})[j]}) \oplus \mathbf{C}(\mathbf{x})[j]$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

– $\mathbf{Hybrid}_5$: This is the final hybrid experiment where the distribution is derived exactly as a tuple derived by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ in Algorithm 7.

$$\mathbf{Hybrid}_5 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),(\Delta,\mathbf{e}),\mathbf{d}) \leftarrow \mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^\kappa,\mathbf{C}), \mathbf{X} = \mathsf{GS}^{\mathcal{H}}.\mathsf{En}((\Delta,\mathbf{e}),\mathbf{x}), \kappa \in \mathbb{N}}$$

This differs from the previous experiment in that, here, for each wire, instead of first calculating the active label and then deriving the inactive label with respect to it, the

0-label is calculated (irrespective of whether it is the active label) and the 1-label is derived. Note that in this distribution and the previous, the algebraic relationships between each element derived remain the same. As such, this distribution is identical to the previous one $\mathbf{Hybrid}_4 \equiv \mathbf{Hybrid}_5$.

Security Reduction to Weak CCR. To prove the theorem, it only remains now to show that $\mathbf{Hybrid}_2 \overset{c}{\approx} \mathbf{Hybrid}_3$. We do so by demonstrating a security reduction to the weak CCR security property (Definition 7) of the underlying hash function family $\mathcal{H}$ used in the garbling scheme. Let D_{GS} be a PPT distinguisher that can distinguish between the hybrid distributions $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ with non-negligible advantage ϵ. If such a distinguisher exists, we show that it can be used in a black-box way by a PPT adversary A to win the weak CCR security game (Definition 7) for the hash function family $\mathcal{H}$ with non-negligible advantage as follows:

- The adversary A receives a circuit $\mathbf{C}$ and input $\mathbf{x}$ from D_{GS} that expects to receive a tuple of the form $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ in return.
- **Formulating a Query Strategy:** Given $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ with $\mathsf{q}_{\mathsf{AND}}$ AND gates, and input $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$, the adversary A designs a PPT algorithm with the following interface,
$$\mathcal{Q} = \{(x_j, j, b_j, y_j)\}_{j \in [3\mathsf{q}_{\mathsf{AND}}]} = \mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}(1^\kappa; r)$$
where the strategy accepts randomness of the form $r \leftarrow \{0,1\}^{\mathsf{n}\kappa}$. This is detailed in Algorithm 12. This is a randomized algorithm that is given to the challenger $\mathcal{C}$ for the weak CCR security game.
- The challenger $\mathcal{C}$ accepts this $\mathsf{QueryStrategy}^{\mathcal{O}}$ and operates as implicit in Definition 7. That is, it samples a bit $c \leftarrow \{0,1\}$ and if $c = 0$, it samples randomness r and executes $\mathsf{QueryStrategy}^{\mathcal{R}}_{\mathbf{C},\mathbf{x}}$. Otherwise, it samples r and Δ, and executes $\mathsf{QueryStrategy}^{\mathcal{C}_{\mathsf{H},\Delta}}_{\mathbf{C},\mathbf{x}}$. It gives a tuple $(\mathcal{Q}, \mathsf{H})$ to A.
- Given the set of query responses $\mathcal{Q}$ and the hash algorithm H, the adversary A can now execute Algorithm 10 with active input labels as indicated in the queries in $\mathcal{Q}$ and using the items in $\mathcal{Q}$ where ever oracle queries are required. The output of this is a tuple $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ which is distributed as $\mathbf{Hybrid}_2$ if the bit chosen by the challenger is $c = 0$, and is distributed as $\mathbf{Hybrid}_3$ otherwise.
- A gives $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ to D_{GS} and outputs whatever it outputs.

Note that in the above game, A has the same advantage as that of D_{GS}, which is non-negligible. However, since H is sampled from a weak CCR secure hash function family $\mathcal{H}_\kappa$, it follows that no such A can exist and therefore no such D_{GS} can exist. This completes the proof.

D Proof of Theorem 7

In this section, we first begin with fully describing the garbling scheme from [33], adapted to using a hash function H that is sampled from a weak CCR secure hash function family $\mathcal{H}$ (Definition 7). [33] defines their garbling scheme for switch systems. These are circuit-like objects establishing constraints over arithmetic wires that each hold values modulo 2^k, for various *widths* k. Although [33] defines their garbling scheme for general oblivious switch systems for arithmetic circuits, for simplicity, we restrict ourselves to the Boolean setting (i.e. the *max-width* of the switch system is 1) wherein the garbling scheme uses a hash

Algorithm 12 Query Strategy $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}$ for $\mathsf{GS}^{\mathcal{H}}$ adapted from [52]

procedure $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{C},\mathbf{x}}(1^\kappa; r \leftarrow \{0,1\}^{n\kappa})$
  initialize $\mathcal{Q} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathsf{L}_i = r[(i-1)\kappa + 1 : i\kappa] \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ in topological order **do**
    parse gate $g = (A, B, C, f_g)$
    if $f_g == \mathsf{XOR}$ **then**
      set $\mathsf{L}_C = \mathsf{L}_A \oplus \mathsf{L}_B$
    else
      $k_g^0 = 3g - 2$, $k_g^1 = 3g - 1$, $k_g^2 = 3g$, $p_a = \mathsf{lsb}(\mathsf{L}_A)$, $p_b = \mathsf{lsb}(\mathsf{L}_B)$
      derive $\mathbf{T} \leftarrow \mathsf{AND}(p_a, p_b)$ and sample $\mathbf{R}, \mathbf{r} \leftarrow \mathsf{SampleR}(\mathbf{T})$
      compute $\mathbf{z} \parallel \begin{bmatrix}\mathbf{C}\\ \mathbf{G}\end{bmatrix} = \mathbf{V}^{-1}(\mathbf{r} \parallel (\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathsf{L}_A\\ \mathsf{L}_B\\ \Delta\end{bmatrix} \oplus \mathbf{MH})$, where within $(\mathbf{R} \oplus \mathbf{T}) \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \Delta\end{bmatrix} \oplus \mathbf{MH}$,
        if $M_A = \mathcal{O}(\mathsf{L}_A, k_g^0, 0)$ is activated, set $\mathcal{Q}[k_g^0] = (\mathsf{L}_A, k_g^0, 0, M_A)$
        if $M'_A = \mathcal{O}(\mathsf{L}_A, k_g^0, 1)$ is activated, set $\mathcal{Q}[k_g^0] = (\mathsf{L}_A, k_g^0, 1, M'_A)$
        if $M_B = \mathcal{O}(\mathsf{L}_B, k_g^1, 0)$ is activated, set $\mathcal{Q}[k_g^1] = (\mathsf{L}_B, k_g^1, 0, M_B)$
        if $M'_B = \mathcal{O}(\mathsf{L}_B, k_g^1, 1)$ is activated, set $\mathcal{Q}[k_g^1] = (\mathsf{L}_B, k_g^1, 1, M'_B)$
        if $M_{AB} = \mathcal{O}(\mathsf{L}_A \oplus \mathsf{L}_B, k_g^2, 0)$ activates, $\mathcal{Q}[k_g^2] = (\mathsf{L}_A \oplus \mathsf{L}_B, k_g^2, 0, M_{AB})$
        if $M'_{AB} = \mathcal{O}(\mathsf{L}_A \oplus \mathsf{L}_B, k_g^2, 1)$ activates, $\mathcal{Q}[k_g^2] = (\mathsf{L}_A \oplus \mathsf{L}_B, k_g^2, 1, M'_{AB})$
      set $\mathsf{L}^c_C = \mathbf{C}[0] || \mathbf{C}[1]$ and $\mathbf{F}[g] = \mathbf{G}$
    end if
  end for
  return $\mathcal{Q}$
end procedure

function with interface $\mathsf{H} : \{0,1\}^\kappa \times \mathbb{N} \to \{0,1\}^\kappa$ that is compatible with Definition 7. We refer the reader to Remark 5 for a note on how this definition and its realization can extend beyond the case of Boolean inputs and outputs.

Oblivious Switch Systems and Notation. A switch system is a model of computation that generalizes circuits. We re-state its formal definition from [33] below.

Definition 10. *A **switch system** is a system of constraints on wires (i.e., constrained variables) holding values over moduli 2^k for various k. The system is defined in terms of gates. Non-input wires in the system are initially not set (i.e., have no value), and as the system runs, wires become set according to the rules of each gate. The types of gates are as follows:*

– *A **switch** takes as input a binary control wire $\mathsf{ctrl} \in \mathbb{Z}_2$ and data wire $x \in \mathbb{Z}_{2^k}$. The gate outputs data wire $y \in \mathbb{Z}_{2^k}$, and it establishes the following implication constraint:*

$$\mathsf{ctrl} = 0 \to x = y$$

The output of a switch is denoted by writing $x \vdash \mathsf{ctrl}$. Switches are bidirectional in the sense that the system ensures that if $\mathsf{ctrl} = 0$, then $x = y$, regardless of which data wire is set first.

– *A **join** takes input wires $x, y \in \mathbb{Z}_{2^k}$ and establishes an equality constraint:*

$$x = y$$

This is denoted by $x \bowtie y$. Joins are bidirectional in the sense that the system ensures that $x = y$, regardless of which wire is set first.

– *An **affine gate** is parameterized by an affine map $f : \mathbb{Z}_{2^k}^{\mathsf{in}} \to \mathbb{Z}_{2^k}^{\mathsf{out}}$, where $\mathsf{in}, \mathsf{out} \in \mathbb{N}$. It takes as input a vector of wires $\mathbf{x} \in \mathbb{Z}_{2^k}^{\mathsf{in}}$, and it outputs a vector of wires $\mathbf{y} \in \mathbb{Z}_{2^k}^{\mathsf{out}}$. The gate establishes the following constraint:*

$$f(\mathbf{x}) = \mathbf{y}$$

Affine gates are denoted by writing affine constraints of wires and are multi-directional in the sense that the system uses set wires to solve for the unset wires.

– *A **modulus gate** takes as input a wire $x \in \mathbb{Z}_{2^{k+c}}$ for arbitrary c, k. It outputs a wire $y \in \mathbb{Z}_{2^k}$, and establishes the following constraint:*

$$y = x \mod 2^k$$

This is a one-directional gate: the system uses x to solve for y.

– *A **division gate** takes as input a wire $x \in \mathbb{Z}_{2^{k+c}}$ where it is guaranteed that 2^c divides x. It outputs a wire $y \in \mathbb{Z}_{2^k}$, and it establishes the following constraint:*

$$y = \frac{x}{2^c} \mod 2^k$$

This is a one-directional gate: the system uses x to solve for y.

*Each gate has a unique identifier g and a switch system S has **input wires** and output wires. $S(\mathbf{x})$ denotes the values on the output wires after running with input wires $\mathbf{x}$.*

Informally, one can think of a switch system as a collection of sets of linear equations. Depending on which variables in these equations have known values (are set), each set of linear equations may have either infinite, unique or no solution. The variables that are initially set (before the evaluation starts) are inputs $\mathbf{x}$ and a set of *controls* dictate the order in which the equations are evaluated with outputs of one equation possibly feeding into the input wires of another. This ordering is indicated by the gate IDs and formalized below.

Definition 11. *Let S be a switch system and let $\mathbf{x}$ be an assignment of input wires. The **controls of S on $\mathbf{x}$**, denoted $\mathsf{ctrl}(S, \mathbf{x}) \in \mathbb{Z}_2^*$, is the set of all switch control wire values, each labeled by its gate ID g.*

Informally, a switch system as given above is legal if the assignments of values to wires, starting from the inputs and then as dictated by the controls, lead to a unique solution of this system.

Definition 12. *A switch system S is **legal** if for any input $\mathbf{x}$ there exists only one assignment of circuit wires that satisfies the gate constraints, i.e., wire values are a function of the input wires.*

Definition 13. *Consider a switch system S, and let $x \in \mathbb{Z}_{2^k}$ denote a wire modulo 2^k in S. The wire x has **width** k, denoted by $\mathsf{width}(x) = k$. For a join $x \bowtie y$, the width is the same as the width of x (and y):*

$$\mathsf{width}(x \bowtie y) = \mathsf{width}(x) = \mathsf{width}(y)$$

*The **join width** of S is the sum of the widths of all join gates, denoted as $\mathsf{Jwidth}(S)$.*

[33] shows that for any Boolean circuit $\mathbf{C}$ there exits a legal switch system S as above such that:

- $|S| = O(|C|)$ and $\mathsf{Jwidth}(S) = O(|C|)$,
- for all inputs $\mathbf{x}$, $S(\mathbf{x}) = C(\mathbf{x})$.

The notion of a switch system can also be defined as randomized (in contrast with deterministic) where part of the input wires are designated to accept randomness that originates from a predetermined distribution. This is formalized in the definition below.

Definition 14. *A **randomized switch system** is a pair consisting of a switch system S and a distribution D. The execution $(S, D)(\mathbf{x})$ of a randomized switch system on input $\mathbf{x}$ is defined by randomly sampling $\mathbf{r} \leftarrow D$, then running $S(\mathbf{x}; \mathbf{r})$.*

Definition 15. *A family of legal randomized switch systems (S_κ, D_κ) for $\kappa \in \mathbb{N}$ is **oblivious** if the distribution of controls of (S_κ, D_κ) can be simulated. That is, there exists a PPT simulator $\mathsf{Sim}_{\mathsf{ctrl}}$ such that for all inputs $\mathbf{x}$,*

$$\mathsf{Sim}_{\mathsf{ctrl}}(1^\kappa) \overset{s}{\approx} \{\mathsf{ctrl}(S_\kappa, (\mathbf{x}; \mathbf{r}))|\mathbf{r} \leftarrow D_\kappa\}$$

where $\overset{s}{\approx}$ denotes that the distributions are statistically close w.r.t. κ.

[33] shows that for any Boolean circuit $\mathbf{C}$, there exits an oblivious switch system (S, D) such that:

- $|S| = O(|C|)$ and $\mathsf{Jwidth}(S) = O(|C|)$,
- for all inputs $\mathbf{x}$, $(S, D)(\mathbf{x}) = C(\mathbf{x})$.

Algorithms 14–15 detail the garbling scheme GS^{H} in [33] for such oblivious switch systems and Algorithm 13 contains a simplification of the garbling and encoding algorithms for Boolean wires (width $k = 1$). In the simplified scheme, each wire w has two associated labels $\mathsf{L}^0_w, \mathsf{L}^1_w \in \{0,1\}^\kappa$. For a vector $\mathbf{x} \in \{0,1\}^n$, we denote by $\vec{\mathsf{L}^0_{\mathbf{x}}} = \{\mathsf{L}^0_{\mathbf{x}[i]}\}_{i\in[n]}$ and $\vec{\mathsf{L}^1_{\mathbf{x}}} = \{\mathsf{L}^1_{\mathbf{x}[i]}\}_{i\in[n]}$ the vector of labels for the 0 and 1 value for each bit position in $[n]$. Further, the modulus and division gates reduce to equality for the Boolean wire system.

We go on to show that this scheme (for Boolean labels) can be proven selectively secure in the presence of PPT adversaries as long as the underlying hash function H is sampled uniformly at random from a weakly CCR secure family of hash function $\mathcal{H}$ (Definition 7) during the garbling procedure.

Remark 5 (Weak CCR security for hash families with general domains). Definition 7 of weak CCR secure hash function families and the proof of Theorem 8 can be directly extended to families of hash functions whose members have the form $\mathsf{H} : \mathbb{Z}^\kappa_{2^k} \times \mathbb{N} \to \mathbb{Z}^\kappa_{2^k}$ and offsets are defined as $\Delta \leftarrow \mathbb{Z}^{\kappa-1}_{2^k}||1$ where these domains are vectors of elements from a larger integer space $\mathbb{Z}_{2^k}$ than Boolean values. Further, all operations within the oracle $\mathcal{C}_{\mathsf{H},\Delta}$ would now be over $\mathbb{Z}_{2^k}$ (i.e., element-wise addition and multiplication over the field, instead of Boolean AND and XOR). This resulting hash function family would suffice for instantiating the arithmetic garbling scheme $\mathsf{GS}^{\mathcal{H}}$ in [33] in its complete generality. The construction of such hash function families would be the same as Construction 1, except the underlying PPRF would have input and output domain $\mathbb{Z}^\kappa_{2^k}$ respectively. The proof for weak CCR security would follow from the same set of hybrid experiments, with the exception that this change of PPRF domains and arithmetic operation over the new domain needs to be reflected.

D.1 Proof

Theorem 7 *Let $\mathcal{H} = \{\mathcal{H}_\kappa\}_{\kappa\in\mathbb{N}}$ be a wCCR secure hash function family ensemble (Definition 7) and $\mathsf{GS}^{\mathcal{H}} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be the garbling scheme as defined in Algorithms 14 and 15 where, for security parameter κ, in the beginning of Gb a hash function H is sampled uniformly at random from $\mathcal{H}_\kappa$ and output together with the garbled circuit $\mathbf{F}$. Then $\mathsf{GS}^{\mathcal{H}}$ satisfies selective privacy by simulation (PRIV-SIM – Definition 6).*

To prove the theorem, we show that if there existed a PPT distinguisher D that has non-negligible advantage in the PRIV-SIM security game (Definition 6), this can be used in a black-box way by a PPT adversary A to gain non-negligible advantage in the security game for weak CCR (Definition 7). To show this, let us first describe the PPT simulator Sim that the challenger in PRIV-SIM invokes.

Simulator for $\mathsf{GS}^{\mathcal{H}}$. The simulator for the garbling is a PPT algorithm $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\kappa, (S, D), S(\mathbf{x}))$ that needs to output the selective privacy challenge tuple created only using the circuit and the function output. Note that the garbling algorithm $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ garbles an oblivious switch system (S, D) gate by gate in topological order and Sim needs to output a tuple that contains values that are indistinguishable from those derived from the real garbling scheme on the system and input.

Let $\mathbf{X}$ denote the set of *active input labels* for the garbling $\mathbf{F}, \mathbf{d}$. That is, for a real garbling $(\mathbf{F}, \mathbf{d})$, this is the input encoding $\mathbf{X} = \mathsf{GS}^{\mathcal{H}}.\mathsf{En}((\boldsymbol{\Delta}, \mathbf{e}), \mathbf{x})$. For each wire in the switch system (S, D), we refer to the label derived from evaluating $\mathbf{F}$ using $\mathbf{X}$ as the *active label*. In the simulation, for each wire we assign the label representing the 0-value as the active label. Note that only the garbling of the switch gates requires calls to the hash function and this is one hash call of the form either $\mathcal{C}_{\mathsf{H},\Delta}(\mathsf{L}_{\mathsf{ctrl}}, g, 0)$ (when the control bit $\mathsf{ctrl} = 1$) or $\mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$ (when $\mathsf{ctrl} = 0$). This allows representing the garbling in terms of the active labels and the weak CCR hash oracle $\mathsf{H}(x \oplus \Delta, i) \oplus b\Delta = \mathcal{C}_{\mathsf{H},\Delta}(x, i, b)$ only. Further, note that throughout the garbling algorithm, the gate indices $g \in [\mathsf{q}]$ act as a counter for the oracle queries made, rendering the set of garbling oracle queries and responses as a legal query strategy satisfying weak CCR (Definition 7).

Given this observation, the simulator Sim works as follows: first, a set of control bits are simulated using a call to $\mathsf{ctrl} \leftarrow \mathsf{Sim}_{\mathsf{ctrl}}(1^\kappa)$ that exists for every oblivious switch system. Then a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled at random from the family of weak CCR secure hash functions. For each input wire, an active label is sampled as in the real garbling. Then the garbling algorithm is executed with each instance of weak CCR oracle output being replaced by a κ-bit value sampled uniformly at random, as required in the construction. This process also derives the active wire label for all other wires in the system, including the output wires. Finally, the output decoding information is set such that the active output labels map to the function output as required. Algorithm 16 formalizes the actions of the simulator.

Hybrid Experiments. To prove the theorem, it now remains to show that the distribution of the output of this simulator is computationally indistinguishable from a tuple derived by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ in Algorithm 14. We do so by considering the following list of hybrid experiments:

– $\mathbf{Hybrid}_0$: This is the distribution that is derived as the outcome of Sim (Algorithm 16) over all the internal randomness used by the simulator.

$$\mathbf{Hybrid}_0 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d})\leftarrow\mathsf{Sim}(1^\kappa,(S,D),S(\mathbf{x})),\kappa\in\mathbb{N}}$$

Algorithm 13 Garbling Scheme GS^{H} for Switch Systems over Boolean Domain

procedure $\mathsf{Gb}(1^\kappa, (S, D))$
  initialize $\mathbf{F} = []$, $\mathbf{e} = []$ and $\mathbf{d} = []$
  sample $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$
  sample $\mathbf{r} \leftarrow D_\kappa$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$ and set $\mathsf{L}_i^1 = \mathsf{L}_i^0 \oplus \Delta$
    set $\mathbf{e}[i] = \mathsf{L}_i^0$
  end for
  for every randomness wire i set to $\mathbf{r}[i] \in \{0,1\}$ **do**
    set $\mathsf{L}_i^0 = \mathbf{r}[i] \cdot \Delta \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      $\mathsf{L}_y^0 = \mathsf{L}_x^0 \oplus \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}^0, g)$
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}_{\mathsf{ctrl}}^0)$
    end if
    if gate g is a join gate: $x \bowtie y$ **then**
      $\mathbf{F}[g] = \mathsf{L}_y^0 \oplus \mathsf{L}_x^0$
    end if
    if gate g is an affine gate: $\mathbf{y} = f(\mathbf{x})$ **then**
      $\mathsf{L}_{\mathbf{y}}^{\vec{0}} = f(\mathsf{L}_{\mathbf{x}}^{\vec{0}})$
    end if
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j][0] = \mathsf{H}(\mathsf{L}_j^0, \mathsf{q} + j)$
    set $\mathbf{d}[j][1] = \mathsf{H}(\mathsf{L}_j^0 \oplus \Delta, \mathsf{q} + j)$
  end for
  return $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$
end procedure

procedure $\mathsf{En}((\Delta, \mathbf{e}), \mathbf{x})$
  initialize $\mathbf{X} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathbf{X}[i] = \mathbf{e}[i] \oplus \mathbf{x}[i]\Delta$
  end for
  Return $\mathbf{X}$
end procedure

– $\mathbf{Hybrid}_1$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a random oracle $\mathcal{R}$, a PPT algorithm $\mathsf{Sim}^{\mathcal{R}}$ operates exactly as (Algorithm 16) except in Step 15 and Step 22. Instead, in this experiment, the values L_y^0 (resp. M) are derived as $\mathsf{L}_x^0 \oplus \mathcal{R}(\mathsf{L}_{\mathsf{ctrl}}, g, x \oplus y)$ (resp. $\mathcal{R}(\mathsf{L}_j^0, \mathsf{q} + j, 0)$)

$$\mathbf{Hybrid}_1 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{Sim}^{\mathcal{R}}(1^\kappa,(S,D),S(\mathbf{x})),\kappa \in \mathbb{N}}$$

  Due to the nature of the outputs of the random oracle, we have that this distribution is identical to the former one $\mathbf{Hybrid}_0 \equiv \mathbf{Hybrid}_1$.
– $\mathbf{Hybrid}_2$: This is a distribution that is derived as an outcome of a hybrid experiment where, first a hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled. Then given access to a random oracle

Algorithm 14 Garbling Scheme GS^{H} for Switch Systems [33]

procedure $\mathsf{Gb}(1^\kappa, (S, D))$
  initialize $\mathbf{F} = []$, $\mathbf{e} = []$ and $\mathbf{d} = []$
  sample $\Delta \leftarrow \mathbb{Z}^{\kappa-1}_{2^{k_{\mathbf{max}}}} || 1$
  sample $\mathbf{r} \leftarrow D_\kappa$
  for every $i \in [\mathsf{n}]$ with width k **do**
    sample $\mathsf{L}^0_i \leftarrow \mathbb{Z}^\kappa_{2^k}$
    set $\mathbf{e}[i] = \mathsf{L}^0_i$
  end for
  for every randomness wire i set to $[\mathbf{r}[i]] \in [2^k]$ **do**
    set $\mathsf{L}^0_i = [(0 - \mathbf{r}[i]) \cdot \Delta] \mod 2^k$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      $\mathsf{L}^0_y = \mathsf{L}^0_x + \mathsf{H}(\mathsf{L}^0_{\mathsf{ctrl}}, g)$
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}^0_{\mathsf{ctrl}})$
    end if
    if gate g is a join gate: $x \bowtie y$ **then**
      $\mathbf{F}[g] = \mathsf{L}^0_y - \mathsf{L}^0_x$
    end if
    if gate g is an affine gate: $\mathbf{y} = f(\mathbf{x})$ **then**
      $\vec{\mathsf{L}}^0_{\mathbf{y}} = f(\vec{\mathsf{L}}^0_{\mathbf{x}})$
    end if
    if gate g is a modulus gate: $y = x \mod 2^k$ **then**
      $\mathsf{L}^0_y = \mathsf{L}^0_x \mod 2^k$
    end if
    if gate g is a division gate: $y = \frac{x}{2^c} \mod 2^k$ **then**
      $\mathsf{L}^0_y = \frac{\mathsf{L}^0_x}{2^c} \mod 2^k$
    end if
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j][0] = \mathsf{H}(\mathsf{L}^0_j, \mathsf{q} + j)$
    set $\mathbf{d}[j][1] = \mathsf{H}(\mathsf{L}^0_j \oplus \Delta, \mathsf{q} + j)$
  end for
  return $(\mathbf{F}, (\Delta, \mathbf{e}), \mathbf{d})$
end procedure

procedure $\mathsf{En}((\Delta, \mathbf{e}), \mathbf{x})$
  initialize $\mathbf{X} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathbf{X}[i] = \mathbf{e}[i] \oplus \mathbf{x}[i]\Delta$
  end for
  Return $\mathbf{X}$
end procedure

$\mathcal{R}$, a PPT algorithm $\mathsf{G}^{\mathcal{R},\mathsf{H}}$ operates as given in Algorithm 17. Unlike in the previous experiment, G receives the switch system input $\mathbf{x}$ as input and samples $\mathbf{r} \leftarrow D$ and uses this to derive the values of all the active labels. This includes the control values (as opposed to them being derived from $\mathsf{Sim}_{\mathsf{ctrl}}$ as in the previous hybrid). Overall, this

Algorithm 15 Algorithms to Evaluate the Garbling

procedure $\mathsf{Ev}(\mathbf{F}, \mathbf{X})$
  initialize $\mathbf{Y} = []$
  for each gate $g \in [\mathsf{q}]$ in a topological order **do**
    use active input labels (that are set) to derive further labels
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      let L_x (resp. L_y) and $\mathsf{L}_{\mathsf{ctrl}}$ be set
      parse $\mathbf{F}[g] = \mathsf{lsb}_{\mathsf{ctrl}}$ and derive ctrl
      if $\mathsf{ctrl} = 0$ **then**
        compute $\mathsf{L}_y = \mathsf{L}_x + \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$ (resp. $\mathsf{L}_x = \mathsf{L}_y - \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$)
      end if
    end if
    if gate g is a join gate: $x \bowtie y$ **then**
      let L_x (resp. L_y) be set
      compute $\mathsf{L}_y = \mathbf{F}[g] + \mathsf{L}_x$ (resp. $\mathsf{L}_x = \mathbf{F}[g] - \mathsf{L}_y$)
    end if
    if gate g is an affine gate: $\mathbf{y} = f(\mathbf{x})$ **then**
      let $\vec{\mathsf{L}_{\mathbf{x}}}$ be set
      compute $\vec{\mathsf{L}_{\mathbf{y}}} = f(\vec{\mathsf{L}_{\mathbf{x}}})$ by solving the set of equations
    end if
    if gate g is a modulus gate: $y = x \mod 2^k$ **then**
      drop the MSBs of L_x until $\mathsf{L}_y = \mathsf{L}_x \mod 2^k$
    end if
    if gate g is a division gate: $y = \frac{x}{2^c} \mod 2^k$ **then**
      drop the LSBs of L_x until $\mathsf{L}_y = \frac{\mathsf{L}_x}{2^c} \mod 2^k$
    end if
    if C is a circuit output wire **then**
      $\mathbf{Y}[C] = \mathsf{L}_C$
    end if
  end for
  return $\mathbf{Y}$
end procedure

procedure $\mathsf{De}(\mathbf{Y}, \mathbf{d})$
  initialize $\mathbf{y} = []$
  for $j \in [\mathsf{m}]$ **do**
    $\mathbf{y}[j] = \begin{cases} 0 & \text{if } \mathsf{H}(\mathbf{Y}[j], \mathsf{q}+j) = \mathbf{d}[j][0] \\ 1 & \text{if } \mathsf{H}(\mathbf{Y}[j], \mathsf{q}+j) = \mathbf{d}[j][1] \end{cases}$
  end for
  return $\mathbf{y}$
end procedure

hybrid experiment differs from the previous one only in that the labeling of the active labels is no longer designated as 0, but it carries the same values as in the real execution of the garbling evaluation.

$$\mathbf{Hybrid}_2 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}^{\mathcal{R},\mathsf{H}}(1^\kappa,(S,D),\mathbf{x}),\kappa\in\mathbb{N}}$$

Algorithm 16 Simulator Sim for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{Sim}(1^\kappa, (S, D), S(\mathbf{x}))$
  sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ and initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  $\mathsf{ctrl} \leftarrow \mathsf{Sim}_{\mathsf{ctrl}}(1^\kappa)$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^0 \leftarrow \{0,1\}^\kappa$ and set $\mathbf{X}[i] = \mathsf{L}_i^0$
  end for
  for every randomness wire i **do**
    set $\mathsf{L}_i^0 = 0^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      if $\mathsf{ctrl}[g] = 0$ **then**
        $\mathsf{L}_y^0 = \mathsf{L}_x^0 \oplus \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}^0, g)$
      else
        sample $\mathsf{L}_y^0 \leftarrow \{0,1\}^\kappa$
      end if
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}_{\mathsf{ctrl}}^0) \oplus \mathsf{ctrl}[g]$
    end if
    $\cdots$ ▷ join, affine, modulus and division gates garbled like in Gb
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j][S(\mathbf{x})[j]] = \mathsf{H}(\mathsf{L}_j^0, \mathsf{q} + j)$
    set $\mathbf{d}[j][\neg S(\mathbf{x})[j]] = M \leftarrow \{0,1\}^\kappa$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

As such, this distribution is identical to the former one since the (re-)labeling of the active wire labels is never visible to the adversary and the output of $\mathsf{Sim}_{\mathsf{ctrl}}$ is statistically close to that of a real assignment of the control wires $\mathbf{Hybrid}_1 \overset{\mathrm{s}}{\approx} \mathbf{Hybrid}_2$.

– $\mathbf{Hybrid}_3$: Here first a weak CCR secure hash function $\mathsf{H} \leftarrow \mathcal{H}_\kappa$ is sampled along with $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ sampled uniformly at random. Then $\mathcal{C}_{\mathsf{H},\Delta}$ is the weak CCR oracle that outputs, $\mathcal{C}_{\mathsf{H},\Delta}(x, i, b) = \mathsf{H}(x \oplus \Delta, i) \oplus b\Delta$. The distribution $\mathbf{Hybrid}_3$ is derived as an outcome of a hybrid experiment where, given access to an oracle $\mathcal{C}_{\mathsf{H},\Delta}$, a PPT algorithm $\mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta},\mathsf{H}}$ operates as given in Algorithm 17. G receives the switch system input $\mathbf{x}$ as input, samples randomness and uses this to derive the values of all the active labels.

$$\mathbf{Hybrid}_3 = \left\{(\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}^{\mathcal{C}_{\mathsf{H},\Delta}}(1^\kappa,(S,D),\mathbf{x}),\kappa \in \mathbb{N}}$$

This hybrid experiment differs from the previous only in Step 16 and Step 23 in that the oracle query used are no longer made to the random oracle $\mathcal{R}$, but to $\mathcal{C}_{\mathsf{H},\Delta}$. We show later that this hybrid distribution is computationally indistinguishable from the previous by reduction to the weak CCR security of the hash function family $\mathbf{Hybrid}_2 \overset{\mathrm{c}}{\approx} \mathbf{Hybrid}_3$.

– $\mathbf{Hybrid}_4$: This is a distribution that is derived as an outcome of a hybrid experiment where, given access to a weak CCR oracle $\mathcal{C}_{\mathsf{H},\Delta}$ as above, a PPT algorithm $\mathsf{G}_*^{\mathcal{C}_{\mathsf{H},\Delta}}$ operates as given in Algorithm 18. This differs from the previous hybrid experiment only in that the lines marked in blue additionally exist in Algorithm 18, where they were not present

Algorithm 17 Hybrid Experiment $\mathsf{G}^{\mathcal{O},\mathsf{H}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}^{\mathcal{O},\mathsf{H}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  sample $\mathbf{r} \leftarrow D_\kappa$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for every randomness wire i set to $\mathbf{r}[i] \in \{0,1\}$ **do**
    set $\mathsf{L}_i^{\mathbf{r}[i]} = \mathbf{r}[i] \cdot \Delta \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      if $\mathsf{ctrl}[g] = 0$ **then**
        $\mathsf{L}_y = \mathsf{L}_x \oplus \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$
      else
        compute $\mathsf{L}_y = \mathsf{L}_x \oplus \mathcal{O}(\mathsf{L}_{\mathsf{ctrl}}, g, x \oplus y)$
      end if
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}_{\mathsf{ctrl}}) \oplus \mathsf{ctrl}[g]$
    end if
    $\cdots$ ▷ join, affine, modulus and division gates garbled like in Gb
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \{S(\mathbf{x})[j] : \mathsf{H}(\mathsf{L}_j, \mathsf{q}+j),\ \neg S(\mathbf{x})[j] : M = \mathcal{O}(\mathsf{L}_j, \mathsf{q}+j, 0)\}$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

in Algorithm 17.

$$\mathbf{Hybrid}_4 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}) \leftarrow \mathsf{G}_*^{\mathcal{C}_{\mathsf{H}},\Delta}(1^\kappa,(S,D),\mathbf{x}),\kappa\in\mathbb{N}}$$

As the extra lines executed in this hybrid experiment do not affect the output distribution, it follows that this distribution is identical to the previous one $\mathbf{Hybrid}_3 \equiv \mathbf{Hybrid}_4$.

– $\mathbf{Hybrid}_5$: This is the final hybrid experiment where the distribution is derived exactly as a tuple derived by executing $\mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}$ and $\mathsf{GS}^{\mathcal{H}}.\mathsf{En}$ in Algorithm 14.

$$\mathbf{Hybrid}_5 = \left\{(\mathbf{F},\mathsf{H}),\mathbf{X},\mathbf{d}\right\}_{((\mathbf{F},\mathsf{H}),(\Delta,\mathbf{e}),\mathbf{d}) \leftarrow \mathsf{GS}^{\mathcal{H}}.\mathsf{Gb}(1^\kappa,(S,D)),\mathbf{X}=\mathsf{GS}^{\mathcal{H}}.\mathsf{En}((\Delta,\mathbf{e}),\mathbf{x}),\kappa\in\mathbb{N}}$$

This differs from the previous experiment in that, here, for each wire, instead of first calculating the active label and then deriving the inactive label with respect to it, the 0-label is calculated (irrespective of whether it is the active label) and the 1-label is derived. Note that in this distribution and the previous, the algebraic relationships between each element derived remain the same. As such, this distribution is identical to the previous one $\mathbf{Hybrid}_4 \equiv \mathbf{Hybrid}_5$.

Security Reduction to Weak CCR. To prove the theorem, it only remains now to show that $\mathbf{Hybrid}_2 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_3$. We do so by demonstrating a security reduction to the weak

Algorithm 18 Hybrid Experiment $\mathsf{G}_*^{\mathcal{O}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{G}_*^{\mathcal{O}}(1^\kappa, \mathbf{C}, \mathbf{x})$
  initialize $\mathbf{F} = []$, $\mathbf{X} = []$ and $\mathbf{d} = []$
  sample $\mathsf{H} \leftarrow \mathcal{H}_\kappa$, $\Delta \leftarrow \{0,1\}^{\kappa-1}||1$ and set $\mathcal{O} = \mathcal{C}_{\mathsf{H},\Delta}$
  sample $\mathbf{r} \leftarrow D_\kappa$
  for every $i \in [\mathsf{n}]$ **do**
    sample $\mathsf{L}_i^{\mathbf{x}[i]} \leftarrow \{0,1\}^\kappa$ and set $\mathsf{L}_i^{\neg\mathbf{x}[i]} = \mathsf{L}_i^{\mathbf{x}[i]} \oplus \Delta$
    set $\mathbf{X}[i] = \mathsf{L}_i^{\mathbf{x}[i]}$
  end for
  for every randomness wire i set to $\mathbf{r}[i] \in \{0,1\}$ **do**
    set $\mathsf{L}_i^{\mathbf{r}[i]} = \mathbf{r}[i] \cdot \Delta \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      if $\mathsf{ctrl}[g] = 0$ **then**
        $\mathsf{L}_y = \mathsf{L}_x \oplus \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$
      else
        compute $\mathsf{L}_y = \mathsf{L}_x \oplus \mathcal{O}(\mathsf{L}_{\mathsf{ctrl}}, g, x \oplus y)$
      end if
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}_{\mathsf{ctrl}}) \oplus \mathsf{ctrl}[g]$
    end if
    $\cdots$ ▷ join, affine, modulus and division gates garbled like in Gb
    set each inactive output label $\mathsf{L}'_y = \mathsf{L}_y \oplus \Delta$
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \{S(\mathbf{x})[j] : \mathsf{H}(\mathsf{L}_j, \mathsf{q}+j),\ \neg S(\mathbf{x})[j] : \mathcal{O}(\mathsf{L}_j, \mathsf{q}+j, 0)\}$
  end for
  return $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$
end procedure

CCR security property (Definition 7) of the underlying hash function family $\mathcal{H}$ used in the garbling scheme. Let D_{GS} be a PPT distinguisher that can distinguish between the hybrid distributions $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ with non-negligible advantage ϵ. If such a distinguisher exists, we show that it can be used in a black-box way by a PPT adversary A to win the weak CCR security game (Definition 7) for the hash function family $\mathcal{H}$ with non-negligible advantage as follows:

- The adversary A receives an oblivious switch system $(\mathbf{S}, \mathbf{D})$ and input $\mathbf{x}$ from D_{GS} that expects to receive a tuple of the form $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ in return.
- **Formulating a Query Strategy:** Given $\mathbf{S} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ for a Boolean circuit with $\mathsf{q}_{\mathsf{ctrl}}$ switch gates, randomness $\mathbf{r}$ sampled from distribution $\mathbf{D}$ and input $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$, the adversary A designs a PPT algorithm with the following interface,

$$\mathcal{Q} = \{(x_j, j, b_j, y_j)\}_{j \in [\mathsf{q}_{\mathsf{ctrl}}+\mathsf{m}]} \leftarrow \mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{S},\mathbf{D},\mathbf{x},\mathbf{r}}(1^\kappa; r)$$

  where the strategy accepts randomness of the form $r \leftarrow \{0,1\}^{\mathsf{n}\kappa}$. This is detailed in Algorithm 19. This is a randomized algorithm that is given to the challenger $\mathcal{C}$ for the weak CCR security game.
- The challenger $\mathcal{C}$ accepts this $\mathsf{QueryStrategy}^{\mathcal{O}}$ and operates as implicit in Definition 7. That is, it samples a bit $c \leftarrow \{0,1\}$ and if $c = 0$, it samples randomness r and executes

Algorithm 19 Query Strategy $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{S},\mathbf{D},\mathbf{x}}$ for Garbling Scheme $\mathsf{GS}^{\mathcal{H}}$

procedure $\mathsf{QueryStrategy}^{\mathcal{O}}_{\mathbf{S},\mathbf{D},\mathbf{x},\mathbf{r}}(1^\kappa; r \leftarrow \{0,1\}^{\mathsf{n}\kappa})$
  initialize $\mathcal{Q} = []$
  for every $i \in [\mathsf{n}]$ **do**
    set $\mathsf{L}_i^{\mathbf{x}[i]} = r[(i-1)\kappa + 1 : i\kappa] \in \{0,1\}^\kappa$
  end for
  for every randomness wire i set to $\mathbf{r}[i] \in \{0,1\}$ **do**
    set $\mathsf{L}_i^{\mathbf{r}[i]} = \mathbf{r}[i] \cdot \Delta \in \{0,1\}^\kappa$
  end for
  for each $g \in [\mathsf{q}]$ from S in topological order **do**
    if gate g is a switch gate: $y = x \vdash \mathsf{ctrl}$ **then**
      if $\mathsf{ctrl}[g] = 0$ **then**
        $\mathsf{L}_y = \mathsf{L}_x \oplus \mathsf{H}(\mathsf{L}_{\mathsf{ctrl}}, g)$
      else
        compute $\mathsf{L}_y = \mathsf{L}_x \oplus \mathcal{O}(\mathsf{L}_{\mathsf{ctrl}}, g, x \oplus y)$
        set $\mathcal{Q}[g] = (\mathsf{L}_{\mathsf{ctrl}}, g, x \oplus y, \mathsf{L}_x \oplus \mathsf{L}_y)$
      end if
      $\mathbf{F}[g] = \mathsf{lsb}(\mathsf{L}_{\mathsf{ctrl}}) \oplus \mathsf{ctrl}[g]$
    end if
    ⋯ ▷ join, affine, modulus and division gates garbled like in Gb
  end for
  for every $j \in [\mathsf{m}]$ **do**
    set $\mathbf{d}[j] = \{S(\mathbf{x})[j] : \mathsf{H}(\mathsf{L}_j, \mathsf{q}+j),\ \neg S(\mathbf{x})[j] : M = \mathcal{O}(\mathsf{L}_j, \mathsf{q}+j, 0)\}$
    set $\mathcal{Q}[\mathsf{q}+j] = (\mathsf{L}_j, \mathsf{q}+j, 0, M)$
  end for
  return $\mathcal{Q}$
end procedure

$\mathsf{QueryStrategy}^{\mathcal{R}}_{\mathbf{S},\mathbf{D},\mathbf{x}}$. Otherwise, it samples r and Δ, and executes $\mathsf{QueryStrategy}^{\mathcal{C}_{\mathsf{H},\Delta}}_{\mathbf{S},\mathbf{D},\mathbf{x}}$. It gives a tuple $(\mathcal{Q}, \mathsf{H})$ to A.

- Given the set of query responses $\mathcal{Q}$ and the hash algorithm H, the adversary A can now execute Algorithm 17 with active input labels as indicated in the queries in $\mathcal{Q}$ and using the items in $\mathcal{Q}$ where ever oracle queries are required. The output of this is a tuple $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ which is distributed as $\mathbf{Hybrid}_2$ if the bit chosen by the challenger is $c = 0$, and is distributed as $\mathbf{Hybrid}_3$ otherwise.
- A gives $((\mathbf{F}, \mathsf{H}), \mathbf{X}, \mathbf{d})$ to D_{GS} and outputs whatever it outputs.

Note that in the above game, A has the same advantage as that of D_{GS}, which is non-negligible. However, since H is sampled from a weak CCR secure hash function family $\mathcal{H}_\kappa$, it follows that no such A can exist and therefore no such D_{GS} can exist. This completes the proof.

E Supporting Lemmas for Theorem 8

Lemma 1. *If* iO *is secure and* F *is a puncturable PRF, then* $\mathbf{Hybrid}_0$ *and* $\mathbf{Hybrid}_1$ *are computationally indistinguishable.*

Proof. Note that $\mathbf{Hybrid}_0 = \mathbf{Hybrid}_{0,0}$ and $\mathbf{Hybrid}_1 = \mathbf{Hybrid}_{Q,0}$. First, we prove that the sequence of hybrids from $\mathbf{Hybrid}_{0,0}$ to $\mathbf{Hybrid}_{Q,0}$ are computationally indistinguish-

able. For this we need to only prove that $\mathbf{Hybrid}_{j,0}, \mathbf{Hybrid}_{j,1}, \mathbf{Hybrid}_{j,2}$ and $\mathbf{Hybrid}_{j+1,0}$ are computationally indistinguishable for all $j \in [0 : Q(\kappa) - 1]$. Before proceeding we fix some notations. For a PPT distinguisher D and $j \in [0, Q(\kappa) - 1]$, define:

$$p^{\mathsf{D}}_{j,0}(\kappa) \coloneqq \left| \Pr_{x \leftarrow \mathbf{Hybrid}_{j,0}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{j,1}}[\mathsf{D}(x, 1^\kappa) = 1] \right|$$

$$p^{\mathsf{D}}_{j,1}(\kappa) \coloneqq \left| \Pr_{x \leftarrow \mathbf{Hybrid}_{j,1}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{j,2}}[\mathsf{D}(x, 1^\kappa) = 1] \right|$$

$$p^{\mathsf{D}}_{j,2}(\kappa) \coloneqq \left| \Pr_{x \leftarrow \mathbf{Hybrid}_{j,0}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{j+1,1}}[\mathsf{D}(x, 1^\kappa) = 1] \right|.$$

In the following, we will write Q instead of $Q(\kappa)$ except when we want to emphasize that Q is a fixed polynomial.

1. For indistinguishability of $\mathbf{Hybrid}_{j,0}$ and $\mathbf{Hybrid}_{j,1}$ note that

   $$F_k \,||\, \{(x_i \oplus \Delta, 1) \mapsto y^*_i \oplus b_i \Delta\}_{i \in [1:j]}$$

   and

   $$F_{k_{(x_{j+1} \oplus \Delta, j+1)}} \,||\, \{(x_i \oplus \Delta, 1) \mapsto y^*_i \oplus b_i \Delta\}_{i \in [1:j]} \cup \{(x_{j+1} \oplus \Delta, j+1) \mapsto y_{j+1} \oplus b_{j+1} \Delta\}$$

   are functionally equivalent as the output $F_k \,||\, \{(x_i \oplus \Delta, 1) \mapsto y^*_i \oplus b_i \Delta\}_{i \in [1:j]}$ is simply hard-coded on the punctured point. Hence by definition of iO the above two are indistinguishable. Since everything else in the two hybrids is the same we get that any PPT adversary's advantage in distinguishing $\mathbf{Hybrid}_{j,0}$ and $\mathbf{Hybrid}_{j,1}$ is negligible. Thus for any PPT distinguisher D

   $$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_{j,0}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{j,1}}[\mathsf{D}(x, 1^\kappa) = 1] \right| = p^{\mathsf{D}}_{j,0} \leq \mathsf{Adv}^{\mathsf{iO}}_{\mathsf{D}}(\kappa). \tag{14}$$

2. For indistinguishability of $\mathbf{Hybrid}_{j,1}$ and $\mathbf{Hybrid}_{j,2}$, we reduce it to the pseudorandomness of puncturable PRFs. Suppose there exists a PPT adversary A that can distinguish $\mathbf{Hybrid}_{j,1}$ and $\mathbf{Hybrid}_{j,2}$ with a non-negligible advantage then we can create an adversary B that given the punctured key $k_{(x_{j+1} \oplus \Delta, j+1)}$ for puncturing point $(x_{j+1} \oplus \Delta, j+1)$ distinguishes $F_k(x_{j+1} \oplus \Delta, j+1)$ from a randomly sampled value. B would play a PPRF game where it does the following:

   (a) First samples $\Delta \leftarrow \{0,1\}^{\kappa-1} || 1$ and then chooses the puncture point at $(x_{j+1} \oplus \Delta, j+1)$.
   (b) B receives the punctured key $k_{(x_{j+1} \oplus \Delta, j+1)}$ and challenge c from the challenger. The challenge c is either $F_k(x_{j+1} \oplus \Delta, j+1)$ or a uniformly random value.
   (c) B simulates a hybrid distinguishing game with A
   (d) Upon receiving $\mathsf{QueryStrategy}^{(\cdot)}$ from A, B fixes a random r and samples $y^*_i \leftarrow \{0,1\}^\kappa \; \forall i \in [1 : j]$.

(e) Then B creates

$$\begin{aligned}
\mathsf{H} =& \mathsf{iO}(F_{k_{(x_{j+1}\oplus\Delta,j+1)}} \,||\, \{(x_i \oplus \Delta, 1) \mapsto y_i^* \oplus b_i\Delta\}_{i\in[1:j]} \cup \\
& \{(x_{j+1} \oplus \Delta, j+1) \mapsto c' \oplus b_{j+1}\Delta\}) \\
& \text{such that} \\
(x_1, b_1) =& \mathsf{QueryStrategy}(1^\kappa; r), \quad y_1 = \mathcal{C}_{\mathsf{H},\Delta}(x_1, 1, b_1) = y_1^* \\
(x_i, b_i) =& \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_{i-1}^*)\, \forall i \in [2:j] \\
y_i =& \mathcal{C}_{\mathsf{H},\Delta}(x_i, i, b_i) = y_i^* \;\; \forall i \in [2:j] \\
(x_{j+1}, b_{j+1}) =& \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*) \\
c' =& c \oplus b_{j+1}\Delta \\
y_{j+1} =& \mathsf{H}(x_{j+1} \oplus \Delta, j+1) \oplus b_{j+1}\Delta = c' = c \oplus b_{j+1}\Delta \\
(x_i, b_i) =& \mathsf{QueryStrategy}(1^\kappa; r, y_1^*, \cdots, y_j^*, y_{j+1}, \cdots, y_{i-1}) \;\; \forall i \in [j+2:Q] \\
y_i =& \mathsf{H}(x_i \oplus \Delta, i) \oplus b_i\Delta \;\; \forall i \in [j+2:Q]
\end{aligned}$$

(f) B sends to A $\left\{\mathsf{H}, (((x_i, i), b_i, y_i^*))_{i\in[1:j]}, ((x_i, i, b_i, y_i))_{i\in[j+1:Q]})\right\}$

(g) If A replies with $\mathbf{Hybrid}_{j,1}$, B replies to PPRF challenger with 0, otherwise with 1.

We can infer B's advantage by observing that c is either $F_k(x_{j+1} \oplus \Delta, j+1)$ or a uniformly random value. If c is a uniformly random value then so is c'. Thus A gets either $\mathbf{Hybrid}_{j,1}$ or $\mathbf{Hybrid}_{j,2}$. Therefore $\mathsf{Adv}_{\mathsf{B}}^{PPRF} \geq \mathsf{Adv}_{\mathsf{A}}$ on hybrid game. Thus for any PPT distinguisher D, there exists an adversary B such that,

$$\left| \Pr_{x\leftarrow\mathbf{Hybrid}_{j,1}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x\leftarrow\mathbf{Hybrid}_{j,2}}[\mathsf{D}(x, 1^\kappa) = 1] \right| = p_{j,1}^{\mathsf{D}} \leq \mathsf{Adv}_{\mathsf{B}}^{PPRF}(\kappa). \quad (15)$$

3. For indistinguishability of $\mathbf{Hybrid}_{j,2}$ and $\mathbf{Hybrid}_{j+1,0}$, note that similar to $\mathbf{Hybrid}_{j,0}$ to $\mathbf{Hybrid}_{j,1}$, we have function equivalence of

$$F_{k_{(x_{j+1}\oplus\Delta,j+1)}} \,||\, \{(x_i \oplus \Delta, 1) \mapsto y_i^* \oplus b_i\Delta\}_{i\in[1:j]} \cup \{(x_{j+1} \oplus \Delta, j+1) \mapsto y_{j+1}^* \oplus b_{j+1}\Delta\}$$

and

$$F_k \,||\, \{(x_i \oplus \Delta, 1) \mapsto y_i^* \oplus b_i\Delta\}_{i\in[1:j]} \cup \{(x_{j+1} \oplus \Delta, j+1) \mapsto y_{j+1}^* \oplus b_{j+1}\Delta\}$$

by definition of PPRF and hard-coding. Thus the advantage of any PPT distinguisher D is bounded above by $\mathsf{Adv}_{\mathsf{D}}^{\mathsf{iO}}(\kappa)$. Therefore, for any PPT distinguisher D

$$\left| \Pr_{x\leftarrow\mathbf{Hybrid}_{j,2}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x\leftarrow\mathbf{Hybrid}_{j+1,0}}[\mathsf{D}(x, 1^\kappa) = 1] \right| = p_{j,2}^{\mathsf{D}} \leq \mathsf{Adv}_{\mathsf{D}}^{\mathsf{iO}}(\kappa) \quad (16)$$

Suppose $\mathbf{Hybrid}_{0,0}, \mathbf{Hybrid}_{Q,0}$ are not computational indistinguishable *i.e.* there exists a PPT distinguisher D and a positive polynomial $p(\cdot)$ such that for every $N \in \mathbb{N}$ there exists $\kappa > N$ such that

$$\left| \Pr_{x\leftarrow\mathbf{Hybrid}_{0,0}}[\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x\leftarrow\mathbf{Hybrid}_{Q,0}}[\mathsf{D}(x, 1^\kappa) = 1] \right| > \frac{1}{p(\kappa)}.$$

Using triangle inequality, we get

$$\begin{aligned}
&\left|\Pr_{x\leftarrow\mathbf{Hybrid}_{0,0}}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_{0,1}}[\mathsf{D}(x,1^\kappa)=1]\right|+\\
&\left|\Pr_{x\leftarrow\mathbf{Hybrid}_{0,1}}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_{0,2}}[\mathsf{D}(x,1^\kappa)=1]\right|+\\
&\vdots\\
&+\left|\Pr_{x\leftarrow\mathbf{Hybrid}_{Q-1,2}}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_{Q,0}}[\mathsf{D}(x,1^\kappa)=1]\right|\\
&\geq\left|\Pr_{x\leftarrow\mathbf{Hybrid}_{0,0}}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_{Q,0}}[\mathsf{D}(x,1^\kappa)=1]\right|\\
&>\frac{1}{p(\kappa)}
\end{aligned}$$

This implies

$$p^{\mathsf{D}}_{0,0}+p^{\mathsf{D}}_{0,1}+p^{\mathsf{D}}_{0,2}+p^{\mathsf{D}}_{1,0}+\cdots+p^{\mathsf{D}}_{Q,0}>\frac{1}{p(\kappa)}$$

Combining this with results from Eqs. (14) to (16) we get that

$$\begin{aligned}
2\cdot Q(\kappa)\cdot\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{D}}(\kappa)+Q(\kappa)\cdot\mathsf{Adv}^{PPRF}_{\mathsf{B}}(\kappa)&>\frac{1}{p(\kappa)}\\
\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{D}}(\kappa)+\mathsf{Adv}^{PPRF}_{\mathsf{B}}(\kappa)&>\frac{1}{2\cdot p(\kappa)\cdot Q(\kappa)}
\end{aligned}$$

By averaging, either $\mathsf{Adv}^{\mathsf{iO}}_{D}>\frac{1}{4\cdot p(\kappa)\cdot Q(\kappa)}$ or $\mathsf{Adv}^{PPRF}_{\mathsf{B}}>\frac{1}{4\cdot p(\kappa)\cdot Q(\kappa)}$ must hold. Either way, we will get that there exists a PPT adversary D or B and a polynomial $4\cdot p(\kappa)\cdot Q(\kappa)$ such that for all $N\in\mathbb{N}$ there exists $\kappa>N$ such that $\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{D}}>\frac{1}{4\cdot p(\kappa)\cdot Q(\kappa)}$ or $\mathsf{Adv}^{PPRF}_{\mathsf{B}}>\frac{1}{4\cdot p(\kappa)\cdot Q(\kappa)}$. This contradicts either iO assumption (Definition 3) or PPRF assumption (Definition 2).

Lemma 2. *If iO is secure and F is a puncturable PRF, then* $\mathbf{Hybrid}_1$ *and* $\mathbf{Hybrid}_7$ *are computationally indistinguishable.*

Proof. We proceed via a hybrid argument.

1. For $\mathbf{Hybrid}_1, \mathbf{Hybrid}_2, \mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$, we simply note that the following four circuits are functionally equivalent:

$$F_k \mid\mid \{(x_i\oplus\Delta,i)\mapsto y^*_i\oplus b_i\Delta\}_{i\in[Q]}$$

$$C(x,i)=\begin{cases}y^*_i\oplus b_i(x\oplus x_i), & \text{if } x\oplus x_i=\Delta\\ F_k(x,i), & \text{otherwise.}\end{cases}$$

$$C(x,i)=\begin{cases}y^*_i\oplus b_i(x\oplus x_i), & \text{if } \delta_\Delta(x\oplus x_i)=1\\ F_k(x,i), & \text{otherwise.}\end{cases}$$

$$C(x,i)=\begin{cases}y^*_i\oplus b_i(x\oplus x_i), & \text{if } \delta_{\pi(\Delta)}\circ\pi(x\oplus x_i)=1\\ F_k(x,i), & \text{otherwise.}\end{cases}$$

Since the description of rest of the hybrids remains the same, by security of iO we get that $\mathbf{Hybrid}_1, \mathbf{Hybrid}_2, \mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$ are computationally indistinguishable. Formally, for any PPT distinguisher D, any positive polynomial $p(\cdot)$ and large enough κ, and $\forall i \in [1,3]$

$$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_i} [\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{i+1}} [\mathsf{D}(x, 1^\kappa) = 1] \right| \leq \frac{1}{p(\kappa)}.$$

2. For $\mathbf{Hybrid}_4$ and $\mathbf{Hybrid}_5$, suppose there is a PPT adversary A such that it can distinguish $\mathbf{Hybrid}_4, \mathbf{Hybrid}_5$ with non-negligible probability. Then we can create a PPT adversary B which can compute $h(\Delta)$ given $f(\Delta)$ for a randomly generated Δ with a non-negligible probability. This would contradict the property of a hardcore predicate (Definition 1). B upon receiving $f(\Delta)$ for a randomly sampled Δ simply puts $\pi'(\Delta) = (f(\Delta), 0)$ and simulates $\mathbf{Hybrid}_4$ except with $\delta_{\pi'(\Delta)}$ in place of $\delta_{\pi(\Delta)}$. Now if A replies with 0 (corresponding to $\mathbf{Hybrid}_4$) then B also replies with 0 otherwise with 1. We should note that B doesn't need to know Δ to simulate $\mathbf{Hybrid}_4$ with $\delta_{\pi'(\Delta)}$ in place of $\delta_{\pi(\Delta)}$. It just needs to know $f(\Delta)$ which it receives from the hardcore predicate challenger. Formally, for any PPT distinguisher D, any positive polynomial $p(\cdot)$ and large enough κ,

$$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_4} [\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_5} [\mathsf{D}(x, 1^\kappa) = 1] \right| \leq \frac{1}{p(\kappa)}$$

3. For $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_6$, note that $\delta_{\overline{\pi}(\Delta) \circ \pi(x \oplus x_i)}$ is always 0. Thus the two circuits in $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_6$ are functionally equivalent. Since the remaining part of the hybrid is the same, by definition of iO the hybrids are indistinguishable. Thus, the advantage of distinguishing them is bounded above by $\mathsf{Adv}^{\mathsf{iO}}(\kappa)$, which is negligible. Formally, for any PPT distinguisher D, any positive polynomial $p(\cdot)$ and large enough κ,

$$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_5} [\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_6} [\mathsf{D}(x, 1^\kappa) = 1] \right| \leq \frac{1}{p(\kappa)}$$

4. For $\mathbf{Hybrid}_6$ and $\mathbf{Hybrid}_7$ note that by definition of $\mathcal{R}$, the distribution of $\{y_i^*\}_{i \in [Q]}$ is same as $\{y_i : y_i \leftarrow \mathcal{R}(x_i, i, b_i)\}_{i \in [Q]}$. Since the rest is the same in both hybrids, they are identical. Formally, for any PPT distinguisher D

$$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_5} [\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_6} [\mathsf{D}(x, 1^\kappa) = 1] \right| = 0$$

Now, suppose for contradiction that $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_7$ are not computationally indistinguishable *i.e.* there exists a PPT distinguisher D and a positive polynomial $p(\cdot)$ such that for every $N \in \mathbb{N}$ there exists $\kappa > N$ such that

$$\left| \Pr_{x \leftarrow \mathbf{Hybrid}_{0,0}} [\mathsf{D}(x, 1^\kappa) = 1] - \Pr_{x \leftarrow \mathbf{Hybrid}_{Q,0}} [\mathsf{D}(x, 1^\kappa) = 1] \right| > \frac{1}{p(\kappa)}.$$

Using triangle inequality, we get

$$\begin{aligned}
&\left|\Pr_{x\leftarrow\mathbf{Hybrid}_0}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_1}[\mathsf{D}(x,1^\kappa)=1]\right|+\\
&\left|\Pr_{x\leftarrow\mathbf{Hybrid}_1}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_2}[\mathsf{D}(x,1^\kappa)=1]\right|+\\
&\vdots\\
&+\left|\Pr_{x\leftarrow\mathbf{Hybrid}_6}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_7}[\mathsf{D}(x,1^\kappa)=1]\right|\\
&\geq\left|\Pr_{x\leftarrow\mathbf{Hybrid}_0}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_1}[\mathsf{D}(x,1^\kappa)=1]\right|\\
&>\frac{1}{p(\kappa)}
\end{aligned}$$

By an averaging argument, there exists at least one $i\in[1,6]$ such that

$$\left|\Pr_{x\leftarrow\mathbf{Hybrid}_i}[\mathsf{D}(x,1^\kappa)=1]-\Pr_{x\leftarrow\mathbf{Hybrid}_{i+1}}[\mathsf{D}(x,1^\kappa)=1]\right|>\frac{1}{6\cdot p(\kappa)}$$

If $i<3$, this contradicts Item 1. If $i=4,5,6$, this contradicts Items 2 to 4 respectively.