Unmodified Half-Gates and Three-Halves are Adaptively Secure

Abstract—**Adaptive security is a crucial property of garbling schemes in pushing the communication of garbled circuits to an offline phase when the input is unknown yet. It is a much-needed property when minimizing online latency. In this paper, we show that the popular selective-secure half-gates scheme (Eurocrypt'15), without any modification, is adaptively secure in the non-programmable random permutation model (npRPM). Since real implementations of half-gates already use npRPM for hardware acceleration (S&P'13, S&P'20), our result shows that these implementations are already adaptively secure under the same condition where selective security is achieved. A practical implication of our result is that existing systems using garbled circuits can safely send the majority of their communication before the inputs are known without changing the garbling scheme, thus improving online latency without any additional computational overhead. Additionally, we expand our analysis to cover the recent three-halves construction (Crypto'21). As a byproduct, we discuss some optimizations and separation when considering the programmable random permutation model instead.**

1. Introduction

Garbled circuit (GC) [1], [2] is a core building block of many cryptographic applications, such as secure two-party computation (2PC) [1], zero-knowledge proofs [3], and identity-based encryption [4]. In a garbling scheme, a garbler creates a garbled circuit $\widehat{f}$ and a garbled input $\widehat{x}$ for circuit f and input x, respectively. An evaluator can compute a garbled version of $f(x)$ from $(\widehat{f}, \widehat{x})$ and decode it.

Two categories of GC security have been widely adopted in the applications: selective security and adaptive security. In selective security, the adversary chooses (f, x) in one shot and is asked to distinguish between a real-world execution consisting of honestly computed $(\widehat{f}, \widehat{x})$ and an ideal-world execution where the values are simulated using only $f(x)$. In adaptive security, the adversary is given more power to adaptively choose input x based on $\widehat{f}$: it first chooses f and receives $\widehat{f}$ in the offline phase, and then chooses x and receives $\widehat{x}$ in the online phase. Clearly, adaptive security is stronger than selective security. It is also more desireable because it allows us to achieve significantly lower online communication cost, by pushing the communication of garbled circuits (linear in $|f|$) to the offline phase. As a result, a direct application of adaptive garbling is to design 2PC protocols with low online communication, e.g., [5], [6], [7], [8], [9], [10]. In addition, adaptively secure garbling schemes can facilitate one-time programs [11], functional encryption [12], [13], verifiable computation [14], [15] in reducing the online cost.

Both security notions have been heavily studied but with very different philosophies and outcomes:

- For selective security, most prior works focus on how to reduce the computational cost and the size of garbled circuit $\widehat{f}$. An impressive line of GC research [1], [16], [17], [18], [19], [20], [21], [22], [23] have reduced the size of $\widehat{f}$ from 8λ to 2λ [22] or 1.5λ [23] bits per `AND` gate, while `XOR` gates are free using the free-XOR technique [18], where λ is the security parameter. To reduce the computational cost, almost existing implementations instantiate the required correlation robust hash functions in the non-programmable random permutation model (RPM), which provides forward and backward queries to a public random permutation. It is then commonly instantiated by fixed-key AES and accelerated using AES-NI [24], [25]. This leads to garbling schemes producing more than 20 million garbled `AND` gates per second.
- Most work in adaptive security take a more theoretical aspect. In particular, prior scheme in adaptive security [26], [27], [28], [29], [30], [31] are mostly in the standard model but all require some compromise either in exponential security loss or in undesirable online communication, unless Cryptomania assumptions are used. When adaptive garbling is needed in implementations, they all need a programmable random oracle (RO) which is often instantiated using a cryptographic hash function. It can be done either following the work by Bellare et al. [26] where a generic selective-to-adaptive transformation was proposed [7], [9], or directly proving that the garbling scheme is secure in programmable RO directly [8].

As a summary, practically implemented selective garblings live in the non-programmable RPM, while practically usable adaptive garblings are all in the programmable ROM, which is significantly slower. In Table 1, we present all garbling schemes with adaptive security and polynomial security loss.

Contribution. In this work, we realize concretely efficient adaptive garbling by showing that half-gates [22] and three-halves [23], the two state-of-the-art selectively secure garbling schemes, already comply with the adaptive security in the non-programmable RPM (npRPM) [32], [24]. As real implementations of the two schemes already use a random permutation to instantiate circular correlation robust (CCR) hash functions [22], [25], [23], our results are essentially proven in the same model as the selective-secure setting. We view our result as something valuable in both practice and theory. On the practical side, by not changing the garbling schemes at all, the resulting schemes incur zero additional

Schemes	Offline Cost	Online Cost	Assumption
[27]	$\mathsf{C} \cdot 4\lambda$	$m + n\lambda + O(w \cdot \lambda^2 \cdot \log \mathsf{C})$	OWF
[30]	$\mathsf{C} \cdot \mathsf{poly}(\lambda)$	$m + n + \mathsf{poly}(\lambda, \log \mathsf{C})$	CDH/LWE/etc
[26]	$\mathsf{A} \cdot \{1.5\lambda, 2\lambda\} + m$	$n\lambda$	pROM
[8]	$\mathsf{A} \cdot 3\lambda + m$	$n\lambda$	pROM
This work	$\mathsf{A} \cdot \{1.5\lambda, 2\lambda\}$	$m + n\lambda$	npRPM
	$\mathsf{A} \cdot \{1.5\lambda, 2\lambda\} + m$	$n\lambda$	pRPM

TABLE 1: **Comparison of adaptively secure garbling schemes.** We present the schemes without exponential security loss. The offline and online costs only focus on communication, where minor terms are omitted for simplicity. The notation C (resp., A) is the total number of all gates (resp., AND gates only). Let λ, n, m, w be the security parameter, input size, output size, and circuit width, respectively. OWF denotes one-way function. pROM denotes the programmable ROM. pRPM (resp., npRPM) denotes the programmable (resp., non-programmable) RPM.

computation/communication overhead and no change of implementation. That is, we can also use AES-NI provided by modern CPUs to improve the efficiency of adaptive garbling (see Table 2). On the theoretic side, although the resulting construction still needs an ideal model, it is the first time that the adaptive security of concretely efficient garbling schemes is proven in a non-programmable model. Moreover, a λ-to-λ random permutation appears more plausible than a random oracle with unlimited entropy. In conclusion,

1) We prove that both half-gates and three-halves can be adaptively secure in the npRPM by simply postponing a decoding table, which is output by their garbling algorithms, to the online phase. This modification is easy to existing implementations and offloads some communication from the offline phase to the online one. The obtained adaptively secure schemes can be essentially as computation- and communication-efficient as half-gates and three-halves, improving the end-to-end performance of many applications of adaptive garbling.
2) We also prove that the unmodified half-gates and three-halves are already adaptively secure in the programmable RPM (pRPM) to cover the known implementations that send the decoding table in the offline phase. This result complements the previous one by allowing one to make a trade-off between concrete online efficiency and cryptographic assumption since a non-programmable model is more conservative and welcomed. For applications that have long output length and want to reduce their online latency, this result is of the most relevant importance.
3) We prove that the above difference in online communication is inherent in the programmability of random permutation, which leads to a separation of adaptively secure garbling schemes in the two RPMs and implies a lower bound of online communication in the npRPM. This lower bound extends the one [6], [33] obtained in the standard model.

Schemes	GC Size (Bytes)	Offline Garbling (CPU cycles)	Online Evaluation (CPU cycles)
[8]	64	2376	1188
[26]	32	650	622
This work	32	56	28

TABLE 2: **Comparison of concretely efficient adaptively secure garbling schemes.** We measure the communication (i.e., garbling size) and computation (i.e., offline garbling and online evaluation) per `AND` gate, while all `XOR` gates are free. The numbers of CPU cycles are estimated from the performance reported in [25]. We use SHA256 compression function to instantiate the short-input RO and fixed-key AES to instantiate the random permutation.

2. Technical Overview

2.1. Previous Techniques

We note that known techniques fail to prove adaptive security of half-gates [22] and three-halves [23], the two state-of-the-art garbling schemes with selective security. These schemes adopt free-XOR optimization [18], where an active label XORed with its coupled inactive label matches a global offset $\varDelta$ for each wire. In these schemes, a gate ciphertext is an one-time pad (OTP) encryption with a mask $\mathsf{H}(X \oplus \varDelta, k) \oplus b\varDelta$ for active label X, tweak k, and bit b *dependent on a truth bit.*

The selective security of the two schemes can be reduced to circular correlation robust (CCR) hash functions by properly dealing with tweaks. More specifically, the reduction algorithm adaptively calls the CCR oracle to obtain pseudorandom masks of form $\mathsf{H}(X \oplus \varDelta) \oplus b\varDelta$ when it is given an input x chosen by the selective adversary. Here, input x is used to compute all truth bits on circuit f to get all b's in the masks. The obtained masks bridge a real garbled circuit and a simulated one. However, for adaptive security, the same reduction algorithm fails since it has not received an online input from the adaptive adversary when an offline garbled circuit should be simulated.

For adaptive security, almost all previous proofs [27], [28], [29], [30], [31] follow the proof method [34] of Yao's garbling scheme in the selective setting (i.e., the approach based on pebbling games)[1]. Such a pebbling game changes all real-world garbled gates (i.e., white pebbles) to simulated ones (i.e., black pebbles) using a carefully designed hybrid argument, where each hybrid bridges an input-dependent garbled gate (i.e., gray pebble) and a real-world or simulated one. To ensure that such gray pebbles are consistent with the input (being undefined until the online phase), these works adopt somewhere equivocal encryption or piecewise guessing in the construction of gray pebbles, which incurs high overhead or non-negligible security loss.

In the pebbling-game-based works, there are two notable facts: (i) a pebbling hybrid changes only one pebble, and (ii) the indistinguishability between a gray pebble and a

[1] The exceptions are the work by Bellare et al. [26] using circuit-wise padding in the pROM and the work by Lindell et al. [8] with a gate-by-gate use of the pROM.

white/black one comes from a *black-box* reduction to the security of some cryptographic primitive. So, any two pebbling hybrids should be respectively reduced to two *independent* instances of the primitive in a black-box way. In these works, this primitive acts as an encryption scheme for truth bits and all encryptions should be independent. However, all garbled gates in half-gates and three-halves are correlated under the same free-XOR offset $\varDelta$. This correlation implies that a pebble-by-pebble hybrid argument cannot prove the adaptive security of these schemes.

To sum up, we need to address the following two challenges *simultaneously* to prove the adaptive security of half-gates and three-halves:

(c1) How to consider all garbled gates as a whole garbled circuit to capture that they are correlated under a global offset $\varDelta$? This challenge is *not* solved by the prior works of adaptive garbling without relying on programmable ROM.

(c2) How to prove the indistinguishability between a simulated garbled circuit and a real one based on a proper computational assumption, without modifying the two schemes?
Clearly, this indistinguishability is necessary for adaptive garbling since the adaptive adversary is given the garbled circuit. The known proofs of the two unmodified schemes turn to CCR hash functions. As recalled, these CCR-based proofs can only address this challenge in the selective setting. For adaptive security, they fail because the online input is unspecified when the *input-dependent* CCR queries should be made to the CCR oracle to simulate the garbled circuit.

2.2. Our Approach

Below, we outline how to address the above two challenges and prove adaptive security of the two schemes, followed by a toy example.

Core idea: Using statistical distance instead of complexity-theoretic reduction. In our proofs, we do not pursue security reduction to computationally secure primitives but study adaptive garbling in a general statistical framework: a computationally unbounded non-uniform adversary $\mathcal{A}$ adaptively interacts with either a real-world oracle $\mathcal{O}_0$ or an ideal-world oracle $\mathcal{O}_1$ and outputs its decision bit after the interaction. Both oracles provide the same query interfaces to $\mathcal{A}$. The interaction between $\mathcal{A}$ and $\mathcal{O}_b$ defines a random variable Z_b of transcripts, which records query-response pairs in order. Without loss of generality, we assume that non-uniform adversary $\mathcal{A}$ is *deterministic*[2] so that its decision bit is a deterministic function of its auxiliary input and a transcript sampled according to Z_b. It is well-known that the advantage of adaptive adversary $\mathcal{A}$ is upper bounded by statistical distance $\mathsf{SD}(Z_0, Z_1)$. This statistical perspective considers a stronger adaptive adversary than the complexity-theoretic one and paves a way to prove adaptive security other than reduction to adaptively secure primitives.

More specifically, adaptivity in this framework is captured by the ordered query-response pairs in a transcript. Note that $\mathsf{SD}(Z_0, Z_1)$ is defined from probability $\Pr[Z_b = \tau]$ for each possible transcript τ and $b \in \{0,1\}$. To compute this probability, it is crucial to deal with the adaptivity that is implicit in the defined random variable of transcripts. Let us consider fixed $b \in \{0,1\}$ and transcript τ of ordered pairs $((q_1, r_1), \ldots, (q_n, r_n))$. Intuitively, probability $\Pr[Z_b = \tau] = 0$ if the next-message function of a fixed non-uniform deterministic $\mathcal{A}$ can never produce queries $q_1, \ldots, q_n$ in order when responses $r_1, \ldots, r_n$ arrives in order. Otherwise, $\mathcal{A}$ certainly produces these queries upon receiving $r_1, \ldots, r_n$ in order (as $\mathcal{A}$ is deterministic and it is a yes-or-no event) and $\Pr[Z_b = \tau]$ quantitatively matches the probability that $\mathcal{A}$ is given responses $r_1, \ldots, r_n$ in order, i.e., the probability that oracle $\mathcal{O}_b$ produces responses $r_1, \ldots, r_n$ in order if queries $q_1, \ldots, q_n$ arrives in order. Plugging this observation (which is implicit in the analysis [36], [37], [38], [39] of symmetric-key primitives and recalled in Lemma 1) into the framework, we have a general proof blueprint of adaptive security:

Proof blueprint of adaptive security

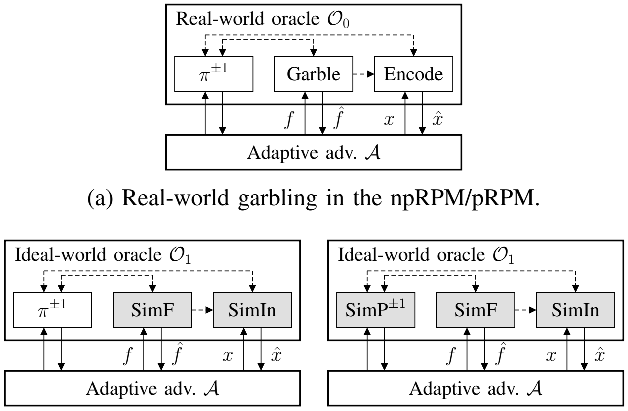
> For each transcript τ with $\Pr[Z_0 = \tau] \neq 0$ or $\Pr[Z_1 = \tau] \neq 0$ (i.e., τ raises $\mathsf{SD}(Z_0, Z_1)$), if $\mathcal{O}_0$ and $\mathcal{O}_1$ have *statistically close* probability of being "*compatible with* τ", i.e., producing the ordered responses in τ when given the ordered queries in τ, then the advantage of $\mathcal{A}$ is negligible.

Note that the two probabilities are only taken over the randomness of the two oracles, respectively.

In essence, this statistical blueprint quantifies over all possible interaction transcripts in both two worlds and replaces the adaptive adversary with all sequences of ordered queries in these transcripts. Using this quantification, we no longer run an adaptive adversary to extract its adaptive queries and use the oracle of some low-level primitive to answer each query as in complexity-theoretic reduction. Instead, we consider all possible outcomes of adaptive queries to bound their effect on the advantage of an adaptive adversary. So, the proof blueprint will not confront a challenge in the reduction-based proofs of adaptive security: given the real execution where a response will be consistent with future adaptive queries as per the real oracle of some low-level primitive, and the ideal execution where the response is sampled at random as in the ideal oracle, how to bridge the two executions via a security reduction to the primitive? This challenge has been noticed by the known proofs of adaptive garbling (see challenge (c2) in Section 2.1) and causes additional costs.

Remark 1 (Computationally secure primitives are useless in the proof blueprint)**.** *Note that the proof blueprint requires that the two probabilities of being compatible with each possible transcript have negligible difference. Clearly, this statistical closeness cannot be bounded by a computational advantage in the complexity-theoretic reduction to some*

[2] A non-uniform adversary is at least as powerful as a probabilistic adversary [35].


Real-world oracle $\mathcal{O}_0$
$\pi^{\pm 1}$
Garble
Encode
f
$\hat{f}$
x
$\hat{x}$
Adaptive adv. $\mathcal{A}$
Ideal-world oracle $\mathcal{O}_1$
$\pi^{\pm 1}$
SimF
SimIn
f
$\hat{f}$
x
$\hat{x}$
Adaptive adv. $\mathcal{A}$
Ideal-world oracle $\mathcal{O}_1$
SimP$^{\pm 1}$
SimF
SimIn
f
$\hat{f}$
x
$\hat{x}$
Adaptive adv. $\mathcal{A}$

(a) Real-world garbling in the npRPM/pRPM.

(b) Ideal-world garbling in the npRPM.

(c) Ideal-world garbling in the pRPM.

Figure 1: Experiments of adaptive garbling in the npRPM and pRPM. (a) versus (b): The experiment in the npRPM, with simulator $(\mathsf{SimF}, \mathsf{SimIn})$. (a) versus (c): The experiment in the pRPM, with simulator $(\mathsf{SimF}, \mathsf{SimIn}, \mathsf{SimP}^{\pm 1})$. In the experiments, $\mathcal{A}$ can make permutation queries at any time.

computationally secure primitive. Instead, one can turn to explicit randomness (e.g., ideal models and uniform coins) to compute this difference directly.

RPM-based adaptive garbling from the proof blueprint. We prove the adaptive security of half-gates and three-halves by instantiating the blueprint. In our proof, real-world oracle $\mathcal{O}_0$ runs the garbling scheme while ideal-world oracle $\mathcal{O}_1$ is defined from the simulation. As the two schemes are in RPM, both oracles provide adaptive adversary $\mathcal{A}$ with not only an offline interface for garbled circuit and an online interface for garbled input, but also two interfaces for the random permutation and its inverse. In Figure 1, we illustrate two experiments of adaptive garbling in this blueprint according to whether the RPM is programmable or not. In either RPM, $\mathcal{A}$ can make queries to the random permutation and its inverse at any time; the online interface can only be queried by $\mathcal{A}$ after the offline one. In each world, the probability of being compatible with any fixed transcript is taken over the random permutation and other uniform random coins. Challenge (c1) is addressed as a garbled circuit is a part of a transcript, and we have discussed that the blueprint can bypass challenge (c2) in general.

To simplify probability analysis in the blueprint, our proof considers two relaxations of the statistical distance. First, we study the statistical distance for a more powerful adaptive adversary, which is explicitly given more messages by padding them into transcripts. Since the adversary can omit these padding messages at will, a quick proof using the optimal-distinguisher-based definition of statistical distance can show that this new statistical distance is at least the original one. Roughly, the padding messages fix (i) the RPM-based hash values of all active labels (along with the query-response pairs of the random permutation and its inverse therein), (ii) all truth bits and active labels on internal and output wires, (iii) free-XOR offset $\varDelta$ (which is revealed at the end of interaction so that no more permutation queries can depend on it), and (iv) all randomization bits (only in three-halves). The padding messages along with the original transcript explicitly fix all randomness in garbling but capture that no arbitrary permutation query depends on $\varDelta$. The fixed randomness makes it easier to compute the probability that the random permutation and its inverse are consistent with the query-response pairs in transcripts. We call this relaxation *transcript padding*.

Second, we use the H-coefficient technique [36], [37], [38] to bound the above statistical distance instead of computing it directly. For any fixed (more powerful) $\mathcal{A}$, let $\mathcal{T}_{\mathcal{A}}$ be a set collecting every transcript τ with $\Pr[Z_0 = \tau] \neq 0$ or $\Pr[Z_1 = \tau] \neq 0$. This technique divides $\mathcal{T}_{\mathcal{A}}$ into a set $\mathcal{T}_{\mathsf{bad}}$ of *bad transcripts* and a set $\mathcal{T}_{\mathsf{good}} := \mathcal{T}_{\mathcal{A}} \setminus \mathcal{T}_{\mathsf{bad}}$ of *good transcripts*. It proves that $\mathsf{SD}(Z_0, Z_1) \leq \varepsilon_1 + \varepsilon_2$ if $\sum_{\tau \in \mathcal{T}_{\mathsf{bad}}} \Pr[Z_1 = \tau] \leq \varepsilon_1$ and, for every $\tau \in \mathcal{T}_{\mathsf{good}}$, $1 - \Pr[Z_0 = \tau] / \Pr[Z_1 = \tau] \leq \varepsilon_2$.

How to define bad and good transcripts? The definition of these transcripts depends on whether the RPM is programmable or not. In our proofs of half-gates and three-halves, npRPM considers the known implementations that send the decoding information in the online phase (e.g., Obliv-C [40], ObliVM [41], and TinyGarble [42]) while pRPM can facilitate those with offline decoding information (e.g., ABY [43] and MP-SPDZ [44]).

As the main component of transcripts, a (simulated) garbled circuit is sampled by the ideal-world oracle at random. In the real world, we require a statistically close distribution of real garbled circuit. Very roughly, in a real garbled circuit, each gate ciphertext equals the XOR of (a) the two RPM-based hash values of one active label and its coupled inactive label, (b) a linear function of two active labels, and (c) a linear function of offset $\varDelta$. In free-XOR, an inactive label is its coupled active one XORed with $\varDelta$. In the two schemes, the hash function is tweakable and can be instantiated with $\mathsf{H}(X, k) = \pi(X \oplus k) \oplus \sigma(X \oplus k)$, where π is a random permutation and σ is a linear orthomorphism (see [25] or Section 3.1 for details). Using this hash function in (a), we see each gate ciphertext is effectively masked by some permutation image of form $\pi(X \oplus \varDelta \oplus k)$ for active label X and tweak k. To achieve the above statistical closeness, such masks should be OTPs so that they are not trivially revealed in transcripts or do not have pairwise collision.

In the npRPM, a transcript is bad if and only if it violates this OTP requirement. First, such a mask is revealed if and only if there exists a query-response pair of form $(X \oplus \varDelta \oplus k, \pi(X \oplus \varDelta \oplus k))$ in the transcript. Second, two masks of form $\pi(X \oplus \varDelta \oplus k)$ lead to pairwise collision if and only if at least one the following events occurs:

1) There exists collision between two permutation pre-images of form $X \oplus \varDelta \oplus k$. Equivalently, there exists collision between two permutation pre-images of form $X \oplus k$, or between two permutation images of form $\pi(X \oplus k)$. All implications follow from that the random permutation is invertible by querying its inverse. For the equivalent event, all relevant values of pre-images

and images are fixed by the query-response pairs in the transcript given (i) in transcript padding.

2) There exists collision between the values of two masks, each of which is from subtracting other values from a masked gate ciphertext. These subtracted values are fixed by the transcript as per (a), (b), and (c). In particular, all linear functions in (b) and (c) are defined from (ii) (as well as (iv) in three-halves) in transcript padding.

The badness of such transcripts results from that the attacks against OTPs work when the adversary is given these transcripts, blowing up the statistical distance.

Remark 2 (On unique tweaks to resist a trivial pairwise collision between masks)**.** *Some attacks (e.g., [45]) have shown that the pairwise collision between masks can be exploited to distinguish between the real and ideal executions. Since the adversary can choose a circuit at will, this circuit can have some wire i (or rather, its active label X_i) being used more than once (e.g., in circuit $x_j = \mathtt{AND}(x_i, x_i)$). In this case, bad transcripts also capture the pre-image collision between $X_i \oplus k$ and $X_i \oplus k'$, i.e., between tweaks k and k'. Since all tweaks are public, they are required to be pairwise distinct to avoid this trivial collision.*

In half-gates, a tweak is computed from a unique gate ID and an indicator bit of either input wire of this gate using the tweak. In the three-halves with its computational optimization[3], we have a counter ctr_i for each wire i and, upon every use of active label X_i, use $(i \,\|\, \mathsf{ctr}_i)$ as a fresh tweak and increase ctr_i by 1. The above definitions ensure globally unique tweaks. Even if free-XOR in the two schemes produces two syntactically identical active labels $X_i = X_j$ on two wires $i \neq j$, such tweaks are still pairwise distinct. Indeed, Nieminen and Schneider [45] have pointed out that half-gates and three-halves are not *vulnerable to the attack.*

In the pRPM, bad transcripts correspond to the attacks against OTPs or failed programming. Note that this model allows for offline decoding information, which should be consistent with the LSBs of active output labels and the truth bits on output wires. Since these truth bits are not fixed until the online phase, we program the random permutation and its inverse to let them output the active labels consistent with the decoding information fixed in the offline phase. The programming manipulates all permutation entries accessed to evaluate a garbled circuit. These entries are fixed by all active labels and RPM-based hash values in a transcript with padding. A programming is successful if and only if these entries have not been occupied by the query-response pairs before the online phase. This complements our definition of bad transcripts in addition to that in the npRPM.

A toy proof. Based on our methodology, we present a toy proof to demonstrate how to compute ε_1 and ε_2. Here, we consider half-gates and an adaptive adversary choosing a circuit f of an $\mathtt{AND}$ gate g with two input wires (a, b) and an output wire c. This proof can be generalized to three-halves and arbitrary circuits[4] chosen by the adversary.

For any fixed adaptive adversary $\mathcal{A}$ in transcript padding, a transcript $\tau \in \mathcal{T}_{\mathcal{A}}$ consists of offline part $(f, \widehat{f})$, online part $(x = (x_a, x_b), \widehat{x})$, query-response pairs w.r.t. the random permutation and its inverse, and free-XOR offset $\varDelta$ with $\mathsf{lsb}(\varDelta) = 1$. The query-response pairs are recorded before $\varDelta$ at any time and include the adaptive queries chosen by $\mathcal{A}$ and their responses.

In the npRPM, $\widehat{f}$ contains two gate ciphertexts (G_0, G_1) while $\widehat{x}$ fixes output bit $x_c = x_a \cdot x_b$, all active labels $\{X_i = W_i \oplus x_i \varDelta\}_{i \in \{a,b,c\}}$ for zero-bit labels $\{W_i\}_{i \in \{a,b,c\}}$, decoding information $d_c = x_c \oplus \mathsf{lsb}(X_c)$, and two RPM-based hash values

$$\begin{aligned} \mathsf{H}(X_a, k_0) &= \pi(X_a \oplus k_0) \oplus \sigma(X_a \oplus k_0), \\ \mathsf{H}(X_b, k_1) &= \pi(X_b \oplus k_1) \oplus \sigma(X_b \oplus k_1) \end{aligned} \tag{1}$$

for two distinct tweaks (k_0, k_1). Since $\mathsf{lsb}(\varDelta) = 1$, for each wire $i \in \{a, b, c\}$, *permuted bit* $p_i = \mathsf{lsb}(W_i)$ and *masked bit* $s_i = \mathsf{lsb}(X_i)$ satisfy $s_i = p_i \oplus x_i$. In the real world, oracle $\mathcal{O}_0$ samples (W_a, W_b) at random and computes

$$\begin{cases} G_0 &= \mathsf{H}(W_a, k_0) \oplus \mathsf{H}(W_a \oplus \varDelta, k_0) \oplus p_b \varDelta \\ G_1 &= \mathsf{H}(W_b, k_1) \oplus \mathsf{H}(W_b \oplus \varDelta, k_1) \oplus W_a \\ W_c &= \mathsf{H}(W_a \oplus p_a \varDelta, k_0) \oplus \mathsf{H}(W_b \oplus p_b \varDelta, k_1) \oplus p_a p_b \varDelta \end{cases} \tag{2}$$

and $\widehat{x}$ as above. In contrast, ideal-world oracle $\mathcal{O}_1$ samples (G_0, G_1) at random and computes $\widehat{x}$ as in the real world, except that (X_a, X_b) are uniformly sampled and

$$X_c = \mathsf{H}(X_a, k_0) \oplus \mathsf{H}(X_b, k_1) \oplus s_a G_0 \oplus s_b (G_1 \oplus X_a) \tag{3}$$

is equivalently written as the real-world one. In both worlds, the hash values in (1) are computed by calling the random permutation and consistent with two fixed query-response pairs for permutation pre-images $X_a \oplus k_0$ and $X_b \oplus k_1$.

Suppose that $\mathcal{A}$ makes q distinct queries to the random permutation and its inverse in addition to the above two query-response pairs. First, we bound ε_2 for some fixed $\tau \in \mathcal{T}_{\mathsf{good}} \subseteq \mathcal{T}_{\mathcal{A}}$. Without loss of generality, we can assume $\Pr[Z_1 = \tau] \neq 0$ in the ideal world; otherwise $\varepsilon_2 = 0$ trivially. In the real world, (2) fixes the values of two OTP masks as per the right-hand fixed values in transcript τ:

$$\begin{cases} \pi(X_a \oplus \varDelta \oplus k_0) \\ \quad = G_0 \oplus \sigma(\varDelta) \oplus (s_b \oplus x_b)\varDelta \oplus \pi(X_a \oplus k_0) \\ \pi(X_b \oplus \varDelta \oplus k_1) \\ \quad = G_1 \oplus \sigma(\varDelta) \oplus X_a \oplus x_a \varDelta \oplus \pi(X_b \oplus k_1) \end{cases} \tag{4}$$

while W_c in (2) must be consistent with the right-hand fixed values in τ as per (3). Using the pairwise distinctness in good transcripts, we have that τ fixes $q+4$ linkages between permutation pre-images and images. Taking over uniform (W_a, W_b), random permutation π, and uniform $\varDelta$ with $\mathsf{lsb}(\varDelta) = 1$, it holds that $\Pr[Z_0 = \tau] = \frac{1}{(2^\lambda)^2} \cdot \frac{(2^\lambda - q - 4)!}{(2^\lambda)!} \cdot \frac{1}{2^{\lambda-1}}$. In the ideal world, (G_0, G_1) are uniformly random and need not satisfy (4) to fix two linkages of the random permutation. Meanwhile, these two linkages are not fixed in good

[3] When being implemented, the three-halves scheme with the computational optimization would be preferred.

[4] For general circuits, we require that all the conditions of bad transcripts further hold for the relevant values across any two $\mathtt{AND}$ gates.

transcripts. So, $\Pr[Z_1 = \tau] = \frac{1}{(2^\lambda)^2} \cdot \frac{(2^\lambda - q - 2)!}{(2^\lambda)!} \cdot \frac{1}{(2^\lambda)^2} \cdot \frac{1}{2^{\lambda-1}}$ and $\varepsilon_2 = 0$ since

$$\frac{\Pr[Z_0 = \tau]}{\Pr[Z_1 = \tau]} = \frac{(2^\lambda)_{q+2} \cdot (2^\lambda)^2}{(2^\lambda)_{q+4}} = \frac{(2^\lambda)^2}{(2^\lambda - q - 2)_2} \geq 1.$$

Second, we claim that a negligible ε_1 bounds the probability of bad transcripts in the ideal world. This claim resorts to a negligible probability of pairwise collision and a negligible one of the existence of any $\varDelta$-dependent query-response pair. Intuitively, the former probability is taken over the (nearly) uniformly random pre-images (i.e., $X_a \oplus k_0$ or $X_b \oplus k_1$) or images (i.e., $\pi(X_a \oplus k_0)$ or $\pi(X_b \oplus k_1)$). The latter probability results from the entropy of $\varDelta$ since no query-response pair can depend on it.

This analysis of ε_1 and ε_2 concludes the adaptive security of half-gates in the npRPM. As for the adaptive security in the pRPM, transcripts are defined as in the npRPM, except that decoding information d_c is moved from online garbled input $\widehat{x}$ to offline garbled circuit $\widehat{f}$. Then, ε_1 increases due to the additional bad transcripts failing programming. Note that permutation entries $(X_a \oplus k_0, \pi(X_a \oplus k_0))$ and $(X_b \oplus k_1, \pi(X_b \oplus k_1))$ should never be occupied by the query-response pairs before the online phase to ensure a successful programming. It also follows from the randomness of these pre-images and images that the increased ε_1 is still negligible. For formal proofs, we refer readers to Section 4 for the half-gates in the npRPM, and Appendix A and B for other sketched proofs of adaptive security.

3. Preliminaries

3.1. Notation

Let PPT be short for "probabilistic polynomial-time". In this paper, we use $\lambda \in \mathbb{N}$ to denote the security parameter. We use $\mathsf{poly}(\cdot)$ (resp., $\mathsf{negl}(\cdot)$) for an unspecified polynomial (resp., negligible) function. For $a, b \in \mathbb{N}$ with $a \leq b$, we denote by $[a, b]$ the set $\{a, \ldots, b\}$ and by $(b)_a$ the falling factorial $b \cdot (b-1) \cdots (b-a+1)$. We use $x \leftarrow S$ to denote the uniform sampling of x from a finite set S. We use $:=$ to denote assigning a value or an output of a deterministic algorithm to a left-hand variable. Let $\mathcal{S}_\ell$ denote the set of permutations on $\{0,1\}^\ell$. Let $\mathsf{lsb}(x)$ denote the least significant bit (LSB) of $x \in \{0,1\}^n$. Let $\|$ denote the concatenation of bit-strings. Let $\ominus$ denote the symmetric difference of sets, i.e., for two sets A, B, $A \ominus B := (A \backslash B) \cup (B \backslash A)$.

Linear orthomorphism. A permutation $\sigma : \mathbb{G} \to \mathbb{G}$ over an additive Abelian group $\mathbb{G}$ is a linear orthomorphism if (i) $\sigma(x+y) = \sigma(x) + \sigma(y)$ for any $x, y \in \mathbb{G}$, (ii) $\sigma'(x) := \sigma(x) - x$ is also a permutation, and (iii) σ, σ' and their inverses are efficiently computable. There are two simple instantiations in [25]: (i) $\mathbb{G}$ is a field: $\sigma(x) := c \cdot x$ where $c \neq 0, 1 \in \mathbb{G}$, or (ii) $\mathbb{G} = \{0,1\}^n$: $\sigma(x) := (x_L \oplus x_R) \| x_L$ where x_L and x_R are the left and right halves of x.

Circuits. For a circuit f with fan-in two and fan-out one:

- $|f|$: The number of AND gates.
- $\mathcal{W}(f)$, $\mathcal{W}_{\mathsf{in}}(f)$, $\mathcal{W}_{\mathsf{out}}(f)$, $\mathcal{W}_{\mathsf{and}}(f)$: The sets of wires, circuit input wires, circuit output wires, and output wires of AND gates, resp.
- $\mathcal{G}(f)$, $\mathcal{G}_{\mathsf{and}}(f)$: The sets of gates and AND gates, resp.
- For a gate $g \in \mathcal{G}(f)$, let $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$ be the two input wires and $c := \mathsf{out}(g)$ be the output wire.

3.2. Random Permutation Model

In the random permutation model (RPM) [32], [24], all parties have oracle access to random permutation π and its inverse $\pi^{-1} := \mathsf{inv}(\pi)$. We refer to queries to π as *forward queries* and queries to π^{-1} as *backward queries*. In this work, we consider *non-programmable RPM* (npRPM) and *programmable RPM* (pRPM). Inspired by the separation [46] w.r.t. the random oracle model [47], we formalize the pRPM like a hybrid model, where an ideal functionality provides two $\pi^{\pm 1}$ interfaces. This model gives the simulator the power to "appropriately" choose the responses to the adversary's queries to $\pi^{\pm 1}$. In contrast, all parties (as well as the real-world adversary and the simulator) in the npRPM are given oracle access to a global $\pi^{\pm 1}$, whose responses cannot be chosen by the simulator.

3.3. Adaptive Security of Garbling Schemes

In Definition 1, we adapt the definition of adaptively secure garbling schemes in [27], [28] for a q-query *computationally unbounded* adversary in the npRPM and the pRPM. This definition combines the evaluation and decoding algorithms in the literature [26], [2] and does not explicitly send the decoding table as part of a garbled circuit $\widehat{f}$. To simplify the notation, we assume that all algorithms and the adversary implicitly take the unary security parameter 1^λ as input.

Definition 1 (Adaptively secure garbling scheme). *For some polynomial ℓ, an $\ell(\lambda)$-garbling scheme in the npRPM or pRPM has three PPT algorithms, each of which is given oracle access to a random permutation $\pi \in \mathcal{S}_{\ell(\lambda)}$ and its inverse $\pi^{-1} := \mathsf{inv}(\pi)$:*

- $(\widehat{f}, k) \leftarrow \mathsf{Garble}^{\pi^{\pm 1}(\cdot)}(f)$. $|\widehat{f}|$ *is called* ***offline complexity***.
- $\widehat{x} := \mathsf{Encode}^{\pi^{\pm 1}(\cdot)}(k, x)$. $|\widehat{x}|$ *is called* ***online complexity***.
- $y := \mathsf{DecEval}^{\pi^{\pm 1}(\cdot)}(\widehat{f}, \widehat{x})$.

For polynomials q, s, negligible function ε, and side-information function $\varPhi$, this scheme is $(q(\lambda), s(\lambda), \varepsilon(\lambda), \varPhi)$-adaptively secure in the npRPM (resp., pRPM) if it complies with ***correctness*** *and* ***adaptive security in the npRPM (resp., pRPM)****. By default, $\varPhi(f) = f$ is omitted in the definition.*

- ***Correctness.*** *For every polynomial-size circuit $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$, and every input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$,*

$$\Pr\left[\begin{array}{l} \pi \leftarrow \mathcal{S}_{\ell(\lambda)}, \pi^{-1} := \mathsf{inv}(\pi), \\ (\widehat{f}, k) \leftarrow \mathsf{Garble}^{\pi^{\pm 1}(\cdot)}(f), : \mathsf{DecEval}^{\pi^{\pm 1}(\cdot)}(\widehat{f}, \widehat{x}) = f(x) \\ \widehat{x} := \mathsf{Encode}^{\pi^{\pm 1}(\cdot)}(k, x) \end{array}\right] = 1.$$

- ***Adaptive security in the npRPM.*** *There exists a PPT simulator $\mathsf{Sim} = (\mathsf{SimF}, \mathsf{SimIn})$ with an internal state $\mathsf{st}_{\mathsf{sim}}$ such that, for every auxiliary input $z \in \{0,1\}^*$ and every computationally unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*

totally making $q(\lambda)$ oracle queries and choosing a circuit with $s(\lambda)$ `AND` gates,

$$\left| \Pr\left[\begin{array}{r} \pi \leftarrow \mathcal{S}_{\ell(\lambda)}, \pi^{-1} := \mathsf{inv}(\pi), \\ (f, \mathsf{st}_1) \leftarrow {\mathcal{A}_1}^{\pi^{\pm 1}(\cdot)}(z), \\ (\widehat{f}, k) \leftarrow \mathsf{Garble}^{\pi^{\pm 1}(\cdot)}(f), \\ (x, \mathsf{st}_2) \leftarrow {\mathcal{A}_2}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} := \mathsf{Encode}^{\pi^{\pm 1}(\cdot)}(k, x) \end{array} : {\mathcal{A}_3}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \right] - \Pr\left[\begin{array}{r} \pi \leftarrow \mathcal{S}_{\ell(\lambda)}, \pi^{-1} := \mathsf{inv}(\pi), \\ (f, \mathsf{st}_1) \leftarrow {\mathcal{A}_1}^{\pi^{\pm 1}(\cdot)}(z), \\ \widehat{f} \leftarrow \mathsf{SimF}^{\pi^{\pm 1}(\cdot)}(\varPhi(f)), \\ (x, \mathsf{st}_2) \leftarrow {\mathcal{A}_2}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} \leftarrow \mathsf{SimIn}^{\pi^{\pm 1}(\cdot)}(f(x)) \end{array} : {\mathcal{A}_3}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \right] \right|$$

is at most $\varepsilon(\lambda)$.

- ***Adaptive security in the pRPM.*** *There exists a PPT simulator $\mathsf{Sim} = (\mathsf{SimF}, \mathsf{SimIn}, \mathsf{SimP}^{\pm 1})$ with an internal state $\mathsf{st}_{\mathsf{sim}}$ such that, for every auxiliary input $z \in \{0,1\}^*$ and every computationally unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ totally making $q(\lambda)$ oracle queries and choosing a circuit with $s(\lambda)$ `AND` gates,*

$$\left| \Pr\left[\begin{array}{r} \pi \leftarrow \mathcal{S}_{\ell(\lambda)}, \pi^{-1} := \mathsf{inv}(\pi), \\ (f, \mathsf{st}_1) \leftarrow {\mathcal{A}_1}^{\pi^{\pm 1}(\cdot)}(z), \\ (\widehat{f}, k) \leftarrow \mathsf{Garble}^{\pi^{\pm 1}(\cdot)}(f), \\ (x, \mathsf{st}_2) \leftarrow {\mathcal{A}_2}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} := \mathsf{Encode}^{\pi^{\pm 1}(\cdot)}(k, x) \end{array} : {\mathcal{A}_3}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \right] - \Pr\left[\begin{array}{r} (f, \mathsf{st}_1) \leftarrow \mathcal{A}_1^{\mathsf{SimP}^{\pm 1}(\cdot)}(z), \\ \widehat{f} \leftarrow \mathsf{SimF}(\varPhi(f)), \\ (x, \mathsf{st}_2) \leftarrow \mathcal{A}_2^{\mathsf{SimP}^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} \leftarrow \mathsf{SimIn}(f(x)) \end{array} : \mathcal{A}_3^{\mathsf{SimP}^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \right] \right|$$

is at most $\varepsilon(\lambda)$.

3.4. A Statistical Framework of Adaptive Security and H-coefficient Technique

We prove the adaptive security of garbling schemes in the following framework of adaptive experiments. In this framework, we consider a computationally unbounded non-uniform adversary $\mathcal{A}$, which takes an auxiliary input and outputs a decision bit after making a bounded number of adaptive queries to either a real-world or ideal-world oracle. Each oracle is *stateful*: for a query, a response is a *deterministic* function of the random tape, the query, and previous query-response pairs. The two oracles can give multiple interfaces, each of which has the same syntax in the two worlds. For simplicity, whenever we refer to a non-uniform adversary (e.g., $\mathcal{A}$ or $\mathcal{A}'$), we assume that it has some auxiliary input fixed in its context unless this auxiliary input is given explicitly.

Without loss of generality, we consider a *deterministic* computationally unbounded non-uniform adversary $\mathcal{A}$. The interaction between $\mathcal{A}$ and the oracle produces a transcript, which gives an *ordered* list of query-response pairs from the view of $\mathcal{A}$. For some fixed $\mathcal{A}$, the distribution of transcripts in either world results from the oracle's random tape, or equivalently, the random sampling of a *deterministic* oracle. So, the decision bit of $\mathcal{A}$ is a deterministic function of its (fixed) auxiliary input and a transcript produced in the interaction, and its advantage is at most the statistical distance of transcripts in the two worlds.

Let $\varOmega_{\mathsf{real}}$ (resp., $\varOmega_{\mathsf{ideal}}$) denote the sample space where a *deterministic* real-world (resp., ideal-world) oracle is sampled at random. For any fixed $\mathcal{A}$, let $\mathcal{T}_{\mathcal{A}}$ denote the set of *attainable* transcripts s.t. a transcript $\tau \in \mathcal{T}_{\mathcal{A}}$ if and only if there exists a *deterministic*[5] oracle ω' such that the interaction between $\mathcal{A}$ and ω' produces τ. Let $X : \varOmega_{\mathsf{real}} \to \mathcal{T}_{\mathcal{A}}$ (resp., $Y : \varOmega_{\mathsf{ideal}} \to \mathcal{T}_{\mathcal{A}}$) denote the random variable w.r.t. the transcripts produced in the interaction between $\mathcal{A}$ and a real-world (resp., ideal-world) oracle ω sampled from $\varOmega_{\mathsf{real}}$ (resp., $\varOmega_{\mathsf{ideal}}$). For any fixed τ, let $\mathsf{comp}_{\mathsf{real}}(\tau) \subseteq \varOmega_{\mathsf{real}}$ (resp., $\mathsf{comp}_{\mathsf{ideal}}(\tau) \subseteq \varOmega_{\mathsf{ideal}}$) denote the set of *compatible* real-world (resp., ideal-world) oracles s.t. an oracle $\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)$ (resp., $\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)$) if and only if there exists a *deterministic* non-uniform adversary $\mathcal{A}'$ such that the interaction between $\mathcal{A}'$ and ω produces τ.

A key observation is that the adaptive interaction in random variables X, Y can be "unfolded" to be equivalently but easily studied from the compatibility between oracle and transcript. This compatibility, as defined above, essentially means that an oracle will returns the responses in a fixed transcript τ in order if the queries match their counterparts in τ and are sent (by $\mathcal{A}'$) to the oracle in the given order. This observation is formalized in Lemma 1, and its proof for stateful oracles was sketched in [38, Appendix D].

Lemma 1 ([38]). *Let the notations be defined in Section 3.4. Then, for every auxiliary input $z \in \{0,1\}^*$, every computationally unbounded adversary $\mathcal{A}$, and every attainable transcript $\tau \in \mathcal{T}_{\mathcal{A}(z)}$, it holds that*

$$\Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[X(\omega) = \tau] = \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)],$$
$$\Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[Y(\omega) = \tau] = \Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)].$$

For every fixed non-uniform $\mathcal{A}$, the H-coefficient technique [36], [37], [38] bounds the statistical distance of transcripts in the experiment. It divides $\mathcal{T}_{\mathcal{A}}$ into two disjoint subsets $\mathcal{T}_{\mathsf{bad}} \subseteq \mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}_{\mathsf{good}} := \mathcal{T}_{\mathcal{A}} \setminus \mathcal{T}_{\mathsf{bad}}$. Then, it can prove an upper bound $\varepsilon_1 + \varepsilon_2$ of this statistical distance if

$$\Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[Y(\omega) \in \mathcal{T}_{\mathsf{bad}}] \leq \varepsilon_1,$$
$$\forall \tau \in \mathcal{T}_{\mathsf{good}} : \frac{\Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[X(\omega) = \tau]}{\Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[Y(\omega) = \tau]} \geq 1 - \varepsilon_2,$$

where, for some $\tau \in \mathcal{T}_{\mathsf{good}}$ with $\Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[Y(\omega) = \tau] = 0$, we can define $\varepsilon_2 = 0$.

4. Adaptive Security of Half-Gates in npRPM

We prove that half-gates is adaptively secure in the npRPM. This scheme can be implemented in Figure 2, which slightly differs from the original one of [22] in including decoding table d in garbled input $\widehat{x}$ rather than garbled circuit $\widehat{f}$. We will argue in Appendix C that this

[5] Without loss of generality, we assume that such a deterministic $\omega' \in \varOmega_{\mathsf{real}} \cup \varOmega_{\mathsf{ideal}}$. Otherwise (no such an ω'), the statistical distance is zero.

```
$\underline{\mathsf{HG.Garble}^{\pi^{\pm 1}(\cdot)}(f)}$:
$\Delta \leftarrow \{0,1\}^{\lambda-1} \,\|\, 1$
for $i \in \mathcal{W}_{\mathsf{in}}(f)$ do
    $W_i \leftarrow \{0,1\}^{\lambda}$
for $g \in \mathcal{G}(f)$ in order do
    $(a,b,c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
    if $\mathsf{type}(g) = \mathtt{XOR}$ then $W_c := W_a \oplus W_b$
    else if $\mathsf{type}(g) = \mathtt{AND}$ then
        $k_0^g := 2 \cdot g - 1,\ k_1^g := 2 \cdot g$
        $p_a := \mathsf{lsb}(W_a),\ p_b := \mathsf{lsb}(W_b)$
        $G_0^g := \mathsf{H}(W_a, k_0^g) \oplus \mathsf{H}(W_a \oplus \Delta, k_0^g) \oplus p_b\Delta$
        $G_1^g := \mathsf{H}(W_b, k_1^g) \oplus \mathsf{H}(W_b \oplus \Delta, k_1^g) \oplus W_a$
        $W_c := \mathsf{H}(W_a \oplus p_a\Delta, k_0^g) \oplus \mathsf{H}(W_b \oplus p_b\Delta, k_1^g) \oplus p_a p_b \Delta$
for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $d_i := \mathsf{lsb}(W_i)$
return $\widehat{f} := (f' := f, F := \{(G_0^g, G_1^g)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}),$
        $k := (f, d, \Delta, W)$

$\underline{\mathsf{HG.Encode}^{\pi^{\pm 1}(\cdot)}(k,x)}$:
Parse $k = (f, d, \Delta, W)$
for $i \in \mathcal{W}_{\mathsf{in}}(f)$ do
    $X_i := W_i \oplus x_i\Delta$
return $\widehat{x} := (\{X_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, d)$

$\underline{\mathsf{HG.DecEval}^{\pi^{\pm 1}(\cdot)}(\widehat{f}, \widehat{x})}$:
Parse $\widehat{f} = (f, \{(G_0^g, G_1^g)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}),\ \widehat{x} = (\{X_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, d)$
for $g \in \mathcal{G}(f)$ in order do
    $(a,b,c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
    if $\mathsf{type}(g) = \mathtt{XOR}$ then $X_c := X_a \oplus X_b$
    else if $\mathsf{type}(g) = \mathtt{AND}$ then
        $k_0^g := 2 \cdot g - 1,\ k_1^g := 2 \cdot g$
        $s_a := \mathsf{lsb}(X_a),\ s_b := \mathsf{lsb}(X_b)$
        $X_c := \mathsf{H}(X_a, k_0^g) \oplus \mathsf{H}(X_b, k_1^g) \oplus s_a G_0^g \oplus s_b(G_1^g \oplus X_a)$
for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $y_i := d_i \oplus \mathsf{lsb}(X_i)$
return $y$
```

Figure 2: Half-gates garbling scheme [22].

```
$\underline{\mathsf{SimF}^{\pi^{\pm 1}(\cdot)}(f)}$:
$F := \{(G_0^g, G_1^g)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)} \leftarrow (\{0,1\}^{2\lambda})^{|f|}$
$\{X_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)} \leftarrow (\{0,1\}^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}$
for $g \in \mathcal{G}(f)$ in order do
    $(a,b,c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
    if $\mathsf{type}(g) = \mathtt{XOR}$ then $X_c := X_a \oplus X_b$
    else if $\mathsf{type}(g) = \mathtt{AND}$ then
        $k_0^g := 2 \cdot g - 1,\ k_1^g := 2 \cdot g$
        $s_a := \mathsf{lsb}(X_a),\ s_b := \mathsf{lsb}(X_b)$
        $U_0^g := \pi(X_a \oplus k_0^g) \oplus \sigma(X_a \oplus k_0^g),$
        $U_1^g := \pi(X_b \oplus k_1^g) \oplus \sigma(X_b \oplus k_1^g)$
        $X_c := U_0^g \oplus U_1^g \oplus s_a G_0^g \oplus s_b(G_1^g \oplus X_a)$
return $\widehat{f} := (f, F),$
        $\mathsf{st}_{\mathsf{sim}} := (f, \widetilde{X} := \{X_i\}_{i \in \mathcal{W}(f)}, \widetilde{U} := \{U_0^g, U_1^g\}_{g \in \mathcal{G}_{\mathsf{and}}(f)})$

$\underline{\mathsf{SimIn}^{\pi^{\pm 1}(\cdot)}(f(x))}$:
Parse $\mathsf{st}_{\mathsf{sim}} = (f, \widetilde{X} = \{X_i\}_{i \in \mathcal{W}(f)}, \widetilde{U})$
for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $d_i := f(x)_i \oplus \mathsf{lsb}(X_i)$
return $\widehat{x} := (\{X_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, d),\ \widetilde{X},\ \widetilde{U}.$
```

Figure 3: Our simulator for half-gates in the npRPM.

difference is required to follow the online-complexity lower bound in the npRPM.

Theorem 1. *Let $\mathsf{H}(X,k) = \pi(X \oplus k) \oplus \sigma(X \oplus k)$ be a tweakable hash function where $X, k \in \{0,1\}^{\lambda}$, $\pi \in \mathcal{S}_{\lambda}$ is random permutation, and $\sigma : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ is a linear orthomorphism. Then, half-gates (Figure 2) is a λ-garbling scheme with (q,s,ε)-adaptive security in the npRPM, where $\varepsilon = (16qs + 38s^2)/2^{\lambda}$.*

Proof. The correctness is given by the proof [22] as postponing decoding table d does not affect correctness. We focus on the simulation. Our simulator $\mathsf{Sim} = (\mathsf{SimF}, \mathsf{SimIn})$ is given in Figure 3 and is obviously PPT. Then, we prove this theorem using the following three hybrids:

- Hybrid_0. This is the adaptive experiment using Sim.
- Hybrid_1. This is identical to the previous hybrid, except that we replace $\pi^{\pm 1}$ (which can be equivalently emulated on-the-fly as in Figure 4) by an approximation $\widetilde{\pi}^{\pm 1}$ (given in Figure 5). This approximation is the same as random permutation except that, for a new query of the simulator, it returns a fresh random string as response and records this query-response pair. This hybrid is used to simplify probability analysis.
- Hybrid_2. This is the adaptive experiment using half-gates.

Using the following Corollary 1 for the indistinguishability between Hybrid_0 and Hybrid_1 and Lemma 3 for that between Hybrid_1 and Hybrid_2, this theorem holds. □

Before giving the proofs of Corollary 1 and Lemma 3, we prove the following lemma.

Lemma 2. *Let $\widetilde{\mathcal{P}}_{\ell(\lambda)}$ denote the distribution of $\widetilde{\pi}^{\pm 1}$ in Figure 5 for $n(\lambda) \in \mathbb{N}^{+}$ queries. For every PPT simulator $\mathsf{Sim} = (\mathsf{SimF}, \mathsf{SimIn})$ making $n_{\mathsf{sim}}(\lambda) \leq n(\lambda)$ queries, every auxiliary input $z \in \{0,1\}^{*}$, and every computationally unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ making $n(\lambda) - n_{\mathsf{sim}}(\lambda)$ queries, it holds that*

$$\left| \Pr\left[\begin{array}{c} \pi \leftarrow \mathcal{S}_{\ell(\lambda)}, \pi^{-1} := \mathsf{inv}(\pi), \\ (f, \mathsf{st}_1) \leftarrow \mathcal{A}_1{}^{\pi^{\pm 1}(\cdot)}(z), \\ \widehat{f} \leftarrow \mathsf{SimF}^{\pi^{\pm 1}(\cdot)}(\Phi(f)), : \mathcal{A}_3{}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \\ (x, \mathsf{st}_2) \leftarrow \mathcal{A}_2{}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} \leftarrow \mathsf{SimIn}^{\pi^{\pm 1}(\cdot)}(f(x)) \end{array} \right] - \Pr\left[\begin{array}{c} \widetilde{\pi}^{\pm 1} \leftarrow \widetilde{\mathcal{P}}_{\ell(\lambda)}, \\ (f, \mathsf{st}_1) \leftarrow \mathcal{A}_1{}^{\widetilde{\pi}^{\pm 1}(\cdot)}(z), \\ \widehat{f} \leftarrow \mathsf{SimF}^{\widetilde{\pi}^{\pm 1}(\cdot)}(\Phi(f)), : \mathcal{A}_3{}^{\widetilde{\pi}^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1 \\ (x, \mathsf{st}_2) \leftarrow \mathcal{A}_2{}^{\widetilde{\pi}^{\pm 1}(\cdot)}(\mathsf{st}_1, \widehat{f}), \\ \widehat{x} \leftarrow \mathsf{SimIn}^{\widetilde{\pi}^{\pm 1}(\cdot)}(f(x)) \end{array} \right] \right|$$

is at most $(n(\lambda) \cdot n_{\mathsf{sim}}(\lambda))/2^{\ell(\lambda)-1}$.

Proof. Let $\mathcal{N} \subseteq [1, n(\lambda)]$ denote the index set of the queries made by Sim such that $|\mathcal{N}| = n_{\mathsf{sim}}(\lambda)$. Let $\mathsf{true}(\pi)$ denote the event that $\mathcal{A}_3{}^{\pi^{\pm 1}(\cdot)}(\mathsf{st}_2, \widehat{f}, \widehat{x}) = 1$ in the first experiment, $\mathsf{true}(\widetilde{\pi})$ denote the counterpart in the second one, and bad denote the event that $\vee_{i \in \mathcal{N}}((\ldots, c_i) \in \mathcal{Q}_{i-1} \vee (c_i, \ldots) \in \mathcal{Q}_{i-1})$. We can study the two experiments under the same probability space, i.e., they use the same literal values of (r_{π}, r_{π}^{*}) and the random tapes of Sim and $\mathcal{A}$.

We can see that $\mathsf{true}(\pi) \wedge \neg\mathsf{bad}$ occurs if and only if $\mathsf{true}(\widetilde{\pi}) \wedge \neg\mathsf{bad}$ occurs, i.e., the two experiments proceed identically unless bad occurs. Thus, it follows from the Difference Lemma that

$$\left| \Pr[\mathsf{true}(\pi)] - \Pr[\mathsf{true}(\widetilde{\pi})] \right| \leq \Pr[\mathsf{bad}] \leq \sum_{i \in \mathcal{N}} \frac{2\,|\mathcal{Q}_{i-1}|}{2^{\ell(\lambda)}} \leq \frac{n(\lambda) \cdot n_{\mathsf{sim}}(\lambda)}{2^{\ell(\lambda)-1}},$$

which completes this proof. □

```
1:  Initialize a list $\mathcal{Q}_0 = \varnothing$.
2:  Sample[a] $c_1, \ldots, c_{n(\lambda)} \leftarrow \{0,1\}^{\ell(\lambda)}$.
3:  for $i \in [1, n(\lambda)]$ do
4:      if $\pi$ is queried with $\alpha_i \in \{0,1\}^{\ell(\lambda)}$ then
5:          if $\exists(\alpha_i, \gamma_i) \in \mathcal{Q}_{i-1}$ then Return $\gamma_i$ as response.
6:          if $(\ldots, c_i) \notin \mathcal{Q}_{i-1}$ then $\gamma_i := c_i$.
7:          else Sample[b] $\gamma_i \leftarrow \{s_i \in \{0,1\}^{\ell(\lambda)} \mid (\ldots, s_i) \notin \mathcal{Q}_{i-1}\}$.
8:          Return $\gamma_i$ as response and $\mathcal{Q}_i := \mathcal{Q}_{i-1} \cup \{(\alpha_i, \gamma_i)\}$.
9:      else if $\pi^{-1}$ is queried with $\beta_i \in \{0,1\}^{\ell(\lambda)}$ then
10:         if $\exists(\gamma_i, \beta_i) \in \mathcal{Q}_{i-1}$ then Return $\gamma_i$ as response.
11:         if $(c_i, \ldots) \notin \mathcal{Q}_{i-1}$ then $\gamma_i := c_i$.
12:         else Sample[b] $\gamma_i \leftarrow \{s_i \in \{0,1\}^{\ell(\lambda)} \mid (s_i, \ldots) \notin \mathcal{Q}_{i-1}\}$.
13:         Return $\gamma_i$ as response and $\mathcal{Q}_i := \mathcal{Q}_{i-1} \cup \{(\gamma_i, \beta_i)\}$.

    a. A uniform random tape $r_\pi$ is used.
    b. A uniform random tape $r_\pi^*$ is used to run rejection sampling.
```

Figure 4: The workflow of oracle $\pi^{\pm 1}(\cdot)$ up to $n(\lambda)$ queries.

```
1:  Initialize a list $\mathcal{Q}_0 = \varnothing$.
2:  Sample uniform $c_1, \ldots, c_{n(\lambda)} \leftarrow \{0,1\}^{\ell(\lambda)}$.
3:  for $i \in [1, n(\lambda)]$ do
4:      if $\widetilde{\pi}$ is queried with $\alpha_i \in \{0,1\}^{\ell(\lambda)}$ from Sim then
5:          if $\exists(\alpha_i, \gamma_i) \in \mathcal{Q}_{i-1}$ then Return $\gamma_i$ as response.
6:          Return $\gamma_i := c_i$ as response and $\mathcal{Q}_i := \mathcal{Q}_{i-1} \cup \{(\alpha_i, \gamma_i)\}$.
7:      else if $\widetilde{\pi}^{-1}$ is queried with $\beta_i \in \{0,1\}^{\ell(\lambda)}$ from Sim then
8:          if $\exists(\gamma_i, \beta_i) \in \mathcal{Q}_{i-1}$ then Return $\gamma_i$ as response.
9:          Return $\gamma_i := c_i$ as response and $\mathcal{Q}_i := \mathcal{Q}_{i-1} \cup \{(\gamma_i, \beta_i)\}$.
10:     else if $\widetilde{\pi}$ is queried with $\alpha_i \in \{0,1\}^{\ell(\lambda)}$ $\boxed{\text{from } \mathcal{A}}$ then
11:         Same as step 5 to 8 in oracle $\pi^{\pm 1}(\cdot)$ (Figure 4).
12:     else if $\widetilde{\pi}^{-1}$ is queried with $\beta_i \in \{0,1\}^{\ell(\lambda)}$ $\boxed{\text{from } \mathcal{A}}$ then
13:         Same as step 10 to 13 in oracle $\pi^{\pm 1}(\cdot)$ (Figure 4).
```

Figure 5: The workflow of approximate oracle $\widetilde{\pi}^{\pm 1}(\cdot)$ up to $n(\lambda)$ queries, where the differences are highlighted in box.

Corollary 1. *Let* Sim *be defined as in Figure 3. Then, for every auxiliary input $z \in \{0,1\}^*$ and every computationally unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ totally making q oracle queries and choosing a circuit with s `AND` gates, $\mathcal{A}(z)$ can distinguish* Hybrid_0 *and* Hybrid_1 *with advantage at most* $(2qs + 4s^2)/2^{\lambda-1}$.

Proof. This corollary follows from Lemma 2 by using $\ell(\lambda) = \lambda$, $n_{\mathsf{sim}}(\lambda) = 2\,|f|$, and $n(\lambda) = q + n_{\mathsf{sim}}(\lambda)$, given q queries of $\mathcal{A}$ and $2\,|f| = 2s$ queries of Sim. □

Then, we use the H-coefficient technique (Section 3.4) with "transcript padding" to bound the advantage of distinguishing Hybrid_1 and Hybrid_2.

Lemma 3. *Let* Sim *be defined as in Figure 3. Then, for every auxiliary input $z \in \{0,1\}^*$ and every computationally unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ totally making q oracle queries and choosing a circuit with s `AND` gates, $\mathcal{A}(z)$ can distinguish* Hybrid_1 *and* Hybrid_2 *with advantage at most* $(12qs + 30s^2)/2^{\lambda}$.

Proof. Fix z and $\mathcal{A}$. We regard Hybrid_1 (resp., Hybrid_2) as the ideal (resp., real) world in the H-coefficient technique, where the computationally unbounded non-uniform adversary $\mathcal{A} = \mathcal{A}(z)$ and $\varepsilon_1, \varepsilon_2$ are computed as follows.

Transcript padding. In either world, $\mathcal{A}$ will interact with an integrated oracle that acts as the two-round challenger in the adaptive experiment and provides interfaces $\pi^{*\pm 1} \in \{\pi^{\pm 1}, \widetilde{\pi}^{\pm 1}\}$ for forward/backward permutation queries. Here, $\mathcal{A}$ can learn $\pi^*(\alpha) = \beta$ if and only if it sent forward query α to π^* and received response β, or sent backward query β to π^{*-1} and received response α.

To compute $\varepsilon_1, \varepsilon_2$, we ask the oracle to send more messages to $\mathcal{A}$ and $\mathcal{A}$ to make extra queries (in addition to the supposed q queries) in both worlds. More specifically,

- Upon receiving x from $\mathcal{A}$, the oracle sends $(\widetilde{X} := \{X_i\}_{i \in \mathcal{W}(f)}, d)$ rather than $\widehat{x} := (\{X_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, d)$ to $\mathcal{A}$. In addition to the active input labels given by $\widehat{x}$, the former also gives the active internal and output labels. In the real world, the oracle can run $\mathsf{HG.DecEval}^{\pi^{\pm 1}(\cdot)}$, which determines other active labels in $\widetilde{X}$. In the ideal world, this $\widetilde{X}$ can be directly output by SimIn.
- Along with $(\widetilde{X}, d)$, the oracle sends $\widetilde{x} := \{x_i\}_{i \in \mathcal{W}(f)}$ to $\mathcal{A}$, which are the wire truth values in the evaluation of $f(x)$. Both two oracles "echo" these values, which are self-evident to $\mathcal{A}$, to explicitly include them in transcripts. In the experiment, the real-world oracle uses $x = \{x_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}$ in $\mathsf{HG.Encode}^{\pi^{\pm 1}(\cdot)}$, but the ideal-world oracle can only use $f(x) = \{x_i\}_{i \in \mathcal{W}_{\mathsf{out}}(f)}$ in SimIn.
- Along with $(\widetilde{X}, d)$, the oracle sends $\widetilde{U} := \{U_0^g, U_1^g\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$ to $\mathcal{A}$. In the real world, the oracle computes $U_0^g := \mathsf{H}(X_a, k_0^g)$ and $U_1^g := \mathsf{H}(X_b, k_1^g)$ for each $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$. In the ideal world, this $\widetilde{U}$ is output by SimIn and is essentially the same hash outputs.
- **(Extra queries)** Upon receiving $(\widetilde{X}, d, \widetilde{x}, \widetilde{U})$ from the oracle, $\mathcal{A}$ also makes a forward permutation query $X_a \oplus k_0^g$ (resp., $X_b \oplus k_1^g$) for each $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$, if it has never learned $\pi^*(X_a \oplus k_0^g) = Y$ (resp., $\pi^*(X_b \oplus k_1^g) = Y$) for some Y in its interaction with $\pi^{*\pm 1} \in \{\pi^{\pm 1}, \widetilde{\pi}^{\pm 1}\}$.
- At the end of the experiment (i.e., once all other transcripts are settled), the oracle sends $\varDelta$ to $\mathcal{A}$. In the real world, the oracle gets this $\varDelta$ from the output of $\mathsf{HG.Garble}^{\pi^{\pm 1}(\cdot)}$. In the ideal world, $\varDelta$ is dummy and sampled by the oracle at this time (Sim does not use $\varDelta$).

According to the two oracles, real-world sample space

$$\varOmega_{\mathsf{real}} = \{0,1\}^{\lambda-1} \times \{0,1\}^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda} \times \mathcal{S}_\lambda,$$

and ideal-world sample space

$$\varOmega_{\mathsf{ideal}} = (\{0,1\}^{2\lambda})^{|f|} \times (\{0,1\}^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|} \times \underbrace{\{0,1\}^*}_{\substack{\text{random tape for}\\ \text{the sampling in } \widetilde{\pi}^{\pm 1}(\cdot)}} \times \underbrace{\{0,1\}^{\lambda-1}}_{\text{dummy } \varDelta}.$$

Given the oracle constructions, a transcript in the original adaptive experiment will be padded with more literal values. Note that transcript padding will not lower the advantage of $\mathcal{A}$ since $\mathcal{A}$ can discard the padding values at will. With the padding, a transcript is of the form:

$$\tau = (\mathcal{K}_1, (f, \widehat{f}), \mathcal{K}_2, (x, (\widetilde{X}, d, \widetilde{x}, \widetilde{U})), \mathcal{K}_3, \varDelta),$$

where $\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{K}_3$ are the ordered lists of query-response pairs seen in the interleaved interaction with permutation oracles. We do not explicitly consider query direction in these pairs. Given $\pi^{*\pm 1} \in \{\pi^{\pm 1}, \widetilde{\pi}^{\pm 1}\}$, $\mathcal{A}$ learns $\pi^*(\alpha) = \beta$ if and only if there exists $(\alpha, \beta) \in \cup_{\ell=1}^{3} \mathcal{K}_\ell$.

Let $q_\ell := |\mathcal{K}_\ell|$ for every $\ell \in \{1, 2, 3\}$ and $q_\Sigma := \sum_{\ell=1}^{3} q_\ell$. It follows from the extra queries that $q_\Sigma \leq q + 2\,|f|$. Without loss of generality, we assume that $\mathcal{A}$ only makes *non-repeating* queries, i.e., it never makes forward query α to π^* or backward query β to π^{*-1} for any learned permutation entry (α, β).

Bad transcripts. A transcript $\tau \in \mathcal{T}_{\mathsf{bad}}$ if and only if it incurs at least one of the following events[6]:

- bad_1. There exist distinct $(g, u), (g', u') \in \mathcal{G}_{\mathsf{and}}(f) \times \{0, 1\}$ such that

$$\begin{aligned} & X_w \oplus k_u^g = X_{w'} \oplus k_{u'}^{g'} && (5) \\ \vee \quad & U_u^g \oplus \sigma(X_w \oplus k_u^g) = U_{u'}^{g'} \oplus \sigma(X_{w'} \oplus k_{u'}^{g'}) && (6) \end{aligned}$$

  where $w := \mathsf{in}_u(g)$ and $w' := \mathsf{in}_{u'}(g')$, or
  There exists $g \in \mathcal{G}_{\mathsf{and}}(f)$ such that

$$\begin{aligned} & (s_b \oplus x_b)\Delta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g) \\ & \quad = x_a \Delta \oplus X_a \oplus G_1^g \oplus U_1^g \oplus \sigma(X_b \oplus k_1^g) \qquad (7) \end{aligned}$$

  where $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$, or
  There exist distinct $g, g' \in \mathcal{G}_{\mathsf{and}}(f)$ such that, for $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$ and $(a', b') := (\mathsf{in}_0(g'), \mathsf{in}_1(g'))$,

$$\begin{aligned} & (s_b \oplus x_b)\Delta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g) \\ & \quad = (s_{b'} \oplus x_{b'})\Delta \oplus G_0^{g'} \oplus U_0^{g'} \oplus \sigma(X_{a'} \oplus k_0^{g'}) && (8) \\ \vee \quad & x_a \Delta \oplus X_a \oplus G_1^g \oplus U_1^g \oplus \sigma(X_b \oplus k_1^g) \\ & \quad = x_{a'} \Delta \oplus X_{a'} \oplus G_1^{g'} \oplus U_1^{g'} \oplus \sigma(X_{b'} \oplus k_1^{g'}) && (9) \\ \vee \quad & (s_b \oplus x_b)\Delta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g) \\ & \quad = x_{a'} \Delta \oplus X_{a'} \oplus G_1^{g'} \oplus U_1^{g'} \oplus \sigma(X_{b'} \oplus k_1^{g'}) && (10) \\ \vee \quad & x_a \Delta \oplus X_a \oplus G_1^g \oplus U_1^g \oplus \sigma(X_b \oplus k_1^g) \\ & \quad = (s_{b'} \oplus x_{b'})\Delta \oplus G_0^{g'} \oplus U_0^{g'} \oplus \sigma(X_{a'} \oplus k_0^{g'}) && (11) \end{aligned}$$

  In this case, $\mathcal{A}$ can check the consistency between the value of $G_u^g \oplus G_{u'}^{g'}$ and that of Δ at the end of experiment *without* further sending required queries, which are computed from Δ, to a random permutation or its inverse. In the real world, the consistency certainly holds. However, the ideal-world garbled rows and Δ are independently sampled, leading to the consistency only with negligible probability. So, $\mathcal{A}$ has non-negligible advantage to distinguish the two worlds and the statistical distance, as an upper bound, also blows up.
  More specifically, the real world is as follows in this case. The pre-image collision (5) leads to the syntactically same XOR of two hash masks in $G_u^g, G_{u'}^{g'}$, which can be XORed to cancel all hash masks to check the consistency with Δ without further queries. Moreover, the image collision (6) also implies the pre-image collision (5) since π is permutation. The other equalities imply the image collision $\pi(X_w \oplus \Delta \oplus k_u^g) = \pi(X_{w'} \oplus \Delta \oplus k_{u'}^{g'})$ for some distinct tuple $(g, u), (g', u') \in \mathcal{G}_{\mathsf{and}}(f) \times \{0, 1\}$, $w := \mathsf{in}_u(g)$, and $w' := \mathsf{in}_{u'}(g')$. Given permutation π, this collision implies the pre-image collision (5), which can be used to see the consistency. However, the above cancelling of hash masks will not give this consistency except with negligible probability in the ideal world.
- bad_2. There exists $((\alpha, \beta), g) \in \cup_{\ell=1}^{3} \mathcal{K}_\ell \times \mathcal{G}_{\mathsf{and}}(f)$ such that

$$\begin{aligned} & \alpha = X_a \oplus \Delta \oplus k_0^g && (12) \\ \vee \quad & \alpha = X_b \oplus \Delta \oplus k_1^g && (13) \\ \vee \quad & \beta = \sigma(\Delta) \oplus (s_b \oplus x_b)\Delta \\ & \qquad \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g) && (14) \\ \vee \quad & \beta = \sigma(\Delta) \oplus x_a \Delta \oplus X_a \\ & \qquad \oplus G_1^g \oplus U_1^g \oplus \sigma(X_b \oplus k_1^g) && (15) \end{aligned}$$

  where $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$.
  In this case, $\mathcal{A}$ can essentially guess Δ before receiving this value. It allows $\mathcal{A}$ to distinguish the real world, where every G_i^g is consistent with Δ, and the ideal world with a dummy Δ. So, the statistical distance blows up.
- bad_3. There exists $((\alpha, \beta), g) \in \cup_{\ell=1}^{2} \mathcal{K}_\ell \times \mathcal{G}_{\mathsf{and}}(f)$ such that

$$\begin{aligned} & \alpha = X_a \oplus k_0^g && (16) \\ \vee \quad & \alpha = X_b \oplus k_1^g && (17) \\ \vee \quad & \beta = U_0^g \oplus \sigma(X_a \oplus k_0^g) && (18) \\ \vee \quad & \beta = U_1^g \oplus \sigma(X_b \oplus k_1^g) && (19) \end{aligned}$$

  where $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$.
  In this case, $\mathcal{A}$ can make forward/backward queries w.r.t. some active labels before receiving active input labels and computing other active ones. We use this case to explicitly ensure that these query-response pairs are fixed by the queries to $\pi^{\pm 1}(\cdot)$ in the step 10 to 12 of $\mathsf{HG.Garble}^{\pi^{\pm 1}(\cdot)}$ (when τ is produced in the real world) or the queries to $\widetilde{\pi}^{\pm 1}(\cdot)$ in the step 9 of $\mathsf{SimF}^{\widetilde{\pi}^{\pm 1}(\cdot)}$ (when τ is produced in the ideal world), instead of the extra queries of $\mathcal{A}$.
  This case is used to simplify probability analysis.

Bounding $1 - \varepsilon_2$**.** Without loss of generality, we fix some good transcript τ such that $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ (if this probability is zero, it is trivial by definition that $\varepsilon_2 = 0$ for this τ). Using Lemma 1, we turn to analyze the sampled oracle's compatibility (Section 3.4) with such a transcript, instead of the interaction between $\mathcal{A}$ and the sampled oracle.

Note that there is a computationally unbounded non-uniform adversary $\mathcal{A}'$ such that, for every oracle ω, it sends the queries in τ in order in its interaction with ω (e.g., $\mathcal{A}'$ has auxiliary input τ and sends its ordered queries). Fix $\mathcal{A}'$ in the following compatibility analysis so that any real-world or ideal-world oracle will receive the queries in τ in order. For a response c recorded in a fixed τ, let $\omega \vdash c$ denote the event that, fixing the ordered queries as per τ, oracle ω produces c given the corresponding query. Let $\mathcal{K}^{\mathsf{R}}$ denote

[6] Strictly speaking, such bad_i's are the disjunctive predicates of $\tau \in \mathcal{T}_{\mathcal{A}}$ to define, in set-builder notation, a set $\mathcal{T}_{\mathsf{bad}} \subseteq \mathcal{T}_{\mathcal{A}}$ of bad transcripts. We treat them as "events" to avoid cumbersome notation. The formal events (i.e., the specific sets of oracles) w.r.t. bad_i's are well-defined from $\mathcal{T}_{\mathsf{bad}}$ and the two random variables in Section 3.4.

the order-preserving list of the responses in an ordered list $\mathcal{K}$ of permutation query-response pairs.

First, we compute $\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)]$. Following from half-gates, a real-world oracle $\omega = (\Delta, \{W_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, \pi) \in \Omega_{\mathsf{real}}$. It holds that

$$\begin{aligned} &\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] \\ &\quad = \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, \widehat{f}, \mathcal{K}_2^{\mathsf{R}}, \widetilde{X}, d, \widetilde{x}, \widetilde{U}, \mathcal{K}_3^{\mathsf{R}}, \Delta)\right]. \end{aligned}$$

To begin with, every real-world ω certainly produces f' (i.e., the first value in $\widehat{f}$) and $\widetilde{x}$ fixed in τ, which leads to $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$. This non-zero probability implies that $(f', \widetilde{x})$ in τ is honestly and deterministically computed from the fixed queries (f, x). Otherwise, no ideal-world oracle, which computes $(f', \widetilde{x})$ from the same deterministic procedure, can produce this transcript, contradicting the non-zero probability. As every real-world ω will follow the same deterministic procedure, it certainly produces the two values.

Meanwhile, a real-world oracle ω should have the same literal value of Δ as its counterpart in τ. Conditioned on the compatibility so far, the probability

$$\begin{aligned} &\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] \\ = &\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, F, \mathcal{K}_2^{\mathsf{R}}, \widetilde{X}, d, \widetilde{U}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash \Delta\right] \\ &\quad \cdot \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \vdash (f', \widetilde{x}) \wedge \omega \vdash \Delta] \\ = &\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, F, \mathcal{K}_2^{\mathsf{R}}, \widetilde{X}, d, \widetilde{U}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash \Delta\right] \\ &\quad \cdot \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \vdash (f', \widetilde{x})] \cdot \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \vdash \Delta] \\ = &\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, F, \mathcal{K}_2^{\mathsf{R}}, \widetilde{X}, d, \widetilde{U}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash \Delta\right] \\ &\quad \cdot \frac{1}{2^{\lambda - 1}}. \end{aligned}$$

Conditioned on the compatibility with $(f', \widetilde{x}, \Delta)$, a real-world ω should also be compatible with $(\cup_{\ell=1}^{3} \mathcal{K}_\ell^{\mathsf{R}}, F, \widetilde{U})$ and some active labels in $\widetilde{X}$ such that

(i) $\pi^{\pm 1}$ maps the fixed permutation queries to the responses in $\cup_{\ell=1}^{3} \mathcal{K}_\ell^{\mathsf{R}}$.

(ii) For each $i \in \mathcal{W}_{\mathsf{in}}(f)$, it holds that $X_i = W_i \oplus x_i \Delta$.

(iii) For each $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$, it holds that

$$\begin{aligned} \mathsf{H}(X_a, k_0^g) &:= \pi(X_a \oplus k_0^g) \oplus \sigma(X_a \oplus k_0^g) = U_0^g, \\ \mathsf{H}(X_b, k_1^g) &:= \pi(X_b \oplus k_1^g) \oplus \sigma(X_b \oplus k_1^g) = U_1^g. \end{aligned} \tag{20}$$

(iv) For each $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$, it holds that

$$\begin{aligned} X_c = {}& \mathsf{H}(X_a, k_0^g) \oplus \mathsf{H}(X_b, k_1^g) \\ & \oplus s_a G_0^g \oplus s_b (G_1^g \oplus X_a), \end{aligned} \tag{21}$$

$$\begin{aligned} G_0^g = {}& \mathsf{H}(X_a, k_0^g) \oplus \mathsf{H}(X_a \oplus \Delta, k_0^g) \\ & \oplus (s_b \oplus x_b)\Delta \\ G_1^g = {}& \mathsf{H}(X_b, k_1^g) \oplus \mathsf{H}(X_b \oplus \Delta, k_1^g) \\ & \oplus x_a \Delta \oplus X_a, \end{aligned} \tag{22}$$

where the bits $x_a, x_b, s_a = \mathsf{lsb}(X_a), s_b = \mathsf{lsb}(X_b)$ are given in τ.

Conditioned on the compatibility so far, every real-world oracle ω is always compatible with the leftover values in τ, i.e., decoding table d and other active labels in $\widetilde{X}$, which are deterministically computed from XOR combination. The reason is that, for τ ensuring $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$, these values should be honestly determined by the conditioned values as in the real world. Otherwise, this probability will be zero for an ideal-world oracle, which obtains them from a consistent deterministic computation as per the conditioned values. As every real-world oracle ω honestly follows the real-world computation, this "leftover" compatibility must hold conditioned on the previous compatibility.

It remains to compute the conditional probabilities for (i) to (iv). Consider (iii) and (iv). We note that every good τ with $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ already implies (20) and (21). Otherwise, no ideal-world oracle, which computes these values according to the step 9 and 10 in SimF (using the given tweakable hash function), can produce τ, contradicting the non-zero probability.

Then, consider (22), the leftover part of (iii) and (iv). We rewrite (22) as:

$$\mathcal{V} := \left\{ \begin{aligned} & g \in \mathcal{G}_{\mathsf{and}}(f), (a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g)) : \\ & \quad \underbrace{\pi(X_a \oplus k_0^g)}_{P_{g,0}} \oplus \underbrace{\pi(X_a \oplus \Delta \oplus k_0^g)}_{P_{g,2}} \\ & \qquad\qquad = \sigma(\Delta) \oplus (s_b \oplus x_b)\Delta \oplus G_0^g, \\ & \quad \underbrace{\pi(X_b \oplus k_1^g)}_{P_{g,1}} \oplus \underbrace{\pi(X_b \oplus \Delta \oplus k_1^g)}_{P_{g,3}} \\ & \qquad\qquad = \sigma(\Delta) \oplus x_a \Delta \oplus X_a \oplus G_1^g \end{aligned} \right\}$$

As τ is a good transcript, there are exactly $4|f|$ pairwise distinct permutation pre-images on the left hand (otherwise, there will be a pair of permutation pre-images leading to (5) in bad_1 or a permutation pre-image leading to (12)$\vee$(13) in bad_2 given the extra queries). $\mathcal{V}$ has exact $4|f|$ syntactically different variables $\mathcal{P} := \{P_{g,0}, P_{g,1}, P_{g,2}, P_{g,3}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$. They fix the same number of the entries of permutation π in a real-world ω *if and only if their literal values fixed by τ are also pairwise distinct*. Note that every good transcript τ indeed fixes *exact one* such assignment of these values for the following reasons:

- (20) already holds for τ, i.e., for $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$,

$$\begin{aligned} P_{g,0} &:= \pi(X_a \oplus k_0^g) = U_0^g \oplus \sigma(X_a \oplus k_0^g), \\ P_{g,1} &:= \pi(X_b \oplus k_1^g) = U_1^g \oplus \sigma(X_b \oplus k_1^g). \end{aligned} \tag{23}$$

  The literal values of $\{P_{g,0}, P_{g,1}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$ are fixed by the responses in $\mathcal{K}_3^{\mathsf{R}}$ given the extra queries and will be pairwise distinct due to the impossible (6) from $\neg\mathsf{bad}_1$.
- For $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$, $\{P_{g,2}, P_{g,3}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$ are literally assigned the following

values fixed by τ according to $\mathcal{V}$ and (23):

$$\begin{aligned} & P_{g,2} := \pi(X_a \oplus \Delta \oplus k_0^g) \\ = \; & \sigma(\Delta) \oplus (s_b \oplus x_b)\Delta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g), \\ & P_{g,3} := \pi(X_b \oplus \Delta \oplus k_1^g) \\ = \; & \sigma(\Delta) \oplus x_a\Delta \oplus X_a \oplus G_1^g \oplus U_1^g \oplus \sigma(X_b \oplus k_1^g). \end{aligned} \tag{24}$$

Clearly, one can see that the literal values of $\{P_{g,2}, P_{g,3}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$ are pairwise distinct according to the impossible (7) $\vee$ (8) $\vee$ (9) $\vee$ (10) $\vee$ (11) from $\neg\mathsf{bad}_1$.

- The goodness of τ also ensures that there do not exist

$$P' \in \{P_{g,0}, P_{g,1}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}, P'_{\Delta} \in \{P_{g,2}, P_{g,3}\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}$$

such that $P' = P'_{\Delta}$. Otherwise, this equality and (24) ensure that there exist $(g, u) \in \mathcal{G}_{\mathsf{and}}(f) \times \{0, 1\}$ and $g' \in \mathcal{G}_{\mathsf{and}}(f)$ such that

$$\begin{aligned} & \pi(X_w \oplus k_u^g) = \pi(X_{a'} \oplus \Delta \oplus k_0^{g'}) \\ = \; & \sigma(\Delta) \oplus (s_{b'} \oplus x_{b'})\Delta \oplus G_0^{g'} \oplus U_0^{g'} \oplus \sigma(X_{a'} \oplus k_0^{g'}) \\ \vee \quad & \pi(X_w \oplus k_u^g) = \pi(X_{b'} \oplus \Delta \oplus k_1^{g'}) \\ = \; & \sigma(\Delta) \oplus x_{a'}\Delta \oplus X_{a'} \oplus G_1^{g'} \oplus U_1^{g'} \oplus \sigma(X_{b'} \oplus k_1^{g'}) \end{aligned} \tag{25}$$

where $w := \mathsf{in}_u(g)$ and $(a', b') := (\mathsf{in}_0(g'), \mathsf{in}_1(g'))$. Recall that $\neg\mathsf{bad}_3$ and the extra queries implies $(X_w \oplus k_u^g, \pi(X_w \oplus k_u^g)) \in \mathcal{K}_3$. As a result, (25) leads to contradiction with the impossible (14) $\vee$ (15) from $\neg\mathsf{bad}_2$.

Putting these cases together, we can see that τ yields a value assignment of $\mathcal{P}$, and this assignment fixes exact $4\,|f|$ entries of real-world permutation π.

Conditioned on $\neg\mathsf{bad}_1 \wedge \neg\mathsf{bad}_3$ of good transcript τ, $2\,|f|$ extra queries are non-repeating and the number of non-repeating queries is $q + 2\,|f| = q_\Sigma$. So, $2\,|f|$ responses in $\cup_{\ell=1}^{3} \mathcal{K}_\ell^{\mathsf{R}}$ for the non-repeating extra queries are fixed by the values in $\mathcal{P}$ while the other $q_\Sigma - 2\,|f| = q$ responses are fixed by real-world π (conditioned on the values in $\mathcal{P}$). Using (i), (ii), (iii), and (iv) together with the "leftover" compatibility, we have in the real world that

$$\begin{aligned} & \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}} \left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, F, \mathcal{K}_2^{\mathsf{R}}, \widetilde{X}, d, \widetilde{U}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash \Delta\right] \\ & = \frac{1}{(2^\lambda)^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{(2^\lambda - 4\,|f| - (q_\Sigma - 2\,|f|))!}{(2^\lambda)!} \\ & = \frac{1}{(2^\lambda)^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{1}{(2^\lambda)_{q+4|f|}}, \\ & \Pr_{\omega \leftarrow \Omega_{\mathsf{real}}} [\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] = \frac{1}{(2^\lambda)^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{1}{(2^\lambda)_{q+4|f|}} \cdot \frac{1}{2^{\lambda-1}}. \end{aligned}$$

Second, in the ideal world, condition $\neg\mathsf{bad}_3$ ensures that the responses for the $2\,|f|$ extra queries are fixed by the queries in the step 9 of $\mathsf{SimF}^{\widetilde{\pi}^{\pm 1}(\cdot)}$. So, each U_u^g is independently uniform according to the randomness of $\widetilde{\pi}^{\pm 1}$ and the pairwise distinct pre-images by $\neg\mathsf{bad}_1$. From the garbled rows, the active input labels, and these U_u^g, the active internal and output labels (as well as decoding table d) are fixed in topology order. So, we have

$$\begin{aligned} & \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} [\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)] \\ = & \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} \left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, \mathcal{K}_2^{\mathsf{R}}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (\widehat{f}, \widetilde{X}, d, \widetilde{x}, \widetilde{U}, \Delta)\right] \\ & \cdot \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} [\omega \vdash (\widehat{f}, \widetilde{X}, d, \widetilde{x}, \widetilde{U}, \Delta)] \\ = & \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} \left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, \mathcal{K}_2^{\mathsf{R}}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (\widehat{f}, \widetilde{X}, d, \widetilde{x}, \widetilde{U}, \Delta)\right] \\ & \cdot \frac{1}{2^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda + 2\lambda|f| + (\lambda-1) + 2\lambda|f|}}. \end{aligned}$$

According to the condition $\neg\mathsf{bad}_1 \wedge \neg\mathsf{bad}_3$ and the $2\,|f|$ extra queries, there are exact $q + 2\,|f| = q_\Sigma$ non-repeating queries. Moreover, $2\,|f|$ responses in $\cup_{\ell=1}^{3} \mathcal{K}_\ell^{\mathsf{R}}$ for the non-repeating extra queries are fixed by the conditioned values but the other responses are fixed by $\widetilde{\pi}^{\pm 1}$ for other $q_\Sigma - 2\,|f| = q$ non-repeating queries (which are responded with the rejection sampling, see Figure 5).

Let $\mathcal{N} \subseteq [1, q_\Sigma]$ denote the index set of these q queries in q_Σ non-repeating queries to $\widetilde{\pi}^{\pm 1}(\cdot)$ such that $|\mathcal{N}| = q$. The rejection sampling in $\widetilde{\pi}^{\pm 1}(\cdot)$ gives

$$\begin{aligned} & \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} \left[\omega \vdash (\mathcal{K}_1^{\mathsf{R}}, \mathcal{K}_2^{\mathsf{R}}, \mathcal{K}_3^{\mathsf{R}}) \;\middle|\; \omega \vdash (\widehat{f}, \widetilde{X}, d, \widetilde{x}, \widetilde{U}, \Delta)\right] \\ & = \prod_{i \in \mathcal{N}} \frac{1}{2^\lambda - |\mathcal{Q}_{i-1}|} = \prod_{i \in \mathcal{N}} \frac{1}{2^\lambda - (i-1)} \\ & \leq \frac{1}{2^\lambda - 2\,|f|} \times \cdots \times \frac{1}{2^\lambda - (q_\Sigma - 1)} = \frac{1}{(2^\lambda - 2\,|f|)_q}, \\ & \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} [\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)] \\ & \leq \frac{1}{(2^\lambda)^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{1}{(2^\lambda - 2\,|f|)_q \cdot (2^\lambda)^{4|f|}} \cdot \frac{1}{2^{\lambda-1}}. \end{aligned}$$

So, we can have $\varepsilon_2 = 0$ since, for every $|f| \geq 0$ and every $q \geq 0$,

$$\begin{aligned} \frac{\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}} [\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)]}{\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} [\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)]} & \geq \frac{(2^\lambda - 2\,|f|)_q \cdot (2^\lambda)^{4|f|}}{(2^\lambda)_{q+4|f|}} \\ & \geq \frac{(2^\lambda - 2\,|f|)_q \cdot (2^\lambda)_{2|f|} \cdot (2^\lambda)^{2|f|}}{(2^\lambda)_{q+4|f|}} \\ & = \frac{(2^\lambda)_{q+2|f|} \cdot (2^\lambda)^{2|f|}}{(2^\lambda)_{q+4|f|}} = \frac{(2^\lambda)^{2|f|}}{(2^\lambda - (q + 2\,|f|))_{2|f|}} \geq 1. \end{aligned}$$

Bounding ε_1. First, consider $\mathsf{bad}_1 \vee \mathsf{bad}_3$. For each $i \in [2, 2\,|f|]$, let coll_i denote the event that there exist distinct $(g, u), (g', u') \in$ "the first i pairs of $\mathcal{G}_{\mathsf{and}}(f) \times \{0, 1\}$" such that $X_w \oplus k_u^g = X_{w'} \oplus k_{u'}^{g'}$, where $w := \mathsf{in}_u(g)$ and $w' := \mathsf{in}_{u'}(g')$, query_i denote the event that there exists $((\alpha, \beta), (g, u)) \in \cup_{\ell=1}^{2} \mathcal{K}_\ell \times$ "the first i pairs of $\mathcal{G}_{\mathsf{and}}(f) \times \{0, 1\}$" such that $\alpha = X_w \oplus k_u^g$, where $w := \mathsf{in}_u(g)$, and bFwd_i denote $\mathsf{coll}_i \vee \mathsf{query}_i$. Then, $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bFwd}_{2|f|}] = \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[(5) \vee (16) \vee (17)]$. We will prove the following bound using an induction:

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}} [\mathsf{bFwd}_i] \leq \frac{i(i-1) + 2i \cdot (q_1 + q_2)}{2^{\lambda+1}}.$$

In the base case ($i = 2$), X_w and $X_{w'}$ are two active circuit input labels so that coll_2 occurs with probability at most $1/2^\lambda$ due to the sampling in the step 2 of SimF (this probability is zero if $w = w'$). For query_2, each of the two labels matches a fixed $\alpha \oplus k_u^g$ with probability $1/2^\lambda$. Following from a union bound, the target bound holds here.

Assume that the probability bound holds for $i \in [2, 2|f| - 1]$ and consider the $i + 1$ case. Using the law of total probability, we have

$$\begin{aligned}
&\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1}\right] \\
&= \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1} \;\middle|\; \mathsf{bFwd}_i\right]\cdot \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_i\right] \\
&\quad + \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1} \;\middle|\; \neg\mathsf{bFwd}_i\right]\cdot \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\neg\mathsf{bFwd}_i\right] \\
&\le \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_i\right] + \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1} \;\middle|\; \neg\mathsf{bFwd}_i\right] \\
&\le \frac{i(i-1) + 2i\cdot(q_1+q_2)}{2^{\lambda+1}} + \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1} \;\middle|\; \neg\mathsf{bFwd}_i\right]
\end{aligned}$$

Let (g^*, u^*) denote the $(i+1)$-th pair of $\mathcal{G}_{\mathsf{and}}(f)\times\{0,1\}$ and $w^* := \mathsf{in}_{u^*}(g^*)$. To incur coll_{i+1} conditioned on $\neg\mathsf{bFwd}_i = \neg\mathsf{coll}_i \wedge \neg\mathsf{query}_i$, we have $X_{w^*} \oplus k_{u^*}^{g^*} = X_w \oplus k_u^g$, where $(g, u) \in$ “the first i pairs of $\mathcal{G}_{\mathsf{and}}(f) \times \{0,1\}$” and $w := \mathsf{in}_u(g)$. Recall that each active label X_i is the XOR from (i) some active circuit input labels, and/or (ii) some active output labels of the *precedent* `AND` gates, i.e.,

$$\begin{aligned}
X_i &= \Big(\bigoplus\nolimits_{w\in\mathcal{I}_i\subseteq\mathcal{W}_{\mathsf{in}}(f)} X_w\Big) \oplus \Big(\bigoplus\nolimits_{w\in\mathcal{J}_i\subseteq\mathcal{W}_{\mathsf{and}}(f)} X_w\Big) \\
&= \bigoplus\nolimits_{w\in\mathcal{I}_i\ominus\mathcal{J}_i} X_w \in \{0,1\}^\lambda.
\end{aligned} \tag{26}$$

for $\mathcal{I}_i \ominus \mathcal{J}_i \neq \emptyset$. We use (26) to rewrite the equality to incur coll_{i+1} as follows:

$$\bigoplus\nolimits_{i\in(\mathcal{I}_w\ominus\mathcal{I}_{w^*})\ominus(\mathcal{J}_w\ominus\mathcal{J}_{w^*})} X_i = k_u^g \oplus k_{u^*}^{g^*} \in \{0,1\}^\lambda.$$

Here, the active circuit input labels in $\mathcal{I}_w \ominus \mathcal{I}_{w^*}$ are sampled at random in Sim. For the active labels in $\mathcal{J}_w \ominus \mathcal{J}_{w^*}$, they are masked by the responses sent from $\widetilde{\pi}^{\pm1}(\cdot)$ to Sim. Conditioned on $\neg\mathsf{bFwd}_i$, these responses are taken from uniform $c_1, \ldots, c_{n(\lambda)}$ in $\widetilde{\pi}^{\pm1}(\cdot)$ and pairwise independent since the queries (i.e., $X_w \oplus k_u^g$ for $(g,u) \in$ “the first i pairs of $\mathcal{G}_{\mathsf{and}}(f) \times \{0,1\}$” and $w := \mathsf{in}_u(g)$) are pairwise distinct under this condition. The active labels in $\mathcal{J}_w \ominus \mathcal{J}_{w^*}$ are uniform, and the XOR on the left hand is uniform unless $\mathcal{I}_w = \mathcal{I}_{w^*}$ and $\mathcal{J}_w = \mathcal{J}_{w^*}$. That is, the equality to incur coll_{i+1} holds with probability at most $1/2^\lambda$ for some fixed (g, u). The same argument and probability hold for the equality $\alpha = X_{w^*} \oplus k_{u^*}^{g^*}$ to incur query_{i+1} for some fixed (α, β) under $\neg\mathsf{bFwd}_i$. Using a union bound,

$$\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{i+1} \;\middle|\; \neg\mathsf{bFwd}_i\right] \le \frac{i + (q_1+q_2)}{2^\lambda},$$

which concludes the induction for the $i + 1$ case. We have

$$\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{2|f|}\right] \le \frac{2|f|(2|f|-1) + 4|f|\cdot(q_1+q_2)}{2^{\lambda+1}}. \tag{27}$$

Consider (6), (7), (8), (9), (10), (11), (18), and (19) conditioned on $\mathsf{bFwd}_{2|f|}$. In each of them, $U_u^g \oplus \sigma(X_w \oplus k_u^g)$ is the response for query $X_w \oplus k_u^g$ to $\widetilde{\pi}^{\pm1}(\cdot)$. Conditioned on $\neg\mathsf{bFwd}_i$, these responses are taken from uniform $c_1, \ldots, c_{n(\lambda)}$ in $\widetilde{\pi}^{\pm1}(\cdot)$ and pairwise independent as the queries are pairwise distinct under this condition. Therefore, each of them occurs with probability $1/2^\lambda$ for some fixed quantifier. Let $\mathsf{bBwd} := (6) \vee (7) \vee (8) \vee (9) \vee (10) \vee (11) \vee (18) \vee (19)$. From a union bound over all quantifiers,

$$\begin{aligned}
&\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bBwd} \;\middle|\; \neg\mathsf{bFwd}_{2|f|}\right] \\
&\quad\le \frac{2|f|(2|f|-1) + 2|f|\cdot(q_1+q_2)}{2^\lambda}.
\end{aligned} \tag{28}$$

Using (27) and (28), we have

$$\begin{aligned}
\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bad}_1 \vee \mathsf{bad}_3\right] &= \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{2|f|} \vee \mathsf{bBwd}\right] \\
&= \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bFwd}_{2|f|}\right] + \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bBwd} \;\middle|\; \neg\mathsf{bFwd}_{2|f|}\right] \\
&\le \frac{3|f|(2|f|-1) + 4|f|\cdot(q_1+q_2)}{2^\lambda}.
\end{aligned} \tag{29}$$

Then, consider bad_2. From (12) (resp., (13)), we see $\Delta = \alpha \oplus X_a \oplus k_0^g$ (resp., $\Delta = \alpha \oplus X_b \oplus k_1^g$), occurring with probability $2^{-(\lambda-1)}$ due to the randomness of Δ. In (14), if $s_b \oplus x_b = 0$, linear orthomorphism σ ensures that

$$\Delta = \sigma^{-1}(\beta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g)),$$

which occurs with probability $2^{-(\lambda-1)}$; if $s_b \oplus x_b = 1$, according to permutation $\sigma'(x) := \sigma(x) \oplus x$ well-defined from σ, it holds with the same probability that

$$\Delta = \sigma'^{-1}(\beta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g)).$$

Similar result holds for (15). Taking a union bound over all pairs, we have

$$\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bad}_2\right] \le \frac{4|f|\cdot(q_1+q_2+q_3)}{2^{\lambda-1}}. \tag{30}$$

We have a bound ε_1 from (29) and (30):

$$\begin{aligned}
\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[Y(\omega) \in \mathcal{T}_{\mathsf{bad}}\right] &= \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3\right] \\
&\le \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bad}_1 \vee \mathsf{bad}_3\right] + \Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\mathsf{bad}_2\right] \\
&\le \frac{3|f|(2|f|-1) + 12|f|\cdot(q + 2|f|)}{2^\lambda} \\
&= \frac{12qs + 30s^2 - 3s}{2^\lambda} = \varepsilon_1.
\end{aligned}$$

This lemma follows from the H-coefficient technique with the above $\varepsilon_1, \varepsilon_2$. □

References

[1] A. C.-C. Yao, “How to generate and exchange secrets (extended abstract),” in *IEEE FOCS*, 1986.

[2] M. Bellare, V. T. Hoang, and P. Rogaway, “Foundations of garbled circuits,” in *ACM CCS*, 2012.

[3] M. Jawurek, F. Kerschbaum, and C. Orlandi, “Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently,” in *ACM CCS*, 2013.

[4] N. Döttling and S. Garg, “Identity-based encryption from the Diffie-Hellman assumption,” in *IACR Crypto*, 2017.

[5] Y. Ishai and E. Kushilevitz, “Randomizing polynomials: A new representation with applications to round-efficient secure computation,” in *IEEE FOCS*, 2000.

[6] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters, “Encoding functions with constant online rate or how to compress garbled circuits keys,” in *IACR Crypto*, 2013.

[7] Y. Lindell and B. Riva, “Cut-and-choose Yao-based secure computation in the online/offline and batch settings,” in *IACR Crypto*, 2014.

[8] ——, “Blazing fast 2PC in the offline/online setting with security for malicious adversaries,” in *ACM CCS*, 2015.

[9] P. Rindal and M. Rosulek, “Faster malicious 2-party secure computation with online/offline dual execution,” in *USENIX Security*, 2016.

[10] P. Mohassel and M. Rosulek, “Non-interactive secure 2PC in the offline/online and batch settings,” in *IACR Eurocrypt*, 2017.

[11] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, “One-time programs,” in *IACR Crypto*, 2008.

[12] A. Sahai and H. Seyalioglu, “Worry-free encryption: functional encryption with public keys,” in *ACM CCS*, 2010.

[13] S. Gorbunov, V. Vaikuntanathan, and H. Wee, “Functional encryption with bounded collusions via multi-party computation,” in *IACR Crypto*, 2012.

[14] R. Gennaro, C. Gentry, and B. Parno, “Non-interactive verifiable computing: Outsourcing computation to untrusted workers,” in *IACR Crypto*, 2010.

[15] B. Applebaum, Y. Ishai, and E. Kushilevitz, “From secrecy to soundness: Efficient verification via secure computation,” in *ICALP*, 2010.

[16] D. Beaver, S. Micali, and P. Rogaway, “The round complexity of secure protocols (extended abstract),” in *ACM STOC*, 1990.

[17] M. Naor, B. Pinkas, and R. Sumner, “Privacy preserving auctions and mechanism design,” in *ACM EC*, 1999.

[18] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications,” in *ICALP*, 2008.

[19] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, “Secure two-party computation is practical,” in *IACR Asiacrypt*, 2009.

[20] V. Kolesnikov, P. Mohassel, and M. Rosulek, “FleXOR: Flexible garbling for XOR gates that beats free-XOR,” in *IACR Crypto*, 2014.

[21] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas, “Fast garbling of circuits under standard assumptions,” in *ACM CCS*, 2015.

[22] S. Zahur, M. Rosulek, and D. Evans, “Two halves make a whole - reducing data transfer in garbled circuits using half gates,” in *IACR Eurocrypt*, 2015.

[23] M. Rosulek and L. Roy, “Three halves make a whole? Beating the half-gates lower bound for garbled circuits,” in *IACR Crypto*, 2021.

[24] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, “Efficient garbling from a fixed-key blockcipher,” in *IEEE S&P*, 2013.

[25] C. Guo, J. Katz, X. Wang, and Y. Yu, “Efficient and secure multiparty computation from fixed-key block ciphers,” in *IEEE S&P*, 2020.

[26] M. Bellare, V. T. Hoang, and P. Rogaway, “Adaptively secure garbling with applications to one-time programs and secure outsourcing,” in *IACR Asiacrypt*, 2012.

[27] B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro, and D. Wichs, “Adaptively secure garbled circuits from one-way functions,” in *IACR Crypto*, 2016.

[28] Z. Jafargholi and D. Wichs, “Adaptive security of Yao’s garbled circuits,” in *IACR TCC*, 2016.

[29] Z. Jafargholi, C. Kamath, K. Klein, I. Komargodski, K. Pietrzak, and D. Wichs, “Be adaptive, avoid overcommitting,” in *IACR Crypto*, 2017.

[30] S. Garg and A. Srinivasan, “Adaptively secure garbling with near optimal online complexity,” in *IACR Eurocrypt*, 2018.

[31] Z. Jafargholi and S. Oechsner, “Adaptive security of practical garbling schemes,” in *Indocrypt*, 2020.

[32] P. Rogaway and J. P. Steinberger, “Constructing cryptographic hash functions from fixed-key blockciphers,” in *IACR Crypto*, 2008.

[33] P. Hubacek and D. Wichs, “On the communication complexity of secure function evaluation with long output,” in *ITCS*, 2015.

[34] Y. Lindell and B. Pinkas, “A proof of security of Yao’s protocol for two-party computation,” *J. Cryptology*, vol. 22, no. 2, Apr. 2009.

[35] R. Canetti, “Security and composition of multiparty cryptographic protocols,” *J. Cryptology*, vol. 13, no. 1, Jan. 2000.

[36] J. Patarin, “The “coefficients H” technique (invited talk),” in *Annual International Workshop on Selected Areas in Cryptography (SAC) 2008*, vol. 5381. Springer, 2009.

[37] S. Chen and J. P. Steinberger, “Tight security bounds for key-alternating ciphers,” in *IACR Eurocrypt*, 2014.

[38] Y. Dai, J. Lee, B. Mennink, and J. P. Steinberger, “The security of multiple encryption in the ideal cipher model,” in *IACR Crypto*, 2014.

[39] V. T. Hoang and S. Tessaro, “Key-alternating ciphers and key-length extension: Exact bounds and multi-user security,” in *IACR Crypto*, 2016.

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

[41] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, “ObliVM: A programming framework for secure computation,” in *IEEE S&P*, 2015.

[42] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, “TinyGarble: Highly compressed and scalable sequential garbled circuits,” in *IEEE S&P*, 2015.

[43] D. Demmler, T. Schneider, and M. Zohner, “ABY - A framework for efficient mixed-protocol secure two-party computation,” in *NDSS*, 2015.

[44] M. Keller, “MP-SPDZ: A versatile framework for multi-party computation,” in *ACM CCS*, 2020.

[45] R. Nieminen and T. Schneider, “Breaking and fixing garbled circuits when a gate has duplicate input wires,” *J. Cryptol.*, 2023.

[46] J. B. Nielsen, “Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case,” in *IACR Crypto*, 2002.

[47] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *ACM CCS*, 1993.

[48] A. C.-C. Yao, “Theory and applications of trapdoor functions (extended abstract),” in *IEEE FOCS*, 1982.

[49] B. Barak, R. Shaltiel, and A. Wigderson, “Computational analogues of entropy,” in *APPROX-RANDOM*, 2003.

[50] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin, “Conditional computational entropy, or toward separating pseudoentropy from compressibility,” in *IACR Eurocrypt*, 2007.

Appendix A. Adaptive Security of Half-Gates in pRPM

We argue that the original implementation of the half-gates scheme [22], which is identical to Figure 2 but sends decoding table d as part of garbled circuit $\widehat{f}$, is adaptively secure in the pRPM and does not suffer from the lower bound in the npRPM (see Appendix C).

For a circuit f with fan-in two and fan-out one, we in addition use the following notations in our simulator:

- $\mathcal{Y}_1(f), \ldots, \mathcal{Y}_m(f) \subseteq \mathcal{W}_{\mathsf{out}}(f)$: $m \geq 1$ *maximal* non-empty partitions of circuit output wires such that, for every

```
SimF(f):
1: $F \leftarrow (\{0,1\}^{2\lambda})^{|f|}$, $d \leftarrow \mathbb{F}_2^{|\mathcal{W}_{\mathsf{out}}(f)|}$ such that
   (i) For each $i \in [1,m]$ and each wire $j \in \mathcal{Y}_i(f)$, $d_j = d_{y_i(f)}$.
   (ii) For each $(\mu_1, \dots, \mu_m) \in \mathbb{F}_2^m$ such that $\ominus_{i\in[1,m],\mu_i=1}\mathcal{X}_i(f) = \varnothing$, $\oplus_{i\in[1,m]}\mu_i \cdot d_{y_i(f)} = 0$.
2: return $\widehat{f} := (f' := f, F, d)$, $\mathsf{st}_{\mathsf{sim}} := (\mathsf{st}_{\mathsf{sim}}, f, F, d)$

SimIn(f(x)):
1: $\{X_i\}_{i\in\mathcal{X}(f)} \leftarrow (\{0,1\}^\lambda)^{|\mathcal{X}(f)|}$ such that, for each $j \in [1,m]$, $\mathsf{lsb}(\oplus_{i\in\mathcal{X}_j(f)}X_i) = d_{y_j(f)} \oplus f(x)_{y_j(f)}$.
2: for $g \in \mathcal{G}(f)$ in order do
3:     $(a,b,c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
4:     if $\mathsf{type}(g) = \mathtt{XOR}$ then $X_c := X_a \oplus X_b$
5:     else if $\mathsf{type}(g) = \mathtt{AND}$ then
6:         $k_0^g := 2 \cdot g - 1$, $k_1^g := 2 \cdot g$
7:         $s_a := \mathsf{lsb}(X_a)$, $s_b := \mathsf{lsb}(X_b)$
8:         if $c \in \mathcal{Z}(f)$ then
9:             $U_1^g \leftarrow \{0,1\}^\lambda$
10:            $U_0^g := X_c \oplus U_1^g \oplus s_aG_0^g \oplus s_b(G_1^g \oplus X_a)$
11:        else if $c \notin \mathcal{Z}(f)$ then
12:            $U_0^g, U_1^g \leftarrow \{0,1\}^\lambda$
13:            $X_c := U_0^g \oplus U_1^g \oplus s_aG_0^g \oplus s_b(G_1^g \oplus X_a)$
14:        (Programming) Add two pairs
               $(X_a \oplus k_0^g, U_0^g \oplus \sigma(X_a \oplus k_0^g))$,
               $(X_b \oplus k_1^g, U_1^g \oplus \sigma(X_b \oplus k_1^g))$
           to the list $\mathcal{Q}$ kept in $\mathsf{st}_{\mathsf{sim}}$, if they cause no pre-image collision or image collision in $\mathcal{Q}$, to ensure
               $\mathsf{H}(X_a, k_0^g) = \pi(X_a \oplus k_0^g) \oplus \sigma(X_a \oplus k_0^g) = U_0^g$,
               $\mathsf{H}(X_b, k_1^g) = \pi(X_b \oplus k_1^g) \oplus \sigma(X_b \oplus k_1^g) = U_1^g$.
15: return $\widehat{x} := \{X_i\}_{i\in\mathcal{W}_{\mathsf{in}}(f)}$, $\mathsf{st}_{\mathsf{sim}} := (\mathsf{st}_{\mathsf{sim}}, \widetilde{X}, \widetilde{U})$ where $\mathsf{st}_{\mathsf{sim}}$ on the right hand has an updated list $\mathcal{Q}$, $\widetilde{X} := \{X_i\}_{i\in\mathcal{W}(f)}$, $\widetilde{U} := \{U_0^g, U_1^g\}_{g\in\mathcal{G}_{\mathsf{and}}(f)}$.
```

Figure 6: Our simulator for half-gates in the pRPM.

$i \in [1,m]$ and every distinct $w, w' \in \mathcal{Y}_i(f)$, wire w and w' are from the same XOR combination of (i) some circuit input wires, and/or (ii) some output wires of *last-level* AND gates. We point out that $\sum_{i=1}^m |\mathcal{Y}_i(f)| = |\mathcal{W}_{\mathsf{out}}(f)| \geq m$.

- $\mathcal{Z}(f) \subseteq \mathcal{W}_{\mathsf{and}}(f)$: The set of the output wires of *last-level* AND gates.
- $\mathcal{X}(f) := \mathcal{W}_{\mathsf{in}}(f) \cup \mathcal{Z}(f)$.
- For each $i \in [1,m]$, let $\mathcal{X}_i(f) \subseteq \mathcal{X}(f)$ denote the set collecting the wires used to XOR-combine the circuit output wires in $\mathcal{Y}_i(f)$, and $y_i(f)$ denote the first (i.e., representative) wire in $\mathcal{Y}_i(f)$.

Theorem 2. *Let $\mathsf{H}(X,k) = \pi(X \oplus k) \oplus \sigma(X \oplus k)$ be a tweakable hash function where $X, k \in \{0,1\}^\lambda$, $\pi \in \mathcal{S}_\lambda$ is random permutation, and $\sigma : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a linear orthomorphism. Then, half-gates ([22]) is a λ-garbling scheme with (q,s,ε)-adaptive security in the pRPM, where $\varepsilon = (8qs + 21s^2)/2^{\lambda-1}$.*

Proof (Sketch). The correctness also follows from the original work. As for simulation, our simulator Sim consists of $(\mathsf{SimF}, \mathsf{SimIn})$ in Figure 6 and $\mathsf{SimP}^{\pm 1}$, which emulates the random permutation and its inverse on-the-fly. More specifically, there is a list $\mathcal{Q}$ of query-response pairs in internal state $\mathsf{st}_{\mathsf{sim}}$. Upon receiving forward query α (resp., backward query β) from $\mathcal{A}_i$ to SimP (resp., SimP^{-1}), it reads $\mathcal{Q}$ from $\mathsf{st}_{\mathsf{sim}}$ and checks whether $\exists(\alpha,\gamma) \in \mathcal{Q}$ (resp., $\exists(\gamma,\beta) \in \mathcal{Q}$). If true, it returns γ as response; otherwise it samples $\gamma \leftarrow \{s \in \{0,1\}^\lambda \mid (\dots, s) \notin \mathcal{Q}\}$ (resp., $\gamma \leftarrow \{s \in \{0,1\}^\lambda \mid (s, \dots) \notin \mathcal{Q}\}$), adds (α,γ) (resp., (γ,β)) to $\mathcal{Q}$, and returns γ as response. The programming is the step 14 of SimIn, where (α,β) is added to $\mathcal{Q}$ if and only if there is no pre-image collision with $(\alpha, \dots) \in \mathcal{Q}$ or image collision with $(\dots, \beta) \in \mathcal{Q}$.

To see that Sim is PPT, we note that the circuit-dependent notation can be efficiently computed by traversing the polynomial-size circuit f. Then, the crux is to show that the step 1 in both SimF and SimIn can be polynomial-time.

The runtime of the step 1 of SimF is dominated by the runtime of iterating through all qualified $(\mu_1, \dots, \mu_m) \in \mathbb{F}_2^m$. To find the qualified vectors, one can interpret each $\mathcal{X}_i(f)$ as a one-hot *non-zero* column vector in the space $\mathbb{F}_2^{|\mathcal{W}(f)|}$ and derive a $|\mathcal{W}(f)|$-by-m matrix $\boldsymbol{E}$ from these column vectors. One can check that all qualified vectors fall in the kernel of $\boldsymbol{E}$. This kernel can be efficiently computed from the Gaussian elimination on $\boldsymbol{E}$ and is a subspace spanned by $m - \mathsf{rank}(\boldsymbol{E})$ basis vectors. So, the step 1 of SimF only needs to iterate through these basis vectors, and the other qualified vectors must satisfy the constraints as they are in the subspace. As a result, the step 1 of SimF runs in polynomial time due to the Gaussian elimination plus a linear-time pass to assign random or constrained values to $d_{y_i(f)}$'s according to the $m - \mathsf{rank}(\boldsymbol{E})$ basis vectors.

The step 1 of SimIn only requires one linear-time pass to assign constrained or random values to the active labels so it runs in polynomial time. The linear constraint on the LSBs of these active labels has rank $\mathsf{rank}(\boldsymbol{E})$ and is satisfiable for the $d_{y_i(f)}$'s assigned in SimF.

Then, we fix z and $\mathcal{A}$. We directly use the H-coefficient technique (in Section 3.4) with transcript padding to bound the advantage of distinguishing between the real world (i.e., the adaptive experiment that uses the half-gates scheme) and the ideal world (i.e., the adaptive experiment that uses simulator Sim). In this technique, we consider the computationally unbounded non-uniform adversary $\mathcal{A} = \mathcal{A}(z)$ and compute $\varepsilon_1, \varepsilon_2$.

Transcript padding. Our transcript padding here is almost identical to that in the proof of Lemma 3, except that

1) The forward and backward queries to approximate oracle $\widetilde{\pi}^{\pm 1}(\cdot)$ is now replaced by those to $\mathsf{SimP}^{\pm 1}(\cdot)$ in the simulation.
2) Decoding table d is now included in garbled circuit $\widehat{f}$ in the offline phase.
3) The literal values of $\widetilde{U} := \{U_0^g, U_1^g\}_{g\in\mathcal{G}_{\mathsf{and}}(f)}$ are now fixed by a part of the random tape of the oracle for the programming in SimIn.

As a result, real-world sample space

$$\Omega_{\mathsf{real}} = \{0,1\}^{\lambda-1} \times \{0,1\}^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda} \times \mathcal{S}_\lambda,$$

and ideal-world sample space

$$\begin{aligned}\mathit{\Omega}_{\mathsf{ideal}} = {} & (\{0,1\}^{2\lambda})^{|f|} \times \{0,1\}^{\mathsf{rank}(\boldsymbol{E})} \\ & \times \{0,1\}^{(|\mathcal{W}_{\mathsf{in}}(f)|+|\mathcal{Z}(f)|)\lambda-\mathsf{rank}(\boldsymbol{E})} \\ & \times (\{0,1\}^{\lambda})^{|\mathcal{Z}(f)|} \times (\{0,1\}^{2\lambda})^{|f|-|\mathcal{Z}(f)|} \\ & \times \underbrace{\{0,1\}^{*}}_{\substack{\text{random tape for} \\ \text{the sampling in } \mathsf{SimP}^{\pm 1}(\cdot)}} \times \underbrace{\{0,1\}^{\lambda-1}}_{\text{dummy } \mathit{\Delta}} .\end{aligned}$$

Bad transcripts. Our definition of bad transcripts here is identical to that in the proof of Lemma 3.

A notable point is that bad_3 now meaningfully captures the case where $\mathcal{A}$ can make forward/backward queries w.r.t. some active labels before receiving active input labels and computing other active ones. This case is necessary to ensure successful programming in the ideal world as the values on the right hand should not be queried before the programming (otherwise it fails due to pre-image or image collision). If the programming fails in the ideal world, the two worlds can be distinguishable as the decoding consistency in the ideal world does not always hold as in the real world.

Bounding $1-\varepsilon_2$. The procedure to derive this bound is almost identical to that in the proof of Lemma 3, and we simply claim that $\varepsilon_2 = 0$.

Bounding ε_1. We bound the probabilities of the bad events in the *ideal world*. First, consider bad_1. Note that each active label X_i can be written as the XOR of (i) some active circuit input labels, and/or (ii) some active output labels of the *precedent* `AND` gates, i.e., for $\mathcal{I}_i \ominus \mathcal{J}_i \neq \varnothing$,

$$\begin{aligned} X_i &= \left(\bigoplus\nolimits_{w\in\mathcal{I}_i\subseteq\mathcal{W}_{\mathsf{in}}(f)} X_w\right) \oplus \left(\bigoplus\nolimits_{w\in\mathcal{J}_i\subseteq\mathcal{W}_{\mathsf{and}}(f)} X_w\right) \\ &= \bigoplus\nolimits_{w\in\mathcal{I}_i\ominus\mathcal{J}_i} X_w \in \{0,1\}^{\lambda}. \end{aligned} \tag{26}$$

For (5), we can use (26) to rewrite it as

$$\bigoplus\nolimits_{i\in(\mathcal{I}_w\ominus\mathcal{I}_{w'})\ominus(\mathcal{J}_w\ominus\mathcal{J}_{w'})} X_i = k_u^g \oplus k_{u'}^{g'} \in \{0,1\}^{\lambda}.$$

According to SimIn, each X_i, which is sampled in the step 1 or computed in the step 13, has at least $\lambda-1$ random non-LSBs. Therefore, the equality holds with probability at most $2^{-(\lambda-1)}$ for some fixed distinct k_u^g and $k_{u'}^{g'}$ (or equivalently, (g,u) and (g',u')). If $\mathcal{I}_w = \mathcal{I}_{w'}$ and $\mathcal{J}_w = \mathcal{J}_{w'}$, this probability will be zero for the distinct k_u^g and $k_{u'}^{g'}$. For (6), the worst case is that U_u^g and $U_{u'}^{g'}$ are used for the same `AND` gate with an output wire in $\mathcal{Z}(f)$. In this case, the XOR $U_u^g \oplus U_{u'}^{g'}$ has at least $\lambda-1$ random non-LSBs due to some X_c sampled in the step 1 of SimIn. Such non-LSBs are independent of other (previously fixed) active labels, including X_w and $X_{w'}$. So, the equality holds with probability at most $2^{-(\lambda-1)}$ for some fixed (g,u) and (g',u').

In the same gate g, the upper $\lambda-1$ bits of $U_0^g \oplus U_1^g$ are uniform so that (7) holds with probability at most $2^{-(\lambda-1)}$. Otherwise, the distinct $g \neq g'$ implies that, for every $u, u' \in \{0,1\}$, $U_u^g \oplus U_{u'}^{g'} \in \{0,1\}^{\lambda}$ is uniform. As a result, each of (8), (9), (10), and (11) holds with probability $2^{-\lambda}$.

Taking a union bound over the above cases, we have

$$\begin{aligned} &\Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_1] \\ &\quad \leq \frac{2\,|f|\,(2\,|f|-1)+|f|^2}{2^{\lambda-1}} = \frac{5\,|f|^2-2\,|f|}{2^{\lambda-1}}. \end{aligned} \tag{31}$$

Then, consider bad_2. From (12) (resp., (13)), we have $\mathit{\Delta} = \alpha \oplus X_a \oplus k_0^g$ (resp., $\mathit{\Delta} = \alpha \oplus X_b \oplus k_1^g$), occurring with probability $2^{-(\lambda-1)}$ due to the randomness of $\mathit{\Delta}$. In (14), if $s_b \oplus x_b = 0$, linear orthomorphism σ ensures that

$$\mathit{\Delta} = \sigma^{-1}(\beta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g)),$$

which occurs with probability $2^{-(\lambda-1)}$; if $s_b \oplus x_b = 1$, according to permutation $\sigma'(x) := \sigma(x) \oplus x$ well-defined from σ, it holds with the same probability that

$$\mathit{\Delta} = \sigma'^{-1}(\beta \oplus G_0^g \oplus U_0^g \oplus \sigma(X_a \oplus k_0^g)).$$

Similar result holds for (15). Taking a union bound over all pairs, we have

$$\Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_2] \leq \frac{4\,|f|\cdot(q_1+q_2+q_3)}{2^{\lambda-1}}. \tag{32}$$

Finally, consider bad_3. Recall that each X_i has at least $\lambda-1$ random non-LSBs. Since these bits are independent of $\cup_{\ell=1}^{2}\mathcal{K}_\ell$, each of (16) and (17) holds with probability at most $2^{-(\lambda-1)}$ for some fixed $(\alpha,\ldots)$ and g. For each of (18) and (19), the mask sampled in the step 9 of SimIn (resp., the direct sampling in the step 9 or 12 of SimIn) ensures that $U_0^g \in \{0,1\}^{\lambda}$ (resp., $U_1^g \in \{0,1\}^{\lambda}$) is uniform and independent of X_a (resp., X_b) or $\cup_{\ell=1}^{2}\mathcal{K}_\ell$. So, each equality holds with probability $2^{-\lambda} < 2^{-(\lambda-1)}$ for some fixed $(\ldots,\beta)$ and g. It follows from a union bound that

$$\Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_3] \leq \frac{4\,|f|\cdot(q_1+q_2)}{2^{\lambda-1}}. \tag{33}$$

We have a bound ε_1 from (31), (32), and (33):

$$\begin{aligned} &\Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[Y(\omega)\in\mathcal{T}_{\mathsf{bad}}] \\ &\quad = \Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3] \\ &\quad \leq \Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_1] + \Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_2] + \Pr_{\omega\leftarrow\mathit{\Omega}_{\mathsf{ideal}}}[\mathsf{bad}_3] \\ &\quad \leq \frac{5\,|f|^2-2\,|f|+8\,|f|\cdot(q+2\,|f|)}{2^{\lambda-1}} \\ &\quad = \frac{8qs+21s^2-2s}{2^{\lambda-1}} = \varepsilon_1. \end{aligned}$$

The above ε_1, ε_2 and the H-coefficient technique lead to this theorem. □

Appendix B. Adaptive Security of Three-Halves

We consider the three-halves scheme [23] with the computational optimization using both halves of hash outputs (see Section 6.2 therein). As recalled in Remark 2, it uses a counter per wire rather than an identifier per gate to compute tweaks in hash computation. Similar to half-gates, the three-halves implementation in the pRPM sends decoding table d

in the offline phase while that in the npRPM will postpone d to the online phase.

At a high level, three-halves works similarly like half-gates to perform circuit garbling and evaluation. It maintains an equivalent invariant for each wire i: its active label $\boldsymbol{x}_i = \boldsymbol{w}_i \oplus (p_i \oplus x_i)\boldsymbol{\Delta}$ for its wire label $\boldsymbol{w}_i$ of bit 0 with $\mathsf{lsb}(\boldsymbol{w}_i) = 0$, truth bit x_i, permuted bit p_i, and global key $\boldsymbol{\Delta}$ with $\mathsf{lsb}(\boldsymbol{\Delta}) = 1$. However, compared to half-gates, three-halves has two important changes: (i) slicing: it will slice global key $\boldsymbol{\Delta}$, wire labels, and their hash outputs by half so that the resulting halves are used for more complex *linear combination*, and (ii) dicing: some coefficients (called *control bits*) in such linear combination are randomly sampled by the garbler and masked by some bits of hash outputs, and these control bits can be computed by the evaluator in circuit evaluation. Despite of these changes, the three-halves implementations in both npRPM and pRPM can be proven adaptively secure using our framework in Section 2.2 since the circuit garbling still uses the linear combination of hash outputs to mask truth tables and control bits. That is, by inlining a linear-orthomorphism-based hash instantiation into the three-halves implementations and explicitly appending control bits to transcripts, we can follow similar transcript padding and probability analysis of half-gates (see Section 4 and Appendix A) to prove the adaptive security of three-halves in the two RPMs. This probability analysis has been outlined in Section 2.2, and see the following for details.

B.1. More Preliminaries

We define the following notation in the appendices. Let bold lowercase letters (e.g., $\boldsymbol{a}$) denote column vectors and bold uppercase letters (e.g., $\boldsymbol{A}$) denote matrices. Let $\boldsymbol{I}_n$ denote the n-by-n identity matrix. Let $\mathsf{Half}_0(\boldsymbol{a}) \in \mathbb{F}_{2^n}$ (resp., $\mathsf{Half}_1(\boldsymbol{a}) \in \mathbb{F}_{2^n}$) denote the lower (resp., upper) half of vector $\boldsymbol{a} \in \mathbb{F}_{2^n}^2$. We can also define $\mathsf{lsb}(\boldsymbol{a}) := \mathsf{lsb}(\mathsf{Half}_0(\boldsymbol{a}))$ for vector $\boldsymbol{a} \in \mathbb{F}_{2^n}^2$. Let $\otimes$ denote the Kronecker product of matrices. Let $\boldsymbol{A}^+$ denote the left inverse of matrix $\boldsymbol{A}$. We will use $\mathbb{F}_{2^n}$, $\mathbb{F}_2^n$, and $\{0,1\}^n$ interchangeably. A generalized definition of linear orthomorphism is described as follows:

Definition 2 (Linear orthomorphism)**.** *A permutation $\sigma : \mathbb{G} \to \mathbb{G}$ over an additive Abelian group $\mathbb{G}$ is called a linear orthomorphism for a function family $\mathcal{L}$ of some linear functions from $\mathbb{G}$ to $\mathbb{G}$, if (i) $\sigma(x+y) = \sigma(x) + \sigma(y)$ for any $x, y \in \mathbb{G}$, (ii) $\sigma'(x) := \sigma(x) - L(x)$ is also a permutation for every $L \in \mathcal{L}$, and (iii) σ, σ' and their inverses are efficiently computable. We will simply call σ a linear orthomorphism if $\mathcal{L}$ contains only the identity function.*

For three-halves [23], we require a linear orthomorphism $\sigma : \mathbb{F}_{2^{\lambda/2+1}}^2 \to \mathbb{F}_{2^{\lambda/2+1}}^2$ for the function family

$$\mathcal{L} = \left\{ L_{\xi_1,\xi_2,\xi_3,\xi_4} : \mathbb{F}_{2^{\lambda/2+1}}^2 \to \mathbb{F}_{2^{\lambda/2+1}}^2 \right\}_{\xi_1,\xi_2,\xi_3,\xi_4 \in \mathbb{F}_2},$$

$$L_{\xi_1,\xi_2,\xi_3,\xi_4}\left(\begin{bmatrix} x_L \\ x_R \end{bmatrix} \right) = \begin{bmatrix} \xi_1 x_L \oplus \xi_2 x_R \\ \xi_3 x_L \oplus \xi_4 x_R \end{bmatrix}.$$

The following linear orthomorphism σ is used therein:

$$\sigma\left(\begin{bmatrix} x_L \\ x_R \end{bmatrix} \right) = \begin{bmatrix} c \cdot x_L \\ c \cdot x_R \end{bmatrix}$$

where $c \in \mathbb{F}_{2^{\lambda/2+1}} \setminus \mathbb{F}_{2^2}$ is a fixed element and the multiplication is in $\mathbb{F}_{2^{\lambda/2+1}}$.

Public parameters: (See Appendix B.4 for concrete instantiations)

- $\boldsymbol{M} \in \mathbb{F}_2^{8\times 6}$ with $\boldsymbol{K} \in \mathbb{F}_2^{3\times 8}$ from the co-kernel basis of $\boldsymbol{M}$, i.e., $\boldsymbol{K}\boldsymbol{M} = \boldsymbol{0}_{3\times 6}$.
- $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_{00} & \boldsymbol{V}_{01} & \boldsymbol{V}_{10} & \boldsymbol{V}_{11} \end{bmatrix}^\mathsf{T} \in (\mathbb{F}_2^{2\times 5})^4 \equiv \mathbb{F}_2^{8\times 5}$ with left inverse $\boldsymbol{V}^+ \in \mathbb{F}_2^{5\times 8}$.
- Basis matrices $\boldsymbol{S}_L, \boldsymbol{S}_R \in \mathbb{F}_2^{2\times 4}$.
- Control matrices $\boldsymbol{R}'_1, \boldsymbol{R}'_2 \in \mathbb{F}_2^{4\times 2}$, $\boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{R}_{p,00} & \boldsymbol{R}_{p,01} & \boldsymbol{R}_{p,10} & \boldsymbol{R}_{p,11} \end{bmatrix}^\mathsf{T} \in (\mathbb{F}_2^{2\times 4})^4 \equiv \mathbb{F}_2^{8\times 4}$.
- A distribution $\mathcal{R}_0$ over $(\mathbb{F}_2^{1\times 2})^4 \equiv \mathbb{F}_2^{4\times 2}$ such that, for every $\boldsymbol{R}'_\$ = \begin{bmatrix} \boldsymbol{R}'_{\$,00} & \boldsymbol{R}'_{\$,01} & \boldsymbol{R}'_{\$,10} & \boldsymbol{R}'_{\$,11} \end{bmatrix}^\mathsf{T} \leftarrow \mathcal{R}_0$, (i) $\boldsymbol{K}\begin{bmatrix} \widehat{\boldsymbol{R}}_{\$,00} & \widehat{\boldsymbol{R}}_{\$,01} & \widehat{\boldsymbol{R}}_{\$,10} & \widehat{\boldsymbol{R}}_{\$,11} \end{bmatrix}^\mathsf{T} = \boldsymbol{0}_{3\times 6}$, where $\widehat{\boldsymbol{R}}_{\$,ij} := (\boldsymbol{R}'_{\$,ij} \otimes \boldsymbol{I}_2) \begin{bmatrix} \boldsymbol{S}_L \\ \boldsymbol{S}_R \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & i \\ 0 & 1 & j \end{bmatrix} \otimes \boldsymbol{I}_2 \right)$ for every $i, j \in \mathbb{F}_2$, and (ii) for every $i, j \in \mathbb{F}_2$, the marginal distribution of $\boldsymbol{R}'_{\$,ij}$ is uniform.

TH.SampleR($\boldsymbol{t}$):

```
$\zeta_1 := g(\overline{p_a}, p_b) \oplus g(\overline{p_a}, \overline{p_b})$, $\zeta_2 := g(p_a, \overline{p_b}) \oplus g(\overline{p_a}, \overline{p_b})$
$\boldsymbol{R}'_\$ \leftarrow \mathcal{R}_0$, $\begin{bmatrix} \boldsymbol{r}_{00}^\mathsf{T} \\ \boldsymbol{r}_{01}^\mathsf{T} \\ \boldsymbol{r}_{10}^\mathsf{T} \\ \boldsymbol{r}_{11}^\mathsf{T} \end{bmatrix} = \begin{bmatrix} r_{00L} & r_{00R} \\ r_{01L} & r_{01R} \\ r_{10L} & r_{10R} \\ r_{11L} & r_{11R} \end{bmatrix} := \boldsymbol{R}'_\$ \oplus \zeta_1 \boldsymbol{R}'_1 \oplus \zeta_2 \boldsymbol{R}'_2$
for $i, j \in \mathbb{F}_2$ do $\boldsymbol{R}_{ij} := (\boldsymbol{r}_{ij}^\mathsf{T} \otimes \boldsymbol{I}_2) \begin{bmatrix} \boldsymbol{S}_L \\ \boldsymbol{S}_R \end{bmatrix} \oplus \boldsymbol{R}_{p,ij}$,
    $\widehat{\boldsymbol{R}}_{ij} := \boldsymbol{R}_{ij} \left( \begin{bmatrix} 1 & 0 & i \\ 0 & 1 & j \end{bmatrix} \otimes \boldsymbol{I}_2 \right)$
$\widehat{\boldsymbol{R}} := \begin{bmatrix} \widehat{\boldsymbol{R}}_{00} & \widehat{\boldsymbol{R}}_{01} & \widehat{\boldsymbol{R}}_{10} & \widehat{\boldsymbol{R}}_{11} \end{bmatrix}^\mathsf{T}$,
    $\boldsymbol{r} := \begin{bmatrix} r_{00L} & r_{00R} & r_{01L} & r_{01R} & r_{10L} & r_{10R} & r_{11L} & r_{11R} \end{bmatrix}^\mathsf{T}$
return $(\widehat{\boldsymbol{R}}, \boldsymbol{r})$
```

TH.DecodeR($\boldsymbol{r}_{ij}, i, j$):

```
return $\boldsymbol{R}_{ij} := (\boldsymbol{r}_{ij}^\mathsf{T} \otimes \boldsymbol{I}_2) \begin{bmatrix} \boldsymbol{S}_L \\ \boldsymbol{S}_R \end{bmatrix} \oplus \boldsymbol{R}_{p,ij}$
```

Figure 7: Three-halves garbling scheme [23] (Part I).

In addition to the circuit notations in Section 3.1 and Appendix A, three-halves and our simulators also require the following notations for a circuit f with fan-in two and fan-out one:

- For an AND gate $g \in \mathcal{G}_{\mathsf{and}}(f)$, let $\mathsf{in}_2(g)$ denote the *global* alias of all wires that syntactically use a pair of labels $(W_a \oplus W_b, W_a \oplus W_b \oplus \Delta)$ with $a := \mathsf{in}_0(g)$ and $b := \mathsf{in}_1(g)$ in free-XOR. If no such a wire exists, we define it artificially. See [23, Section 6.2] for details.
- $\mathcal{W}_\cup(f) := \cup_{g \in \mathcal{G}_{\mathsf{and}}(f)} \{\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{in}_2(g)\}$.
- Let $n_i(f) \geq 0$ denote the number of times the wire i is used for the input to the AND gates in f.
- $N := \sum_{i \in \mathcal{W}_\cup(f)} \lceil n_i(f)/2 \rceil$.
- $M := |\{i \in \mathcal{W}_\cup(f) \mid n_i(f) \text{ is odd}\}|$.

B.2. Adaptive Security in pRPM

Theorem 3. *Let $\mathsf{H}(\boldsymbol{x}, k) = \pi((\boldsymbol{0} \,\|\, \boldsymbol{x}) \oplus k) \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}) \oplus k)$ be a tweakable hash function where $\boldsymbol{x} \in \mathbb{F}_{2^{\lambda/2}}^2, k \in \mathbb{F}_{2^{\lambda/2+1}}^2$, $\pi \in \mathcal{S}_{\lambda+2}$ is random permutation, and $\sigma : \mathbb{F}_{2^{\lambda/2+1}}^2 \to \mathbb{F}_{2^{\lambda/2+1}}^2$ is a linear orthomorphism for the function family*

$$\mathcal{L} = \left\{ L_{\xi_1,\xi_2,\xi_3,\xi_4} : \mathbb{F}_{2^{\lambda/2+1}}^2 \to \mathbb{F}_{2^{\lambda/2+1}}^2 \right\}_{\xi_1,\xi_2,\xi_3,\xi_4 \in \mathbb{F}_2},$$

$$L_{\xi_1,\xi_2,\xi_3,\xi_4}\left(\begin{bmatrix} x_L \\ x_R \end{bmatrix} \right) = \begin{bmatrix} \xi_1 x_L \oplus \xi_2 x_R \\ \xi_3 x_L \oplus \xi_4 x_R \end{bmatrix}.$$

```
$\mathsf{TH.Garble}^{\pi^{\pm 1}(\cdot)}(f)$:
$\boldsymbol{\Delta} \leftarrow \begin{bmatrix} \mathbb{F}_{2^{\lambda/2}} \\ \mathbb{F}_{2^{\lambda/2-1}} \,\|\, 1 \end{bmatrix}$
for $i \in \mathcal{W}_{\mathsf{in}}(f)$ do
    $p_i \leftarrow \mathbb{F}_2$, $\boldsymbol{w}_i \leftarrow \begin{bmatrix} \mathbb{F}_{2^{\lambda/2}} \\ \mathbb{F}_{2^{\lambda/2-1}} \,\|\, 0 \end{bmatrix}$
for $i \in \mathcal{W}_{\cup}(f)$ do $\mathsf{ctr}_i := 0$
for $g \in \mathcal{G}(f)$ in order do
    $(a, b, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
    if $\mathsf{type}(g) = \mathtt{XOR}$ then $p_c := p_a \oplus p_b$, $\boldsymbol{w}_c := \boldsymbol{w}_a \oplus \boldsymbol{w}_b$
    else if $\mathsf{type}(g) = \mathtt{AND}$ then
        $\Gamma := \mathsf{in}_2(g)$
        $(\chi_a, \rho_a) := (\lfloor \mathsf{ctr}_a/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_a))$, $\mathsf{ctr}_a := \mathsf{ctr}_a + 1$
        $(\chi_b, \rho_b) := (\lfloor \mathsf{ctr}_b/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_b))$, $\mathsf{ctr}_b := \mathsf{ctr}_b + 1$
        $(\chi_\Gamma, \rho_\Gamma) := (\lfloor \mathsf{ctr}_\Gamma/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_\Gamma))$, $\mathsf{ctr}_\Gamma := \mathsf{ctr}_\Gamma + 1$
        $\boldsymbol{t}^g := \left[ g(p_a, p_b) \;\; g(p_a, \overline{p_b}) \;\; g(\overline{p_a}, p_b) \;\; g(\overline{p_a}, \overline{p_b}) \right]^{\mathsf{T}} \in \mathbb{F}_2^4$
        $(\widehat{\boldsymbol{R}}^g, \boldsymbol{r}^g) \leftarrow \mathsf{TH.SampleR}(\boldsymbol{t}^g)$
        Compute $(\boldsymbol{c}^g, \boldsymbol{g}^g, \boldsymbol{z}^g) \in \mathbb{F}_{2^{\lambda/2}}^2 \times \mathbb{F}_{2^{\lambda/2}}^3 \times \mathbb{F}_2^5$ as follows:
            $\boldsymbol{h}^g := \begin{bmatrix} \mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{w}_a, a \,\|\, \chi_a)) \\ \mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{w}_a \oplus \boldsymbol{\Delta}, a \,\|\, \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{w}_b, b \,\|\, \chi_b)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{w}_b \oplus \boldsymbol{\Delta}, b \,\|\, \chi_b)) \\ \mathsf{Half}_{\rho_\Gamma}(\mathsf{H}(\boldsymbol{w}_a \oplus \boldsymbol{w}_b, \Gamma \,\|\, \chi_\Gamma)) \\ \mathsf{Half}_{\rho_\Gamma}(\mathsf{H}(\boldsymbol{w}_a \oplus \boldsymbol{w}_b \oplus \boldsymbol{\Delta}, \Gamma \,\|\, \chi_\Gamma)) \end{bmatrix} \in \mathbb{F}_{2^{\lambda/2+1}}^6,$
            $\left( \boldsymbol{z}^g \,\middle\|\, \begin{bmatrix} \boldsymbol{c}^g \\ \boldsymbol{g}^g \end{bmatrix} \right)$
            $:= \boldsymbol{V}^{+} \left( \boldsymbol{M}\boldsymbol{h}^g \oplus \left( \boldsymbol{r}^g \,\middle\|\, (\widehat{\boldsymbol{R}}^g \oplus ([\boldsymbol{0}_{4\times 2} \;\; \boldsymbol{t}^g] \otimes \boldsymbol{I}_2)) \begin{bmatrix} \boldsymbol{w}_a \\ \boldsymbol{w}_b \\ \boldsymbol{\Delta} \end{bmatrix} \right) \right)$
        $p_c := \mathsf{lsb}(\boldsymbol{c}^g)$, $\boldsymbol{w}_c := \boldsymbol{c}^g \oplus p_c\boldsymbol{\Delta}$
for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $d_i := p_i$
return $\widehat{f} := (f' := f, F := \{(\boldsymbol{g}^g, \boldsymbol{z}^g)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}, d)$, $k := (f, \boldsymbol{\Delta}, p, \boldsymbol{w})$
```

```
$\mathsf{TH.Encode}^{\pi^{\pm 1}(\cdot)}(k, x)$:
for $i \in \mathcal{W}_{\mathsf{in}}(f)$ do
    $\boldsymbol{x}_i := \boldsymbol{w}_i \oplus (p_i \oplus x_i)\boldsymbol{\Delta}$
return $\widehat{x} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}$
```

```
$\mathsf{TH.DecEval}^{\pi^{\pm 1}(\cdot)}(\widehat{f}, \widehat{x})$:
for $i \in \mathcal{W}_{\cup}(f)$ do $\mathsf{ctr}_i := 0$
for $g \in \mathcal{G}(f)$ in order do
    $(a, b, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
    if $\mathsf{type}(g) = \mathtt{XOR}$ then $\boldsymbol{x}_c := \boldsymbol{x}_a \oplus \boldsymbol{x}_b$
    else if $\mathsf{type}(g) = \mathtt{AND}$ then
        $\Gamma := \mathsf{in}_2(g)$
        $(\chi_a, \rho_a) := (\lfloor \mathsf{ctr}_a/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_a))$, $\mathsf{ctr}_a := \mathsf{ctr}_a + 1$
        $(\chi_b, \rho_b) := (\lfloor \mathsf{ctr}_b/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_b))$, $\mathsf{ctr}_b := \mathsf{ctr}_b + 1$
        $(\chi_\Gamma, \rho_\Gamma) := (\lfloor \mathsf{ctr}_\Gamma/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_\Gamma))$, $\mathsf{ctr}_\Gamma := \mathsf{ctr}_\Gamma + 1$
        $s_a := \mathsf{lsb}(\boldsymbol{x}_a)$, $s_b := \mathsf{lsb}(\boldsymbol{x}_b)$
        $(\boldsymbol{r}^g_{s_a s_b} \,\|\, \boldsymbol{m}^g_{s_a s_b}) :=$
            $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a, a \,\|\, \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b, b \,\|\, \chi_b)) \\ \mathsf{Half}_{\rho_\Gamma}(\mathsf{H}(\boldsymbol{x}_a \oplus \boldsymbol{x}_b, \Gamma \,\|\, \chi_\Gamma)) \end{bmatrix}$
            $\oplus \boldsymbol{V}_{s_a s_b} \left( \boldsymbol{z}^g \,\middle\|\, \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g}^g \end{bmatrix} \right)$
        $\boldsymbol{R}^g_{s_a s_b} := \mathsf{TH.DecodeR}(\boldsymbol{r}^g_{s_a s_b}, s_a, s_b)$
        $\boldsymbol{x}_c := \boldsymbol{m}^g_{s_a s_b} \oplus \boldsymbol{R}^g_{s_a s_b} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix}$
for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $y_i := d_i \oplus \mathsf{lsb}(\boldsymbol{x}_i)$
return $y$
```

Figure 7: Three-halves garbling scheme [23] (Part II).

Then, three-halves (Figure 7) is a $(\lambda + 2)$-garbling scheme with (q, s, ε)-adaptive security in the pRPM, where $\varepsilon = (48qs + 189s^2)/2^{\lambda+1}$.

Proof. The correctness has been proved in the original work [23]. We only need to consider the simulation.

```
$\mathsf{SimF}(f)$:
1: $F \leftarrow (\mathbb{F}_{2^{\lambda/2}}^3 \times \mathbb{F}_2^5)^{|f|}$, $d \leftarrow \mathbb{F}_2^{|\mathcal{W}_{\mathsf{out}}(f)|}$ such that
     (i) For each $i \in [1, m]$ and each wire $j \in \mathcal{Y}_i(f)$, $d_j = d_{y_i(f)}$.
    (ii) For each $(\mu_1, \ldots, \mu_m) \in \mathbb{F}_2^m$ such that $\ominus_{i \in [1,m], \mu_i = 1} \mathcal{X}_i(f) = \varnothing$, $\oplus_{i \in [1,m]} \mu_i \cdot d_{y_i(f)} = 0$.
2: return $\widehat{f} := (f' := f, F, d)$, $\mathsf{st}_{\mathsf{sim}} := (\mathsf{st}_{\mathsf{sim}}, f, F, d)$

$\mathsf{SimIn}(f(x))$:
1: $\{\boldsymbol{x}_i\}_{i \in \mathcal{X}(f)} \leftarrow (\mathbb{F}_{2^{\lambda/2}}^2)^{|\mathcal{X}(f)|}$ such that, for each $j \in [1, m]$, $\mathsf{lsb}(\oplus_{i \in \mathcal{X}_j(f)} \boldsymbol{x}_i) = d_{y_j(f)} \oplus f(x)_{y_j(f)}$.
2: for $i \in \mathcal{W}_{\cup}(f)$ do $\mathsf{ctr}_i := 0$
3: for $g \in \mathcal{G}(f)$ in order do
4:     $(a, b, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
5:     if $\mathsf{type}(g) = \mathtt{XOR}$ then $\boldsymbol{x}_c := \boldsymbol{x}_a \oplus \boldsymbol{x}_b$
6:     else if $\mathsf{type}(g) = \mathtt{AND}$ then
7:         $\Gamma := \mathsf{in}_2(g)$
8:         $(\chi_a, \rho_a) := (\lfloor \mathsf{ctr}_a/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_a))$, $\mathsf{ctr}_a := \mathsf{ctr}_a + 1$
9:         $(\chi_b, \rho_b) := (\lfloor \mathsf{ctr}_b/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_b))$, $\mathsf{ctr}_b := \mathsf{ctr}_b + 1$
10:        $(\chi_\Gamma, \rho_\Gamma) := (\lfloor \mathsf{ctr}_\Gamma/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_\Gamma))$, $\mathsf{ctr}_\Gamma := \mathsf{ctr}_\Gamma + 1$
11:        $s_a := \mathsf{lsb}(\boldsymbol{x}_a)$, $s_b := \mathsf{lsb}(\boldsymbol{x}_b)$
12:        if $c \in \mathcal{Z}(f)$ then
13:            $\boldsymbol{r}^g_{s_a s_b} \leftarrow \mathbb{F}_2^2$, $\boldsymbol{R}^g_{s_a s_b} := \mathsf{TH.DecodeR}(\boldsymbol{r}^g_{s_a s_b}, s_a, s_b)$
14:            $\mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \leftarrow \mathbb{F}_{2^{\lambda/2+1}}$
               $\begin{bmatrix} \mathsf{Half}_{\rho_a}(\boldsymbol{u}_a^{\chi_a}) \\ \mathsf{Half}_{\rho_b}(\boldsymbol{u}_b^{\chi_b}) \end{bmatrix} :=$
15:                $\begin{bmatrix} \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \\ \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \end{bmatrix} \oplus \left( \boldsymbol{r}^g_{s_a s_b} \,\middle\|\, \boldsymbol{x}_c \oplus \boldsymbol{R}^g_{s_a s_b} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix} \right)$
                   $\oplus \boldsymbol{V}_{s_a s_b} \left( \boldsymbol{z}^g \,\middle\|\, \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g}^g \end{bmatrix} \right)$
16:        else if $c \notin \mathcal{Z}(f)$ then
17:            $\mathsf{Half}_{\rho_a}(\boldsymbol{u}_a^{\chi_a}), \mathsf{Half}_{\rho_b}(\boldsymbol{u}_b^{\chi_b}), \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \leftarrow \mathbb{F}_{2^{\lambda/2+1}}$
               $(\boldsymbol{r}^g_{s_a s_b} \,\|\, \boldsymbol{m}^g_{s_a s_b})$
18:                $:= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathsf{Half}_{\rho_a}(\boldsymbol{u}_a^{\chi_a}) \\ \mathsf{Half}_{\rho_b}(\boldsymbol{u}_b^{\chi_b}) \\ \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \end{bmatrix} \oplus \boldsymbol{V}_{s_a s_b} \left( \boldsymbol{z}^g \,\middle\|\, \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g}^g \end{bmatrix} \right)$
19:            $\boldsymbol{R}^g_{s_a s_b} := \mathsf{TH.DecodeR}(\boldsymbol{r}^g_{s_a s_b}, s_a, s_b)$
20:            $\boldsymbol{x}_c := \boldsymbol{m}^g_{s_a s_b} \oplus \boldsymbol{R}^g_{s_a s_b} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix}$
21: for $i \in \mathcal{W}_{\cup}(f)$ do
22:     for $j \in [0, \lfloor n_i(f)/2 \rfloor - 1]$ do
23:         (Programming) Add a pair
                $\left( (\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j), \boldsymbol{u}_i^j \oplus \sigma\left( (\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j) \right) \right)$
            to the list $\mathcal{Q}$ kept in $\mathsf{st}_{\mathsf{sim}}$, if it causes no pre-image collision or image collision in $\mathcal{Q}$, to ensure
                $\underbrace{\pi\left( (\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j) \right) \oplus \sigma\left( (\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j) \right)}_{\mathsf{H}(\boldsymbol{x}_i, i \,\|\, j)}$
                $= \boldsymbol{u}_i^j = \begin{bmatrix} \mathsf{Half}_1(\boldsymbol{u}_i^j) \\ \mathsf{Half}_0(\boldsymbol{u}_i^j) \end{bmatrix}.$
24:     if $n_i(f)$ is odd then
25:         $j := \lfloor n_i(f)/2 \rfloor$, $\mathsf{Half}_1(\boldsymbol{u}_i^j) \leftarrow \mathbb{F}_{2^{\lambda/2+1}}$ and repeat the step 23 for this $j$
26: return $\widehat{x} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}$, $\mathsf{st}_{\mathsf{sim}} := (\mathsf{st}_{\mathsf{sim}}, \widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{r}})$ where $\mathsf{st}_{\mathsf{sim}}$ on the right hand has an updated list $\mathcal{Q}$, $\widetilde{\boldsymbol{x}} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}(f)}$, $\widetilde{\boldsymbol{u}} := \{\boldsymbol{u}_i^j\}_{i \in \mathcal{W}_{\cup}(f), j \in [0, \lceil n_i(f)/2 \rceil - 1]}$, $\widetilde{\boldsymbol{r}} := \{\boldsymbol{r}^g_{s_a s_b}\}_{g \in \mathcal{G}_{\mathsf{and}}(f), (a,b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))}$.
```

Figure 8: Our simulator for three-halves in the pRPM.

Our simulator Sim consists of $(\mathsf{SimF}, \mathsf{SimIn})$ in Figure 8 and $\mathsf{SimP}^{\pm 1}$, which emulates the random permutation and its inverse on-the-fly. More specifically, there is a list $\mathcal{Q}$ of query-response pairs in internal state $\mathsf{st}_{\mathsf{sim}}$.

Upon receiving forward query α (resp., backward query β) from $\mathcal{A}_i$ to SimP (resp., SimP^{-1}), it reads $\mathcal{Q}$ from $\mathsf{st}_{\mathsf{sim}}$ and checks whether $\exists(\alpha,\gamma) \in \mathcal{Q}$ (resp., $\exists(\gamma,\beta) \in \mathcal{Q}$). If true, it returns γ as response; otherwise it samples $\gamma \leftarrow \{s \in \mathbb{F}^2_{2^{\lambda/2+1}} \mid (\dots, s) \notin \mathcal{Q}\}$ (resp., $\gamma \leftarrow \{s \in \mathbb{F}^2_{2^{\lambda/2+1}} \mid (s, \dots) \notin \mathcal{Q}\}$), adds (α,γ) (resp., (γ,β)) to $\mathcal{Q}$, and returns γ as response. The programming is the step 23 of SimIn, where (α,β) is added to $\mathcal{Q}$ if and only if there is no pre-image collision with $(\alpha,\dots) \in \mathcal{Q}$ or image collision with $(\dots,\beta) \in \mathcal{Q}$.

To see that Sim is PPT, we note that the circuit-dependent notation can be efficiently computed by traversing the polynomial-size circuit f. Then, the crux is to show that the step 1 in both SimF and SimIn can be polynomial-time.

The runtime of the step 1 of SimF is dominated by the runtime of iterating through all qualified $(\mu_1,\dots,\mu_m) \in \mathbb{F}_2^m$. To find the qualified vectors, one can interpret each $\mathcal{X}_i(f)$ as a one-hot *non-zero* column vector in the space $\mathbb{F}_2^{|\mathcal{W}(f)|}$ and derive a $|\mathcal{W}(f)|$-by-m matrix $\boldsymbol{E}$ from these column vectors. One can check that all qualified vectors fall in the kernel of $\boldsymbol{E}$. This kernel can be efficiently computed from the Gaussian elimination on $\boldsymbol{E}$ and is a subspace spanned by $m - \mathsf{rank}(\boldsymbol{E})$ basis vectors. So, the step 1 of SimF only needs to iterate through these basis vectors, and the other qualified vectors must satisfy the constraints as they are in the subspace. As a result, the step 1 of SimF runs in polynomial time due to the Gaussian elimination plus a linear-time pass to assign random or constrained values to $d_{y_i(f)}$'s according to the $m - \mathsf{rank}(\boldsymbol{E})$ basis vectors.

The step 1 of SimIn only requires one linear-time pass to assign constrained or random values to the active labels so it runs in polynomial time. The linear constraint on the LSBs of these active labels has rank $\mathsf{rank}(\boldsymbol{E})$ and is satisfiable for the $d_{y_i(f)}$'s assigned in SimF.

Then, we fix z and $\mathcal{A}$. We will use the H-coefficient technique (Section 3.4) with transcript padding to bound the advantage of distinguishing between the real world (i.e., the adaptive experiment that uses the three-halves scheme) and the ideal world (i.e., the adaptive experiment that uses simulator Sim). In this technique, we consider the computationally unbounded non-uniform adversary $\mathcal{A} = \mathcal{A}(z)$ and compute $\varepsilon_1, \varepsilon_2$ as follows.

Transcript padding. In either world, $\mathcal{A}$ will interact with an integrated oracle that acts as the two-round challenger in the adaptive experiment and provides interfaces $\pi^{*\pm1} \in \{\pi^{\pm1}, \mathsf{SimP}^{\pm1}\}$ for forward/backward permutation queries. $\mathcal{A}$ can learn $\pi^*(\alpha) = \beta$ if and only if it sent forward query α to π^* and received response β, or sent backward query β to π^{*-1} and received response α.

To compute $\varepsilon_1, \varepsilon_2$ more easily, we ask the oracle to send more messages to $\mathcal{A}$ and $\mathcal{A}$ to make extra queries (in addition to the supposed q queries) in both two worlds. More specifically,

- Upon receiving x from $\mathcal{A}$, the oracle sends $\widetilde{\boldsymbol{x}} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}(f)}$ instead of $\widehat{x} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}$ to $\mathcal{A}$. In addition to the active input labels in $\widehat{x}$, the former also gives the active internal and output labels. In the real world, the oracle can run $\mathsf{TH.DecEval}^{\pi^{\pm1}(\cdot)}$, which determines other active labels in $\widetilde{\boldsymbol{x}}$. In the ideal world, this $\widetilde{\boldsymbol{x}}$ can be directly output by SimIn.
- Along with $\widetilde{\boldsymbol{x}}$, the oracle sends $\widetilde{x} := \{x_i\}_{i \in \mathcal{W}(f)}$ to $\mathcal{A}$, which denote the wire truth values in the evaluation of $f(x)$. Both two oracles "echo" these values, which are self-evident to $\mathcal{A}$, to explicitly include them in transcripts. In the experiment, the real-world oracle uses $x = \{x_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}$ in $\mathsf{TH.Encode}^{\pi^{\pm1}(\cdot)}$, but the ideal-world oracle can only use $f(x) = \{x_i\}_{i \in \mathcal{W}_{\mathsf{out}}(f)}$ in SimIn.
- Along with $\widetilde{\boldsymbol{x}}$, the oracle sends $\widetilde{\boldsymbol{u}} := \{\boldsymbol{u}_i^j\}_{i \in \mathcal{W}_\cup(f), j \in [0, \lceil n_i(f)/2 \rceil - 1]}$ to $\mathcal{A}$. In the real world, the oracle computes $\boldsymbol{u}_i^j := \mathsf{H}(\boldsymbol{x}_i, i \,\|\, j)$ for each $i \in \mathcal{W}_\cup(f)$ and $j \in [0, \lceil n_i(f)/2 \rceil - 1]$. In the ideal world, this $\widetilde{\boldsymbol{u}}$ is output by SimIn and gives the "hash outputs" fixed by the random tape, which is specified by the oracle to run the programming therein.
- Along with $\widetilde{\boldsymbol{x}}$, the oracle sends $\widetilde{\boldsymbol{r}} := \{\boldsymbol{r}^g_{s_a s_b}\}_{g \in \mathcal{G}_{\mathsf{and}}(f), (a,b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))}$ to $\mathcal{A}$. In the real world, the oracle computes $\boldsymbol{r}^g_{s_a s_b} \in \mathbb{F}_2^2$ from two uniform bits (which span $\boldsymbol{R}'^g_{\$} \leftarrow \mathcal{R}_0$ and match $\boldsymbol{R}'^g_{\$,ij}$ for every $i, j \in \mathbb{F}_2$ as per distribution $\mathcal{R}_0$), the truth table $\boldsymbol{t}^g$ (expressed in terms of the truth values x_a, x_b in $\widetilde{x}$ and the masked bits $s_a = \mathsf{lsb}(\boldsymbol{x}_a), s_b = \mathsf{lsb}(\boldsymbol{x}_b)$ in $\widetilde{\boldsymbol{x}}$), and the two masked bits s_a and s_b. More specifically, it follows from $\mathsf{TH.SampleR}$ that
$$\begin{aligned} {\boldsymbol{r}^g_{s_a s_b}}^{\mathsf{T}} &= \boldsymbol{R}'^g_{\$, s_a s_b} \\ &\quad \oplus \Big(g(\overline{p_a}, p_b) \oplus g(\overline{p_a}, \overline{p_b}) \Big) \boldsymbol{R}'_{1, s_a s_b} \\ &\quad \oplus \Big(g(p_a, \overline{p_b}) \oplus g(\overline{p_a}, \overline{p_b}) \Big) \boldsymbol{R}'_{2, s_a s_b} \\ &= \boldsymbol{R}'^g_{\$, s_a s_b} \\ &\quad \oplus \overline{s_a \oplus x_a} \cdot \boldsymbol{R}'_{1, s_a s_b} \oplus \overline{s_b \oplus x_b} \cdot \boldsymbol{R}'_{2, s_a s_b}. \end{aligned} \tag{34}$$
In the ideal world, this $\widetilde{\boldsymbol{r}}$ is also output by SimIn as per the random tape of the oracle.
- Along with $\widetilde{\boldsymbol{x}}$, the oracle sends $\widetilde{T} := \{T_i\}_{i \in \mathcal{W}_\cup(f), n_i(f) \text{ is odd}}$ to $\mathcal{A}$. In the real world, the oracle computes $T_i := \mathsf{Half}_1(\mathsf{H}(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}, i \,\|\, \lfloor n_i(f)/2 \rfloor)) \in \mathbb{F}_{2^{\lambda/2+1}}$, i.e., the *unused* upper half of the $\lfloor n_i(f)/2 \rfloor$-th $\boldsymbol{\Delta}$-related hash output of the wire i. In the ideal world, this $\widetilde{T}$ is sampled by the oracle at random.
- **(Extra queries)** Upon receiving $(\widetilde{\boldsymbol{x}}, \widetilde{x}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{r}}, \widetilde{T})$ from the oracle, $\mathcal{A}$ will also make a forward permutation query $(\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)$ for every $i \in \mathcal{W}_\cup(f)$ and $j \in [0, \lceil n_i(f)/2 \rceil - 1]$, if it has never learned $\pi^*((\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) = \boldsymbol{y}$ for some $\boldsymbol{y}$ in its interaction with $\pi^{*\pm1} \in \{\pi^{\pm1}, \mathsf{SimP}^{\pm1}\}$.
- At the end of the experiment (i.e., once all other transcripts are settled), the oracle sends $\boldsymbol{\Delta}$ to $\mathcal{A}$. In the real world, the oracle gets this $\boldsymbol{\Delta}$ from the output of $\mathsf{TH.Garble}^{\pi^{\pm1}(\cdot)}$. In the ideal world, $\boldsymbol{\Delta}$ is dummy and sampled by the oracle at this time (note that Sim does not use $\boldsymbol{\Delta}$).

According to the two oracle constructions, real-world sample space

$$\Omega_{\mathsf{real}} = \mathbb{F}_2^{\lambda-1} \times \mathbb{F}_2^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda} \times \underbrace{\mathbb{F}_2^{2|f|}}_{\substack{\text{random tape to run}\\ \mathsf{TH.SampleR}\ |f|\ \text{times}}} \times \mathcal{S}_{\lambda+2},$$

and ideal-world sample space

$$\begin{aligned}\Omega_{\mathsf{ideal}} = {} & (\mathbb{F}_2^{3\lambda/2+5})^{|f|} \times \mathbb{F}_2^{\mathsf{rank}(\boldsymbol{E})} \\ & \times \mathbb{F}_2^{(|\mathcal{W}_{\mathsf{in}}(f)|+|\mathcal{Z}(f)|)\lambda-\mathsf{rank}(\boldsymbol{E})} \\ & \times (\mathbb{F}_2^{\lambda/2+3})^{|\mathcal{Z}(f)|} \times (\mathbb{F}_2^{3\lambda/2+3})^{|f|-|\mathcal{Z}(f)|} \\ & \times \underbrace{(\mathbb{F}_2^{\lambda+2})^M}_{\text{for the step 25 of } \mathsf{SimIn} \text{ and } \widetilde{T}} \\ & \times \underbrace{\{0,1\}^*}_{\substack{\text{random tape for}\\ \text{the sampling in } \mathsf{SimP}^{\pm 1}(\cdot)}} \times \underbrace{\mathbb{F}_2^{\lambda-1}}_{\text{dummy } \boldsymbol{\Delta}} .\end{aligned}$$

Given the oracle constructions, a transcript in the original adaptive experiment will be padded with more literal values. Note that transcript padding will not lower the advantage of $\mathcal{A}$ since $\mathcal{A}$ can discard the padding values at will. With the padding, a transcript is of the form:

$$\tau = (\mathcal{K}_1, (f, \widehat{f}), \mathcal{K}_2, (x, (\widetilde{\boldsymbol{x}}, \widetilde{x}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{r}}, \widetilde{T})), \mathcal{K}_3, \boldsymbol{\Delta}),$$

where $\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{K}_3$ are the ordered lists of query-response pairs seen in the interleaved interaction with permutation oracles. We do not explicitly consider query direction in these pairs. Given $\pi^{*\pm 1} \in \{\pi^{\pm 1}, \mathsf{SimP}^{\pm 1}\}$, $\mathcal{A}$ is able to learn $\pi^*(\alpha) = \beta$ if and only if there exists $(\alpha, \beta) \in \cup_{\ell=1}^3 \mathcal{K}_\ell$.

Let $q_\ell := |\mathcal{K}_\ell|$ for every $\ell \in \{1,2,3\}$ and $q_\Sigma := \sum_{\ell=1}^3 q_\ell$. It follows from the extra queries that $q_\Sigma \leq q + N$. Without loss of generality, we can assume that $\mathcal{A}$ only makes *non-repeating* queries, i.e., it never makes forward query α to π^* or backward query β to π^{*-1} for any learned permutation entry (α, β).

Bad transcripts. A transcript $\tau \in \mathcal{T}_{\mathsf{bad}}$ if and only if it incurs at least one of the following events:

- bad_1. There exist distinct $(i,j) \in \mathcal{W}_{\cup}(f) \times [0, \lceil n_i(f)/2\rceil - 1], (i',j') \in \mathcal{W}_{\cup}(f) \times [0, \lceil n_{i'}(f)/2\rceil - 1]$ such that

  $$(\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j) = (\boldsymbol{0} \,\|\, \boldsymbol{x}_{i'}) \oplus (i' \,\|\, j') \tag{35}$$

  $$\vee \quad \begin{aligned} & \boldsymbol{u}_i^j \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \\ & \quad = \boldsymbol{u}_{i'}^{j'} \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_{i'}) \oplus (i' \,\|\, j')) \end{aligned} \tag{36}$$

  $$\vee \quad \begin{aligned} & \boldsymbol{y}_i^j \oplus \boldsymbol{u}_i^j \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \\ & \quad = \boldsymbol{y}_{i'}^{j'} \oplus \boldsymbol{u}_{i'}^{j'} \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_{i'}) \oplus (i' \,\|\, j')) \end{aligned} \tag{37}$$

  where $\boldsymbol{y}_i^j \in \mathbb{F}_{2^{\lambda/2+1}}^2$ will be defined from τ according to (45) and (47).

  In this case, $\mathcal{A}$ is able to check the consistency between the value of $\boldsymbol{y}_i^j \oplus \boldsymbol{y}_{i'}^{j'}$ and that of $\boldsymbol{\Delta}$ at the end of experiment *without* sending any needed queries, which are computed from $\boldsymbol{\Delta}$, to a random permutation or its inverse. In the real world, the consistency certainly holds. However, the ideal-world garbled rows and $\boldsymbol{\Delta}$ are independently sampled, leading to the consistency only with negligible probability. So, $\mathcal{A}$ has non-negligible advantage to distinguish the two worlds and the statistical distance, as an upper bound, also blows up.

  More specifically, the real world is as follows in this case. The pre-image collision (35) leads to the syntactically same XOR of two hash masks in $\boldsymbol{y}_i^j, \boldsymbol{y}_{i'}^{j'}$, which can be XORed to cancel all hash masks to check the consistency with $\boldsymbol{\Delta}$ without further queries. Furthermore, the image collision (36) also implies the pre-image collision (35) as π is permutation. The collision (37) results in the image collision $\pi((\boldsymbol{0} \,\|\, \boldsymbol{x}_i \oplus \boldsymbol{\Delta}) \oplus (i \,\|\, j)) = \pi((\boldsymbol{0} \,\|\, \boldsymbol{x}_{i'} \oplus \boldsymbol{\Delta}) \oplus (i' \,\|\, j'))$ for some distinct (i,j) and (i',j'). Since π is permutation, this collision implies the pre-image collision (35), which can be used to see the consistency. In contrast, the above cancelling of hash masks cannot give this consistency except with negligible probability in the ideal world.
- bad_2. There exists $((\alpha,\beta), i, j) \in \cup_{\ell=1}^3 \mathcal{K}_\ell \times \mathcal{W}_{\cup}(f) \times [0, \lceil n_i(f)/2\rceil - 1]$ such that

  $$\alpha = (\boldsymbol{0} \,\|\, \boldsymbol{x}_i \oplus \boldsymbol{\Delta}) \oplus (i \,\|\, j) \tag{38}$$

  $$\vee \quad \begin{aligned} \beta = {} & \sigma(\boldsymbol{0} \,\|\, \boldsymbol{\Delta}) \oplus \boldsymbol{y}_i^j \\ & \oplus \boldsymbol{u}_i^j \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \end{aligned} \tag{39}$$

  where $\boldsymbol{y}_i^j \in \mathbb{F}_{2^{\lambda/2+1}}^2$ will be defined from τ according to (45) and (47).

  In this case, $\mathcal{A}$ essentially makes it to guess $\boldsymbol{\Delta}$ before receiving this value. It allows $\mathcal{A}$ to distinguish the real world, where every $\boldsymbol{y}_i^j$ is consistent with $\boldsymbol{\Delta}$, and the ideal world with a dummy $\boldsymbol{\Delta}$. So, the statistical distance blows up.
- bad_3. There exists $((\alpha,\beta), i, j) \in \cup_{\ell=1}^2 \mathcal{K}_\ell \times \mathcal{W}_{\cup}(f) \times [0, \lceil n_i(f)/2\rceil - 1]$ such that

  $$\alpha = (\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j) \tag{40}$$

  $$\vee \quad \beta = \boldsymbol{u}_i^j \oplus \sigma((\boldsymbol{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \tag{41}$$

  In this case, $\mathcal{A}$ can make forward/backward queries w.r.t. some active labels before receiving active input labels and computing other active ones.

  This case is necessary to ensure successful programming in the ideal world as the values on the right hand should not be queried before the programming (otherwise it fails due to pre-image or image collision). If the programming fails in the ideal world, the two worlds can be distinguishable as the decoding consistency in the ideal world does not always hold as in the real world.

Bounding $1 - \varepsilon_2$. Without loss of generality, we can consider some fixed good transcript τ such that $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ (if this probability is zero, it is trivial by definition that $\varepsilon_2 = 0$ for this τ). Using Lemma 1, we turn to analyze the sampled oracle's compatibility (Section 3.4) with such a transcript, instead of the interaction between $\mathcal{A}$ and the sampled oracle.

Note that there is a computationally unbounded non-uniform adversary $\mathcal{A}'$ such that, for every oracle ω, it sends the queries in τ in order in its interaction with ω (e.g., $\mathcal{A}'$ has auxiliary input τ and sends its ordered queries). Fix $\mathcal{A}'$ in the following compatibility analysis so that any real-world or ideal-world oracle will receive the queries in τ in order. For a response c recorded in a fixed τ, let $\omega \vdash c$ denote the event that, fixing the ordered queries as per τ, oracle ω produces c given the corresponding query. Let $\mathcal{K}^{\mathsf{R}}$ denote the order-preserving list of the responses in an ordered list $\mathcal{K}$ of permutation query-response pairs.

First, we compute $\Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)]$. Following from three-halves, a real-world oracle $\omega = (\boldsymbol{\varDelta}, \{(p_i, \boldsymbol{w}_i)\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, \{(r^g_L, r^g_R)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)}, \pi) \in \varOmega_{\mathsf{real}}$. So,

$$\begin{aligned}&\Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] \\ &\quad = \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \vdash (\mathcal{K}^{\mathsf{R}}_1, \widehat{f}, \mathcal{K}^{\mathsf{R}}_2, \widetilde{\boldsymbol{x}}, \widetilde{x}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{r}}, \widetilde{T}, \mathcal{K}^{\mathsf{R}}_3, \boldsymbol{\varDelta})] .\end{aligned}$$

To begin with, every real-world ω certainly produces f' (i.e., the first value in $\widehat{f}$) and $\widetilde{x}$ fixed in τ, which leads to $\Pr_{\omega \leftarrow \varOmega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$. This non-zero probability implies that $(f', \widetilde{x})$ in τ is honestly and deterministically computed from the fixed queries (f, x). Otherwise, no ideal-world oracle, which computes $(f', \widetilde{x})$ from the same deterministic procedure, can produce this transcript, contradicting the non-zero probability. As every real-world ω will follow the same deterministic procedure, it certainly produces the two values.

Meanwhile, a real-world oracle ω should have the same literal value of $\boldsymbol{\varDelta}$ as its counterpart in τ. Then, the real-world sampling $\boldsymbol{R}'^g_{\$} \leftarrow \mathcal{R}_0$ (in Appendix B.4) and $\boldsymbol{r}^g = \begin{bmatrix}\boldsymbol{r}^g_{00} & \boldsymbol{r}^g_{01} & \boldsymbol{r}^g_{10} & \boldsymbol{r}^g_{11}\end{bmatrix}^{\mathsf{T}} \in (\mathbb{F}_2^2)^4$ ensure that, for every $i, j \in \mathbb{F}_2$ and every $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))$,

$$\begin{aligned}&\boldsymbol{r}^{g\ \mathsf{T}}_{ij} = \begin{bmatrix}r^g_{ijL} & r^g_{ijR}\end{bmatrix} \\ &= \boldsymbol{R}'^g_{\$,ij} \oplus \overline{s_a \oplus x_a} \cdot \boldsymbol{R}'_{1,ij} \oplus \overline{s_b \oplus x_b} \cdot \boldsymbol{R}'_{2,ij} \\ &= \begin{bmatrix}r^g_L & r^g_R\end{bmatrix} \oplus \overline{s_a \oplus x_a} \cdot \boldsymbol{R}'_{1,ij} \oplus \overline{s_b \oplus x_b} \cdot \boldsymbol{R}'_{2,ij} .\end{aligned} \tag{42}$$

In particular, (42) holds for $i = s_a$ and $j = s_b$ (i.e., the real-world compatibility between ω and $\widetilde{\boldsymbol{r}}$ in (34) for this g) with probability 2^{-2}, which comes from two uniform coins $(r^g_L, r^g_R) \in \mathbb{F}_2^2$ in ω and independent of the literal values of x_a, x_b, s_a, and s_b. Conditioned on the compatibility so far, the probability

$$\begin{aligned}&\Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] \\ &= \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}^{\mathsf{R}}_1, F, d, \mathcal{K}^{\mathsf{R}}_2, \widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}, \widetilde{T}, \mathcal{K}^{\mathsf{R}}_3) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash (\widetilde{\boldsymbol{r}}, \boldsymbol{\varDelta})\right] \\ &\quad \cdot \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \vdash (f', \widetilde{x}) \wedge \omega \vdash (\widetilde{\boldsymbol{r}}, \boldsymbol{\varDelta})] \\ &= \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}^{\mathsf{R}}_1, F, d, \mathcal{K}^{\mathsf{R}}_2, \widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}, \widetilde{T}, \mathcal{K}^{\mathsf{R}}_3) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash (\widetilde{\boldsymbol{r}}, \boldsymbol{\varDelta})\right] \\ &\quad \cdot \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \vdash (f', \widetilde{x})] \cdot \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \vdash \widetilde{\boldsymbol{r}}] \cdot \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}[\omega \vdash \boldsymbol{\varDelta}] \\ &= \Pr_{\omega \leftarrow \varOmega_{\mathsf{real}}}\left[\omega \vdash (\mathcal{K}^{\mathsf{R}}_1, F, d, \mathcal{K}^{\mathsf{R}}_2, \widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}, \widetilde{T}, \mathcal{K}^{\mathsf{R}}_3) \;\middle|\; \omega \vdash (f', \widetilde{x}) \wedge \omega \vdash (\widetilde{\boldsymbol{r}}, \boldsymbol{\varDelta})\right] \\ &\quad \cdot \frac{1}{2^{2|f| + (\lambda - 1)}} .\end{aligned}$$

Conditioned on the compatibility with $(f', \widetilde{x}, \widetilde{\boldsymbol{r}}, \boldsymbol{\varDelta})$, a real-world ω should be compatible with $(\cup_{\ell=1}^3 \mathcal{K}^{\mathsf{R}}_\ell, F, \widetilde{\boldsymbol{u}}, \widetilde{T})$ and some active labels in $\widetilde{\boldsymbol{x}}$ such that

(i) $\pi^{\pm 1}$ maps the fixed permutation queries to the responses in $\cup_{\ell=1}^3 \mathcal{K}^{\mathsf{R}}_\ell$.

(ii) For each $i \in \mathcal{W}_{\mathsf{in}}(f)$, it holds that $\boldsymbol{x}_i = \boldsymbol{w}_i \oplus (p_i \oplus x_i)\boldsymbol{\varDelta}$.

(iii) For each $i \in \mathcal{W}_{\cup}(f)$ and $j \in [0, \lceil n_i(f)/2 \rceil - 1]$, it holds that

$$\begin{aligned}\mathsf{H}(\boldsymbol{x}_i, i \mathbin\| j) &:= \pi((\boldsymbol{0} \mathbin\| \boldsymbol{x}_i) \oplus (i \mathbin\| j)) \\ &\quad \oplus \sigma((\boldsymbol{0} \mathbin\| \boldsymbol{x}_i) \oplus (i \mathbin\| j)) \\ &= \boldsymbol{u}^j_i = \begin{bmatrix}\mathsf{Half}_1(\boldsymbol{u}^j_i) \\ \mathsf{Half}_0(\boldsymbol{u}^j_i)\end{bmatrix} .\end{aligned} \tag{43}$$

(iv) For each $g \in \mathcal{G}_{\mathsf{and}}(f)$ with $(a, b, \varGamma, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{in}_2(g), \mathsf{out}(g))$ and the non-repeating counters $(\chi_a \mathbin\| \rho_a)$, $(\chi_b \mathbin\| \rho_b)$, and $(\chi_\varGamma \mathbin\| \rho_\varGamma)$ as per some fixed topology order of f, it holds that

$$\begin{aligned}&\left(\boldsymbol{r}^g_{s_a s_b} \;\middle\|\; \boldsymbol{x}_c \oplus \boldsymbol{R}^g_{s_a s_b} \begin{bmatrix}\boldsymbol{x}_a \\ \boldsymbol{x}_b\end{bmatrix}\right) \\ &\quad = \begin{bmatrix}1 & 0 & 1 \\ 0 & 1 & 1\end{bmatrix} \begin{bmatrix}\mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a, a \mathbin\| \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b, b \mathbin\| \chi_b)) \\ \mathsf{Half}_{\rho_\varGamma}(\mathsf{H}(\boldsymbol{x}_\varGamma, \varGamma \mathbin\| \chi_\varGamma))\end{bmatrix} \\ &\qquad \oplus \boldsymbol{V}_{s_a s_b} \left(\boldsymbol{z}^g \;\middle\|\; \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{g}^g\end{bmatrix}\right) ,\end{aligned} \tag{44}$$

$$\begin{aligned}&\left(\boldsymbol{z}^g_{\mathsf{bot}} \;\middle\|\; \boldsymbol{g}^g\right) \\ &= \boldsymbol{V}^+_{\mathsf{bot}} \left(\boldsymbol{M}\boldsymbol{h}^g \oplus \left(\boldsymbol{r}^g \;\middle\|\; (\widehat{\boldsymbol{R}}^g \oplus (\begin{bmatrix}\boldsymbol{0}_{4\times 2} & \boldsymbol{t}^g\end{bmatrix} \otimes \boldsymbol{I}_2)) \begin{bmatrix}\boldsymbol{w}_a \\ \boldsymbol{w}_b \\ \boldsymbol{\varDelta}\end{bmatrix}\right)\right) \\ &= \boldsymbol{V}^+_{\mathsf{bot}}\boldsymbol{M} \begin{bmatrix}\overline{s_a} & s_a & & & & \\ s_a & \overline{s_a} & & & & \\ & & \overline{s_b} & s_b & & \\ & & s_b & \overline{s_b} & & \\ & & & & \overline{s_a \oplus s_b} & s_a \oplus s_b \\ & & & & s_a \oplus s_b & \overline{s_a \oplus s_b}\end{bmatrix} \begin{bmatrix}\mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a, a \mathbin\| \chi_a)) \\ \mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a \oplus \boldsymbol{\varDelta}, a \mathbin\| \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b, b \mathbin\| \chi_b)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b \oplus \boldsymbol{\varDelta}, b \mathbin\| \chi_b)) \\ \mathsf{Half}_{\rho_\varGamma}(\mathsf{H}(\boldsymbol{x}_\varGamma, \varGamma \mathbin\| \chi_\varGamma)) \\ \mathsf{Half}_{\rho_\varGamma}(\mathsf{H}(\boldsymbol{x}_\varGamma \oplus \boldsymbol{\varDelta}, \varGamma \mathbin\| \chi_\varGamma))\end{bmatrix} \\ &\quad \underbrace{\oplus \left(\begin{bmatrix}\overline{s_a \oplus x_a} \\ \overline{s_b \oplus x_b} \\ s_a \oplus x_a \oplus s_b \oplus x_b\end{bmatrix} \;\middle\|\; \boldsymbol{V}^+_{\mathsf{bot}} (\widehat{\boldsymbol{R}}^g \oplus (\begin{bmatrix}\boldsymbol{0}_{4\times 2} & \boldsymbol{t}^g\end{bmatrix} \otimes \boldsymbol{I}_2)) \begin{bmatrix}\boldsymbol{x}_a \oplus s_a\boldsymbol{\varDelta} \\ \boldsymbol{x}_b \oplus s_b\boldsymbol{\varDelta} \\ \boldsymbol{\varDelta}\end{bmatrix}\right)}_{\begin{bmatrix}\mathsf{Half}_{\rho_a}(\boldsymbol{e}^{\chi_a}_a) & \mathsf{Half}_{\rho_b}(\boldsymbol{e}^{\chi_b}_b) & \mathsf{Half}_{\rho_\varGamma}(\boldsymbol{e}^{\chi_\varGamma}_\varGamma)\end{bmatrix}^{\mathsf{T}} \in \mathbb{F}^3_{2^{\lambda/2+1}}} \\ &\Leftrightarrow \underbrace{\left(\boldsymbol{z}^g_{\mathsf{bot}} \;\middle\|\; \boldsymbol{g}^g\right) \oplus \begin{bmatrix}\mathsf{Half}_{\rho_a}(\boldsymbol{e}^{\chi_a}_a) & \mathsf{Half}_{\rho_b}(\boldsymbol{e}^{\chi_b}_b) & \mathsf{Half}_{\rho_\varGamma}(\boldsymbol{e}^{\chi_\varGamma}_\varGamma)\end{bmatrix}^{\mathsf{T}}}_{\begin{bmatrix}\mathsf{Half}_{\rho_a}(\boldsymbol{y}^{\chi_a}_a) & \mathsf{Half}_{\rho_b}(\boldsymbol{y}^{\chi_b}_b) & \mathsf{Half}_{\rho_\varGamma}(\boldsymbol{y}^{\chi_\varGamma}_\varGamma)\end{bmatrix}^{\mathsf{T}} \in \mathbb{F}^3_{2^{\lambda/2+1}}} \\ &= \begin{bmatrix}\mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a, a \mathbin\| \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b, b \mathbin\| \chi_b)) \\ \mathsf{Half}_{\rho_\varGamma}(\mathsf{H}(\boldsymbol{x}_\varGamma, \varGamma \mathbin\| \chi_\varGamma))\end{bmatrix} \oplus \begin{bmatrix}\mathsf{Half}_{\rho_a}(\mathsf{H}(\boldsymbol{x}_a \oplus \boldsymbol{\varDelta}, a \mathbin\| \chi_a)) \\ \mathsf{Half}_{\rho_b}(\mathsf{H}(\boldsymbol{x}_b \oplus \boldsymbol{\varDelta}, b \mathbin\| \chi_b)) \\ \mathsf{Half}_{\rho_\varGamma}(\mathsf{H}(\boldsymbol{x}_\varGamma \oplus \boldsymbol{\varDelta}, \varGamma \mathbin\| \chi_\varGamma))\end{bmatrix} ,\end{aligned} \tag{45}$$

where the bits $x_a, x_b, s_a = \mathsf{lsb}(\boldsymbol{x}_a), s_b = \mathsf{lsb}(\boldsymbol{x}_b)$ are given in τ, and in (45),

$$\begin{aligned}\boldsymbol{V}^+ &= \begin{bmatrix}\boldsymbol{V}^+_{\mathsf{top}} & \boldsymbol{V}^+_{\mathsf{bot}}\end{bmatrix}^{\mathsf{T}} \in (\mathbb{F}_2^{1\times 8})^2 \times (\mathbb{F}_2^{1\times 8})^3, \\ \forall g \in \mathcal{G}_{\mathsf{and}}(f) : \boldsymbol{z}^g &= \begin{bmatrix}\boldsymbol{z}^g_{\mathsf{top}} & \boldsymbol{z}^g_{\mathsf{bot}}\end{bmatrix}^{\mathsf{T}} \in \mathbb{F}_2^2 \times \mathbb{F}_2^3, \\ \forall g \in \mathcal{G}_{\mathsf{and}}(f) : \boldsymbol{t}^g &= \begin{bmatrix}g(s_a \oplus x_a, s_b \oplus x_b) \\ g(s_a \oplus x_a, \overline{s_b \oplus x_b}) \\ g(\overline{s_a \oplus x_a}, s_b \oplus x_b) \\ g(\overline{s_a \oplus x_a}, \overline{s_b \oplus x_b})\end{bmatrix} ,\end{aligned}$$

and for each $g \in \mathcal{G}_{\mathsf{and}}(f)$, $\widehat{\boldsymbol{R}}^g \in \mathbb{F}_2^{8\times 6}$ is fixed by running $\mathsf{TH.SampleR}$ with random coins (r_L^g, r_R^g) in ω, which have been compatible (according to the condition) with $\widetilde{\boldsymbol{r}}$ as per the (42) for this g and $(i,j) = (s_a, s_b)$.

(v) For each $i \in \{i \in \mathcal{W}_{\cup}(f) \mid n_i(f) \text{ is odd}\}$, it holds that

$$\mathsf{Half}_1(\mathsf{H}(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}, i \,\|\, \lfloor n_i(f)/2 \rfloor)) = T_i. \tag{46}$$

Conditioned on the compatibility so far, every real-world oracle ω is always compatible with the leftover values in τ, i.e., decoding table d and other active labels in $\widetilde{\boldsymbol{x}}$, which are deterministically computed from XOR combination. The reason is that, for τ ensuring $\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$, these values should be honestly determined by the conditioned values as in the real world. Otherwise, this probability will be zero for an ideal-world oracle, which obtains them from a consistent deterministic computation as per the conditioned values. As every real-world oracle ω honestly follows the real-world computation, this "leftover" compatibility must hold conditioned on the previous compatibility.

It remains to compute the conditional probabilities for (i) to (v). Consider (iii), (iv), and (v). We claim that every good τ with $\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ already implies (43) and (44). To see this, we use that condition $\neg\mathsf{bad}_3$ for good transcripts and the extra queries ensure that $\mathcal{K}_3$ fixes the pairs of permutation pre-images and images for hash values

$$\{\mathsf{H}(\boldsymbol{x}_i, i \,\|\, j)\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]}.$$

These values are consistent with $\widetilde{\boldsymbol{u}}$ fixed in τ as per (43). Otherwise, τ cannot satisfy $\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ since $\neg\mathsf{bad}_1 \wedge \neg\mathsf{bad}_3$ for every good transcript implies successful programming in the ideal world so that $\mathsf{H}(\boldsymbol{x}_i, i \,\|\, j) = \boldsymbol{u}_i^j$. As a corollary, (44) holds for every good transcript due to this consistency and the step 15 and 18 of SimIn.

Consider (45) and (46), the leftover parts of (iii), (iv), and (v). If we define from the fixed τ that

$$\mathsf{Half}_1(\boldsymbol{y}_i^{\lfloor n_i(f)/2\rfloor}) := \mathsf{Half}_1(\boldsymbol{u}_i^{\lfloor n_i(f)/2\rfloor}) \oplus T_i \in \mathbb{F}_{2^{\lambda/2+1}} \tag{47}$$

for each $i \in \{i \in \mathcal{W}_{\cup}(f) \mid n_i(f) \text{ is odd}\}$, then we can unify (45) and (46) as:

$$\mathcal{V} := \left\{ \begin{aligned} & i \in \mathcal{W}_{\cup}(f), j \in [0, \lceil n_i(f)/2\rceil - 1] : \\ & \qquad \boldsymbol{y}_i^j = \mathsf{H}(\boldsymbol{x}_i, i \,\|\, j) \oplus \mathsf{H}(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}, i \,\|\, j) \\ & \qquad\quad = \underbrace{\pi((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j))}_{P_{i \,\|\, j,0}} \\ & \qquad\qquad \oplus \underbrace{\pi((\mathbf{0} \,\|\, \boldsymbol{x}_i \oplus \boldsymbol{\Delta}) \oplus (i \,\|\, j))}_{P_{i \,\|\, j,1}} \\ & \qquad\qquad \oplus \sigma(\mathbf{0} \,\|\, \boldsymbol{\Delta}) \end{aligned} \right\}$$

since (45) and (46) iterate through both halves of all well-defined $\boldsymbol{y}_i^j \in \mathbb{F}^2_{2^{\lambda/2+1}}$ for $i \in \mathcal{W}_{\cup}(f)$ and $j \in [0, \lceil n_i(f)/2\rceil - 1]$. Since τ is a good transcript, there are exactly $2N$ pairwise distinct permutation pre-images on the right hand (otherwise, there will be a pair of permutation pre-images leading to (35) in bad_1 or a permutation pre-image leading to (38) in bad_2 given the extra queries). Due to this pairwise distinctness, $\mathcal{V}$ includes exact $2N$ syntactically different variables $\mathcal{P} := \{P_{i\,\|\,j,0}, P_{i\,\|\,j,1}\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]}$. They can fix the same number of the entries of permutation π in a real-world ω *if and only if their literal values fixed by τ are also pairwise distinct*. We note that every good transcript τ does fix *exact one* such assignment of these variables for the following reasons:

- (43) already holds for τ, i.e., for $i \in \mathcal{W}_{\cup}(f)$ and $j \in [0, \lceil n_i(f)/2\rceil - 1]$,

$$\begin{aligned} & P_{i \,\|\, j,0} \\ & := \pi((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \\ & = \boldsymbol{u}_i^j \oplus \sigma((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)). \end{aligned} \tag{48}$$

  The literal values of $\{P_{i\,\|\,j,0}\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]}$ are immediate from the responses in $\mathcal{K}_3^{\mathsf{R}}$ given the extra queries and will be pairwise distinct due to the impossible (36) from $\neg\mathsf{bad}_1$.
- For $i \in \mathcal{W}_{\cup}(f)$ and $j \in [0, \lceil n_i(f)/2\rceil - 1]$, the literal value of $P_{i\,\|\,j,1}$ is fixed by τ according to $\mathcal{V}$ and (48):

$$\begin{aligned} & P_{i \,\|\, j,1} \\ & := \pi((\mathbf{0} \,\|\, \boldsymbol{x}_i \oplus \boldsymbol{\Delta}) \oplus (i \,\|\, j)) \\ & = \sigma(\mathbf{0} \,\|\, \boldsymbol{\Delta}) \oplus \boldsymbol{y}_i^j \oplus \boldsymbol{u}_i^j \oplus \sigma((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)). \end{aligned} \tag{49}$$

  The literal values of $\{P_{i\,\|\,j,1}\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]}$ will be pairwise distinct according to the impossible (37) from $\neg\mathsf{bad}_1$.
- The goodness of τ also ensures that there do not exist

$$\begin{aligned} P' &\in \{P_{i\,\|\,j,0}\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]}, \\ P'_{\boldsymbol{\Delta}} &\in \{P_{i\,\|\,j,1}\}_{i\in\mathcal{W}_{\cup}(f), j\in[0,\lceil n_i(f)/2\rceil-1]} \end{aligned}$$

  such that $P' = P'_{\boldsymbol{\Delta}}$. Otherwise, this equality and (49) ensure that there exist $(i,j) \in \mathcal{W}_{\cup}(f) \times [0, \lceil n_i(f)/2\rceil - 1]$ and $(i',j') \in \mathcal{W}_{\cup}(f) \times [0, \lceil n_{i'}(f)/2\rceil - 1]$ such that

$$\begin{aligned} & \pi((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)) \\ & = \pi((\mathbf{0} \,\|\, \boldsymbol{x}_{i'} \oplus \boldsymbol{\Delta}) \oplus (i' \,\|\, j')) \\ & = \sigma(\mathbf{0} \,\|\, \boldsymbol{\Delta}) \oplus \boldsymbol{y}_{i'}^{j'} \oplus \boldsymbol{u}_{i'}^{j'} \oplus \sigma((\mathbf{0} \,\|\, \boldsymbol{x}_{i'}) \oplus (i' \,\|\, j')). \end{aligned} \tag{50}$$

  We have $((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j), \pi((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j))) \in \mathcal{K}_3$ according to $\neg\mathsf{bad}_3$ and the extra queries. So, (50) contradicts the impossible (39) from $\neg\mathsf{bad}_2$.

Putting these cases together, we can see that τ yields a value assignment of $\mathcal{P}$, and this assignment fixes exact $2N$ entries of real-world permutation π.

Based on the condition $\neg\mathsf{bad}_1 \wedge \neg\mathsf{bad}_3$ of good transcript τ, N extra queries are non-repeating and the number of non-repeating queries is $q + N = q_\Sigma$. As a result, N responses in $\cup_{\ell=1}^3 \mathcal{K}_\ell^{\mathsf{R}}$ for the non-repeating extra queries are fixed by the values in $\mathcal{P}$ while the other $q_\Sigma - N = q$ responses are fixed by real-world π (conditioned on the values in $\mathcal{P}$). Based on (i), (ii), (iii), (iv), and (v) together with the "leftover" compatibility, we have in the real world that

$$\begin{aligned}
&\Pr_{\omega\leftarrow\Omega_{\mathsf{real}}}\left[\omega\vdash(\mathcal{K}_1^{\mathsf{R}},F,d,\mathcal{K}_2^{\mathsf{R}},\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{u}},\widetilde{T},\mathcal{K}_3^{\mathsf{R}})\;\middle|\;\omega\vdash(f',\widetilde{x})\wedge\omega\vdash(\widetilde{\boldsymbol{r}},\boldsymbol{\Delta})\right]\\
&\quad=\frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}}\cdot\frac{(2^{\lambda+2}-2N-(q_{\Sigma}-N))!}{(2^{\lambda+2})!}\\
&\quad=\frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}}\cdot\frac{1}{(2^{\lambda+2})_{q+2N}}\\
&\Rightarrow\Pr_{\omega\leftarrow\Omega_{\mathsf{real}}}[\omega\in\mathsf{comp}_{\mathsf{real}}(\tau)]\\
&\quad=\frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}}\cdot\frac{1}{(2^{\lambda+2})_{q+2N}}\cdot\frac{1}{2^{2|f|+(\lambda-1)}}.
\end{aligned}$$

Second, in the ideal world, we can use a similar argument to show

$$\begin{aligned}
&\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[\omega\in\mathsf{comp}_{\mathsf{ideal}}(\tau)]\\
&=\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\omega\vdash(\mathcal{K}_1^{\mathsf{R}},\mathcal{K}_2^{\mathsf{R}},\mathcal{K}_3^{\mathsf{R}})\;\middle|\;\omega\vdash(\widehat{f},\widetilde{\boldsymbol{x}},\widetilde{x},\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{r}},\widetilde{T},\boldsymbol{\Delta})\right]\\
&\qquad\cdot\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[\omega\vdash(\widehat{f},\widetilde{\boldsymbol{x}},\widetilde{x},\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{r}},\widetilde{T},\boldsymbol{\Delta})]\\
&=\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\omega\vdash(\mathcal{K}_1^{\mathsf{R}},\mathcal{K}_2^{\mathsf{R}},\mathcal{K}_3^{\mathsf{R}})\;\middle|\;\omega\vdash(\widehat{f},\widetilde{\boldsymbol{x}},\widetilde{x},\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{r}},\widetilde{T},\boldsymbol{\Delta})\right]\\
&\qquad\cdot\frac{1}{2^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda+(3\lambda+8)|f|+(\lambda+2)M+(\lambda-1)}}.
\end{aligned}$$

According to the condition $\neg\mathsf{bad}_1\wedge\neg\mathsf{bad}_3$ and the N extra queries, there are exact $q+N=q_{\Sigma}$ non-repeating queries. Here, N responses in $\cup_{\ell=1}^{3}\mathcal{K}_{\ell}^{\mathsf{R}}$ for the non-repeating extra queries are fixed by the conditioned values while the other responses are fixed by $\mathsf{SimP}^{\pm1}$ for other $q_{\Sigma}-N=q$ queries.

Let $\mathcal{Q}_{i-1}$ denote the list $\mathcal{Q}$ (which is maintained in internal state $\mathsf{st}_{\mathsf{sim}}$) when it includes $i-1\in[0,q_{\Sigma}-1]$ pairs (note that $\mathcal{Q}$ finally includes q_{Σ} pairs given the q_{Σ} non-repeating queries), and $\mathcal{N}\subseteq[1,q_{\Sigma}]$ denote the index set of these q queries in q_{Σ} non-repeating queries to $\mathsf{SimP}^{\pm1}(\cdot)$ such that $|\mathcal{N}|=q$. We have

$$\begin{aligned}
&\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}\left[\omega\vdash(\mathcal{K}_1^{\mathsf{R}},\mathcal{K}_2^{\mathsf{R}},\mathcal{K}_3^{\mathsf{R}})\;\middle|\;\omega\vdash(\widehat{f},\widetilde{\boldsymbol{x}},\widetilde{x},\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{r}},\widetilde{T},\boldsymbol{\Delta})\right]\\
&\quad=\prod_{i\in\mathcal{N}}\frac{1}{2^{\lambda+2}-|\mathcal{Q}_{i-1}|}=\prod_{i\in\mathcal{N}}\frac{1}{2^{\lambda+2}-(i-1)}\\
&\quad\le\frac{1}{2^{\lambda+2}-N}\times\cdots\times\frac{1}{2^{\lambda+2}-(q_{\Sigma}-1)}\\
&\quad=\frac{1}{(2^{\lambda+2}-N)_q},\\
&\Rightarrow\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[\omega\in\mathsf{comp}_{\mathsf{ideal}}(\tau)]\\
&\quad\le\frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}}\cdot\frac{1}{(2^{\lambda+2}-N)_q\cdot(2^{\lambda+2})^{3|f|+M}}\\
&\qquad\cdot\frac{1}{2^{2|f|+(\lambda-1)}}.
\end{aligned}$$

As iterating an AND gate increases three counters in either world, we have

$$\begin{aligned}
&3\,|f|=\sum_{i\in\mathcal{W}_{\cup}(f)}n_i(f)\\
&=\sum_{\substack{i\in\mathcal{W}_{\cup}(f),\\ n_i(f)\text{ is even}}}2\cdot\left\lceil\frac{n_i(f)}{2}\right\rceil+\sum_{\substack{i\in\mathcal{W}_{\cup}(f),\\ n_i(f)\text{ is odd}}}2\cdot\left(\left\lceil\frac{n_i(f)}{2}\right\rceil-\frac{1}{2}\right)\qquad(51)\\
&=2N-M.
\end{aligned}$$

So, we can have $\varepsilon_2=0$ since, for every $N\ge0$ and every $q\ge0$,

$$\begin{aligned}
&\frac{\Pr_{\omega\leftarrow\Omega_{\mathsf{real}}}[\omega\in\mathsf{comp}_{\mathsf{real}}(\tau)]}{\Pr_{\omega\leftarrow\Omega_{\mathsf{ideal}}}[\omega\in\mathsf{comp}_{\mathsf{ideal}}(\tau)]}\\
&\ge\frac{(2^{\lambda+2}-N)_q\cdot(2^{\lambda+2})^{2N}}{(2^{\lambda+2})_{q+2N}}\\
&\ge\frac{(2^{\lambda+2}-N)_q\cdot(2^{\lambda+2})_N\cdot(2^{\lambda+2})^{N}}{(2^{\lambda+2})_{q+2N}}\\
&=\frac{(2^{\lambda+2})_{q+N}\cdot(2^{\lambda+2})^{N}}{(2^{\lambda+2})_{q+2N}}=\frac{(2^{\lambda+2})^{N}}{(2^{\lambda+2}-(q+N))_N}\ge1.
\end{aligned}$$

Bounding ε_1. We bound the probabilities of the bad events in the *ideal world*. First, consider bad_1. Note that each active label $\boldsymbol{x}_i$ can be written as the XOR of (i) some active circuit input labels, and/or (ii) some active output labels of the *precedent* AND gates, i.e., for $\mathcal{I}_i\ominus\mathcal{J}_i\neq\varnothing$,

$$\begin{aligned}
\boldsymbol{x}_i&=\left(\bigoplus\nolimits_{w\in\mathcal{I}_i\subseteq\mathcal{W}_{\mathsf{in}}(f)}\boldsymbol{x}_w\right)\oplus\left(\bigoplus\nolimits_{w\in\mathcal{J}_i\subseteq\mathcal{W}_{\mathsf{and}}(f)}\boldsymbol{x}_w\right)\\
&=\bigoplus\nolimits_{w\in\mathcal{I}_i\ominus\mathcal{J}_i}\boldsymbol{x}_w\in\mathbb{F}^2_{2^{\lambda/2}}.
\end{aligned}\qquad(52)$$

For (35), we can use (52) to rewrite it as

$$\begin{aligned}
&\left(\boldsymbol{0}_{2\times1}\;\middle\|\;\left(\bigoplus\nolimits_{w\in(\mathcal{I}_i\ominus\mathcal{I}_{i'})\ominus(\mathcal{J}_i\ominus\mathcal{J}_{i'})}\boldsymbol{x}_w\right)\right)\\
&\quad=(i\,\|\,j)\oplus(i'\,\|\,j')\in\mathbb{F}^2_{2^{\lambda/2+1}}.
\end{aligned}$$

According to SimIn, each $\boldsymbol{x}_w$, which is sampled in the step 1 or computed in the step 19, has at least $\lambda-1$ random non-LSBs. Therefore, the equality holds with probability at most $2^{-(\lambda-1)}$ for some fixed distinct (i,j) and (i',j'), and, if (i) $\mathcal{I}_i=\mathcal{I}_{i'}$ and $\mathcal{J}_i=\mathcal{J}_{i'}$ or (ii) the right-hand XOR does not give two leading zero bits, this probability is zero for the distinct (i,j) and (i',j'). For (36), the worst case is that both halves of $\boldsymbol{u}_i^j\oplus\boldsymbol{u}_{i'}^{j'}\in\mathbb{F}^2_{2^{\lambda/2+1}}$ respectively serve (in the step 14 of SimIn) as the lower-half masks of two AND gates both with output wires in $\mathcal{Z}(f)$. In this case, each half of this XOR will have at least $1+(\lambda/2-1)=\lambda/2$ random non-LSBs using the lower-half randomness of $(\boldsymbol{r}^g_{s_as_b}\,\|\,\boldsymbol{x}_c)\in\mathbb{F}^2_{2^{\lambda/2+1}}$ in each of the two AND gates. Note that such $2\cdot\lambda/2=\lambda$ bits are independent of other (previously fixed) active labels, including $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$. Thus, the equality holds with probability at most $2^{-\lambda}<2^{-(\lambda-1)}$ for some fixed (i,j) and (i',j').

Similarly, the worst case for (37) is that both halves of $\boldsymbol{u}_i^j\oplus\boldsymbol{u}_{i'}^{j'}\in\mathbb{F}^2_{2^{\lambda/2+1}}$ are respectively the lower-half masks of two AND gates both with output wires in $\mathcal{Z}(f)$. Otherwise, at least one half of $(\boldsymbol{y}_i^j\oplus\boldsymbol{u}_i^j)\oplus(\boldsymbol{y}_{i'}^{j'}\oplus\boldsymbol{u}_{i'}^{j'})$ is uniform for:

(i) The $\boldsymbol{u}_i^j\oplus\boldsymbol{u}_{i'}^{j'}$ masked with the uniform upper half of some $(\boldsymbol{r}^g_{s_as_b}\,\|\,\boldsymbol{x}_c)$, which cannot be cancelled by $\boldsymbol{y}_i^j$ or $\boldsymbol{y}_{i'}^{j'}$ defined as per (45) and (47), or

(ii) The uniform $\boldsymbol{u}_i^j$ or $\boldsymbol{u}_{i'}^{j'}$ sampled in the step 16 of SimIn and independent of $\boldsymbol{y}_i^j$ or $\boldsymbol{y}_{i'}^{j'}$, or

(iii) The uniform $T_i=\mathsf{Half}_1(\boldsymbol{y}_i^j\oplus\boldsymbol{u}_i^j)$ (resp., $T_{i'}=\mathsf{Half}_1(\boldsymbol{y}_{i'}^{j'}\oplus\boldsymbol{u}_{i'}^{j'})$) when the upper half of the XOR is considered, $n_i(f)$ (resp., $n_{i'}(f)$) is odd, and $j=\lceil n_i(f)/2\rceil-1$ (resp., $j'=\lceil n_{i'}(f)/2\rceil-1$).

In this worst case, each half of $\boldsymbol{u}_i^j \oplus \boldsymbol{u}_{i'}^{j'}$ includes at least $1 + (\lambda/2 - 1) = \lambda/2$ uniform bits for the lower-half non-LSBs of some $(\boldsymbol{r}^g_{s_a s_b} \| \boldsymbol{x}_c)$. These non-LSBs are independent of $\boldsymbol{y}_i^j$ and $\boldsymbol{y}_{i'}^{j'}$ defined as per (45) and (47), or other (previously fixed) active labels, including $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$. Thus, (37) holds with probability $2^{-\lambda}$ for some fixed distinct (i, j) and (i', j').

Taking a union bound over the above cases, we have

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1] \leq \frac{5N(N-1)}{2^{\lambda+1}}. \tag{53}$$

Then, consider bad_2. For (38), it is clear that $(\boldsymbol{0} \| \boldsymbol{\Delta}) = \alpha \oplus (\boldsymbol{0} \| \boldsymbol{x}_i) \oplus (i \| j)$, which occurs with probability $2^{-(\lambda-1)}$ according to the randomness of $\boldsymbol{\Delta}$. Let $\boldsymbol{\Delta} = \begin{bmatrix} \Delta_L & \Delta_R \end{bmatrix}^\top \in \mathbb{F}^2_{2^{\lambda/2}}$. We note that each $\boldsymbol{y}_i^j \in \mathbb{F}^2_{2^{\lambda/2+1}}$ defined from (45) and (47) includes an additive term of the form

$$L_{\xi_1,\xi_2,\xi_3,\xi_4}(\boldsymbol{0} \| \boldsymbol{\Delta}) = \begin{bmatrix} 0 \| \xi_1 \Delta_L \oplus \xi_2 \Delta_R \\ 0 \| \xi_3 \Delta_L \oplus \xi_4 \Delta_R \end{bmatrix} \in \mathbb{F}^2_{2^{\lambda/2+1}}$$

for some $\xi_1, \xi_2, \xi_3, \xi_4 \in \mathbb{F}_2$, while other additive terms in $\boldsymbol{y}_i^j$ are independent of $\boldsymbol{\Delta}$. Linear orthomorphism σ for every $L_{\xi_1,\xi_2,\xi_3,\xi_4}$ turns (39) into

$$\begin{aligned} &\sigma(\boldsymbol{0} \| \boldsymbol{\Delta}) \oplus L_{\xi_1,\xi_2,\xi_3,\xi_4}(\boldsymbol{0} \| \boldsymbol{\Delta}) \\ &= \beta \oplus \boldsymbol{u}_i^j \oplus \sigma((\boldsymbol{0} \| \boldsymbol{x}_i) \oplus (i \| j)) \\ &\quad \oplus \text{“other additive terms in } \boldsymbol{y}_i^j\text{”}, \end{aligned}$$

which is invertible to compute $(\boldsymbol{0} \| \boldsymbol{\Delta})$. This implies that the equality holds with probability $2^{-(\lambda-1)}$ due to the randomness of $\boldsymbol{\Delta}$. Taking a union bound, we have

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_2] \leq \frac{2N \cdot (q_1 + q_2 + q_3)}{2^{\lambda-1}}. \tag{54}$$

Finally, consider bad_3. Recall that each $\boldsymbol{x}_w$ has at least $\lambda - 1$ random non-LSBs. Since these bits are independent of $\cup_{\ell=1}^2 \mathcal{K}_\ell$, (40) holds with probability at most $2^{-(\lambda-1)}$ for some fixed $(\alpha, \dots)$ and (i, j). For (41), the mask sampled in the step 13 or the direct sampling in the step 13 or 16 of SimIn ensures that both halves of $\boldsymbol{u}_i^j \in \mathbb{F}^2_{2^{\lambda/2+1}}$ are uniform and independent of $\boldsymbol{x}_i$ or $\cup_{\ell=1}^2 \mathcal{K}_\ell$. It holds with probability $2^{-(\lambda+2)} < 2^{-(\lambda-1)}$ for some fixed $(\dots, \beta)$ and (i, j). So, we can take a union bound to have that

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_3] \leq \frac{2N \cdot (q_1 + q_2)}{2^{\lambda-1}}. \tag{55}$$

We have a bound ε_1 from (53), (54), and (55):

$$\begin{aligned} &\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) \in \mathcal{T}_{\mathsf{bad}}] = \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3] \\ &\leq \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1] + \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_2] + \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_3] \\ &\leq \frac{5N(N-1) + 16N(q+N)}{2^{\lambda+1}} \\ &\leq \frac{48qs + 189s^2 - 15s}{2^{\lambda+1}} = \varepsilon_1, \end{aligned}$$

where the last inequality comes from (51), $M \leq |\mathcal{W}_\cup(f)| \leq 3|f|$, and $|f| = s$.

The above ε_1, ε_2 and the H-coefficient technique lead to this theorem. □

```
SimF^{π^{±1}(·)}(f):
 1: $F := \{(\boldsymbol{g}^g, \boldsymbol{z}^g)\}_{g \in \mathcal{G}_{\mathsf{and}}(f)} \leftarrow (\mathbb{F}^3_{2^{\lambda/2}} \times \mathbb{F}^5_2)^{|f|}$
 2: $\{\boldsymbol{x}_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)} \leftarrow (\mathbb{F}^2_{2^{\lambda/2}})^{|\mathcal{W}_{\mathsf{in}}(f)|}$
 3: for $i \in \mathcal{W}_\cup(f)$ do $\mathsf{ctr}_i := 0$
 4: for $g \in \mathcal{G}(f)$ in order do
 5:     $(a, b, c) := (\mathsf{in}_0(g), \mathsf{in}_1(g), \mathsf{out}(g))$
 6:     if $\mathsf{type}(g) = \mathtt{XOR}$ then $\boldsymbol{x}_c := \boldsymbol{x}_a \oplus \boldsymbol{x}_b$
 7:     else if $\mathsf{type}(g) = \mathtt{AND}$ then
 8:         $\Gamma := \mathsf{in}_2(g)$
 9:         $(\chi_a, \rho_a) := (\lfloor \mathsf{ctr}_a/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_a))$, $\mathsf{ctr}_a := \mathsf{ctr}_a + 1$
10:         $(\chi_b, \rho_b) := (\lfloor \mathsf{ctr}_b/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_b))$, $\mathsf{ctr}_b := \mathsf{ctr}_b + 1$
11:         $(\chi_\Gamma, \rho_\Gamma) := (\lfloor \mathsf{ctr}_\Gamma/2 \rfloor, \mathsf{lsb}(\mathsf{ctr}_\Gamma))$, $\mathsf{ctr}_\Gamma := \mathsf{ctr}_\Gamma + 1$
12:         $s_a := \mathsf{lsb}(\boldsymbol{x}_a)$, $s_b := \mathsf{lsb}(\boldsymbol{x}_b)$
13:         $\begin{bmatrix} \mathsf{Half}_{\rho_a}(\boldsymbol{u}_a^{\chi_a}) \\ \mathsf{Half}_{\rho_b}(\boldsymbol{u}_b^{\chi_b}) \\ \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \end{bmatrix} := \begin{bmatrix} \mathsf{Half}_{\rho_a}(\pi((\boldsymbol{0} \| \boldsymbol{x}_a) \oplus (a \| \chi_a)) \oplus \sigma((\boldsymbol{0} \| \boldsymbol{x}_a) \oplus (a \| \chi_a))) \\ \mathsf{Half}_{\rho_b}(\pi((\boldsymbol{0} \| \boldsymbol{x}_b) \oplus (b \| \chi_b)) \oplus \sigma((\boldsymbol{0} \| \boldsymbol{x}_b) \oplus (b \| \chi_b))) \\ \mathsf{Half}_{\rho_\Gamma}(\pi((\boldsymbol{0} \| \boldsymbol{x}_a \oplus \boldsymbol{x}_b) \oplus (\Gamma \| \chi_\Gamma)) \oplus \sigma((\boldsymbol{0} \| \boldsymbol{x}_a \oplus \boldsymbol{x}_b) \oplus (\Gamma \| \chi_\Gamma))) \end{bmatrix}$
14:         $(\boldsymbol{r}^g_{s_a s_b} \| \boldsymbol{m}^g_{s_a s_b}) := \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathsf{Half}_{\rho_a}(\boldsymbol{u}_a^{\chi_a}) \\ \mathsf{Half}_{\rho_b}(\boldsymbol{u}_b^{\chi_b}) \\ \mathsf{Half}_{\rho_\Gamma}(\boldsymbol{u}_\Gamma^{\chi_\Gamma}) \end{bmatrix} \oplus \boldsymbol{V}_{s_a s_b}\left(\boldsymbol{z}^g \,\Big\|\, \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g}^g \end{bmatrix}\right)$
15:         $\boldsymbol{R}^g_{s_a s_b} := \mathsf{TH.DecodeR}(\boldsymbol{r}^g_{s_a s_b}, s_a, s_b)$
16:         $\boldsymbol{x}_c := \boldsymbol{m}^g_{s_a s_b} \oplus \boldsymbol{R}^g_{s_a s_b} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix}$
17: return $\widehat{f} := (f, F)$, $\mathsf{st}_{\mathsf{sim}} := (f, \widetilde{\boldsymbol{x}} := \{\boldsymbol{x}_i\}_{i \in \mathcal{W}(f)}, \widetilde{\boldsymbol{u}} := \{\boldsymbol{u}_i^j\}_{i \in \mathcal{W}_\cup(f), j \in [0, \lceil n_i(f)/2 \rceil - 1]}, \widetilde{\boldsymbol{r}} := \{\boldsymbol{r}^g_{s_a s_b}\}_{g \in \mathcal{G}_{\mathsf{and}}(f), (a,b) := (\mathsf{in}_0(g), \mathsf{in}_1(g))})$

SimIn^{π^{±1}(·)}(f(x)):
 1: Parse $\mathsf{st}_{\mathsf{sim}} = (f, \widetilde{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i \in \mathcal{W}(f)}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{r}})$
 2: for $i \in \mathcal{W}_{\mathsf{out}}(f)$ do $d_i := f(x)_i \oplus \mathsf{lsb}(\boldsymbol{x}_i)$
 3: return $\widehat{x} := (\{\boldsymbol{x}_i\}_{i \in \mathcal{W}_{\mathsf{in}}(f)}, d)$, $\widetilde{\boldsymbol{x}}$, $\widetilde{\boldsymbol{u}}$, $\widetilde{\boldsymbol{r}}$.
```

Figure 9: Our simulator for three-halves in the npRPM.

B.3. Adaptive Security in npRPM

We consider a slightly different three-halves implementation from the original one of [23] in that decoding table d is transferred from $\widehat{f}$ to k in $\mathsf{TH.Garble}^{\pi^{\pm 1}(\cdot)}$ and then included in garbled input $\widehat{x}$ in the online phase. This implementation also follows the lower bound in Appendix C and satisfies the adaptive security in the npRPM as per the following theorem.

Theorem 4. *Let $\mathsf{H}(\boldsymbol{x}, k) = \pi((\boldsymbol{0} \| \boldsymbol{x}) \oplus k) \oplus \sigma((\boldsymbol{0} \| \boldsymbol{x}) \oplus k)$ be a tweakable hash function where $\boldsymbol{x} \in \mathbb{F}^2_{2^{\lambda/2}}, k \in \mathbb{F}^2_{2^{\lambda/2+1}}$, $\pi \in \mathcal{S}_{\lambda+2}$ is random permutation, and $\sigma : \mathbb{F}^2_{2^{\lambda/2+1}} \to \mathbb{F}^2_{2^{\lambda/2+1}}$ is a linear orthomorphism for the function family*

$$\mathcal{L} = \left\{ L_{\xi_1,\xi_2,\xi_3,\xi_4} : \mathbb{F}^2_{2^{\lambda/2+1}} \to \mathbb{F}^2_{2^{\lambda/2+1}} \right\}_{\xi_1,\xi_2,\xi_3,\xi_4 \in \mathbb{F}_2},$$

$$L_{\xi_1,\xi_2,\xi_3,\xi_4}\left(\begin{bmatrix} x_L \\ x_R \end{bmatrix}\right) = \begin{bmatrix} \xi_1 x_L \oplus \xi_2 x_R \\ \xi_3 x_L \oplus \xi_4 x_R \end{bmatrix}.$$

Then, three-halves (sketched above) is a $(\lambda+2)$-garbling scheme with (q, s, ε)-adaptive security in the npRPM, where $\varepsilon = (69qs + 234s^2)/2^{\lambda+2}$.

Proof (Sketch). The correctness can be proved as [23] as postponing decoding table d does not affect correctness. We only need to consider the simulation.

Our simulator $\mathsf{Sim} = (\mathsf{SimF}, \mathsf{SimIn})$ is presented in Figure 9 and is obviously PPT. Then, we prove this theorem using the following three hybrids:

- Hybrid_0. This is the adaptive experiment using simulator Sim.
- Hybrid_1. This is identical to the previous hybrid, except that we replace $\pi^{\pm 1}$ (which can be equivalently emulated on-the-fly as in Figure 4) by an approximation $\widetilde{\pi}^{\pm 1}$ (given in Figure 5). This approximation is the same as random permutation except that, for a new query of the simulator, it returns a fresh random string as response and records this query-response pair. This hybrid is used to simplify probability analysis.
- Hybrid_2. This is the adaptive experiment using three-halves scheme.

According to Lemma 2, every adaptive adversary $\mathcal{A}$ can distinguish Hybrid_0 and Hybrid_1 with advantage at most $(q+N) \cdot N/2^{\lambda+1}$ up to the supposed q queries of $\mathcal{A}$ and the N queries of Sim.

Then, we prove the negligible statistical distance between the transcripts in Hybrid_1 and Hybrid_2, which upper bounds the advantage of adaptive adversary $\mathcal{A}$. To simplify probability analysis, we also use the H-coefficient technique and the same transcript padding as in the proof of Theorem 3. We regard Hybrid_1 (resp., Hybrid_2) and the associated oracle as the ideal (resp., real) world in the H-coefficient technique. The real-world sample space is the same as that in the proof of Theorem 3, but the ideal-world sample space is defined as

$$\begin{aligned}\Omega_{\mathsf{ideal}} = (\mathbb{F}_2^{3\lambda/2+5})^{|f|} \times (\mathbb{F}_2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|} \\ \times \underbrace{(\mathbb{F}_2^{\lambda/2+1})^{M}}_{\text{for the sampling of } \widetilde{T}} \\ \times \underbrace{\{0,1\}^*}_{\substack{\text{random tape for} \\ \text{the sampling in } \widetilde{\pi}^{\pm 1}(\cdot)}} \times \underbrace{\mathbb{F}_2^{\lambda-1}}_{\text{dummy } \boldsymbol{\Delta}} .\end{aligned}$$

We will consider the same bad transcripts as in the proof of Theorem 3 (where bad_3 is for probability analysis instead of programming, as its counterpart in the proof of Theorem 1) and assume that, without loss of generality, $\mathcal{A}$ only makes non-repeating queries. Let $q_\ell := |\mathcal{K}_\ell|$ for every $\ell \in \{1,2,3\}$ and $q_\Sigma := \sum_{\ell=1}^{3} q_\ell$.

Bounding $1-\varepsilon_2$. We can follow a similar argument (with the difference that the consistency (43) and (44) trivially hold for good transcripts due to the step 12 to 15 in SimF) in the proof of Theorem 3 to show that, for some fixed good transcript τ such that $\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) = \tau] \neq 0$ and $q_\Sigma = q+N$ (which comes from the condition $\neg\mathsf{bad}_1 \wedge \neg\mathsf{bad}_3$ for good transcripts),

$$\begin{aligned}&\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)] \\ &= \frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{(2^{\lambda+2} - 2N - (q_\Sigma - N))!}{(2^{\lambda+2})!} \cdot \frac{1}{2^{2|f|+(\lambda-1)}} \\ &= \frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}} \cdot \frac{1}{(2^{\lambda+2})_{q+2N}} \cdot \frac{1}{2^{2|f|+(\lambda-1)}}, \\ &\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)] \\ &\leq \frac{1}{(2^{\lambda+2} - N)_q} \\ &\quad \cdot \frac{1}{2^{|\mathcal{W}_{\mathsf{in}}(f)|\lambda + (3\lambda/2+5)|f| + (\lambda/2+1)M + (\lambda-1)} \cdot (2^{\lambda+2})^N} \\ &= \frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}} \\ &\quad \cdot \frac{1}{(2^{\lambda+2} - N)_q \cdot (2^{\lambda/2+1})^{3|f|+M+2N}} \cdot \frac{1}{2^{2|f|+(\lambda-1)}} \\ &= \frac{1}{(2^{\lambda})^{|\mathcal{W}_{\mathsf{in}}(f)|}} \\ &\quad \cdot \frac{1}{(2^{\lambda+2} - N)_q \cdot (2^{\lambda+2})^{2N}} \cdot \frac{1}{2^{2|f|+(\lambda-1)}}, \qquad \text{(By (51))} \\ &\Rightarrow \frac{\Pr_{\omega \leftarrow \Omega_{\mathsf{real}}}[\omega \in \mathsf{comp}_{\mathsf{real}}(\tau)]}{\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\omega \in \mathsf{comp}_{\mathsf{ideal}}(\tau)]} \geq 1.\end{aligned}$$

That is, we have $\varepsilon_2 = 0$.

Bounding ε_1. First, we consider $\mathsf{bad}_1 \vee \mathsf{bad}_3 = (35) \vee (36) \vee (37) \vee (40) \vee (41)$. We have a similar induction in the proof of Theorem 1 to prove the probability of bad values of some forward queries:

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[(35) \vee (40)] \leq \frac{N(N-1) + 2N \cdot (q_1 + q_2)}{2^{\lambda+1}}. \tag{56}$$

We consider (36), (37), and (41) conditioned on $\neg((35) \vee (40))$. In each of them, $\boldsymbol{u}_i^j \oplus \sigma((\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j))$ is the response for query $(\mathbf{0} \,\|\, \boldsymbol{x}_i) \oplus (i \,\|\, j)$ to $\widetilde{\pi}^{\pm 1}(\cdot)$. It follows from this condition that these queries are pairwise distinct so that their responses are taken from uniform $c_1, \ldots, c_{n(\lambda)}$ in $\widetilde{\pi}^{\pm 1}(\cdot)$, where $\ell(\lambda) = \lambda + 2$, and pairwise independent given the pairwise distinct queries. So, each of them occurs with probability $1/2^{\lambda+2}$ for some fixed quantifier. Taking a union bound over all quantifiers, we have

$$\begin{aligned}&\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}\left[(36) \vee (37) \vee (41) \;\middle|\; \neg((35) \vee (40))\right] \\ &\quad \leq \frac{N(N-1) + N \cdot (q_1 + q_2)}{2^{\lambda+2}}.\end{aligned} \tag{57}$$

Using (56) and (57), we have

$$\begin{aligned}&\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1 \vee \mathsf{bad}_3] \\ &= \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[(35) \vee (36) \vee (37) \vee (40) \vee (41)] \\ &= \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[(35) \vee (40)] \\ &\quad + \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}\left[(36) \vee (37) \vee (41) \;\middle|\; \neg((35) \vee (40))\right] \\ &\leq \frac{3N(N-1) + 5N \cdot (q_1 + q_2)}{2^{\lambda+2}}.\end{aligned} \tag{58}$$

Then, consider bad_2. It is easy to see from the randomness of $\boldsymbol{\Delta}$ that

$$\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_2] \leq \frac{2N \cdot (q_1 + q_2 + q_3)}{2^{\lambda-1}}. \tag{59}$$

We have a bound ε_1 from (58) and (59):

$$\begin{aligned}
&\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[Y(\omega) \in \mathcal{T}_{\mathsf{bad}}] \\
= &\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3] \\
\leq &\Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_1 \vee \mathsf{bad}_3] + \Pr_{\omega \leftarrow \Omega_{\mathsf{ideal}}}[\mathsf{bad}_2] \\
\leq &\frac{3N(N-1) + 21N(q+N)}{2^{\lambda+2}} \\
\leq &\frac{63qs + 216s^2 - 9s}{2^{\lambda+2}} = \varepsilon_1,
\end{aligned}$$

where the last inequality comes from (51), $M \leq |\mathcal{W}_{\cup}(f)| \leq 3\,|f|$, and $|f| = s$.

By using the H-coefficient technique with the above ε_1 and ε_2, we have that any adaptive adversary can distinguish Hybrid_1 and Hybrid_2 with advantage at most $(63qs + 216s^2 - 9s)/2^{\lambda+2}$.

Putting the three hybrids together, we arrive at this theorem. □

B.4. Public Parameters of Three-Halves

The following public constants are suggested by three-halves [23].

$$\boldsymbol{M} = \left[\begin{array}{cc|cc|cc} 1&0&0&0&1&0 \\ 0&0&1&0&1&0 \\ \hline 1&0&0&0&0&1 \\ 0&0&0&1&0&1 \\ \hline 0&1&0&0&0&1 \\ 0&0&1&0&0&1 \\ \hline 0&1&0&0&1&0 \\ 0&0&0&1&1&0 \end{array}\right] \in \mathbb{F}_2^{8\times 6},$$

$$\boldsymbol{K} = \left[\begin{array}{cc|cc|cc|cc} 1&0&1&0&1&0&1&0 \\ 0&1&0&1&0&1&0&1 \\ \hline 0&0&0&1&1&0&1&1 \end{array}\right] \in \mathbb{F}_2^{3\times 8},$$

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_{00} \\ \boldsymbol{V}_{01} \\ \boldsymbol{V}_{10} \\ \boldsymbol{V}_{11} \end{bmatrix} = \left[\begin{array}{cc|ccc} 1&0&0&0&0 \\ 0&1&0&0&0 \\ \hline 1&0&0&0&1 \\ 0&1&0&1&1 \\ \hline 1&0&1&0&1 \\ 0&1&0&0&1 \\ \hline 1&0&1&0&0 \\ 0&1&0&1&0 \end{array}\right] \in (\mathbb{F}_2^{2\times 5})^4 \equiv \mathbb{F}_2^{8\times 5},$$

$$\boldsymbol{V}^+ = \left[\begin{array}{cc|cc|cc|cc} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ \hline 1&1&0&0&1&1&0&0 \\ 1&1&1&1&0&0&0&0 \\ 0&0&0&0&1&0&1&0 \end{array}\right] \in \mathbb{F}_2^{5\times 8},$$

$$\boldsymbol{S}_L = \left[\begin{array}{cc|cc} 1&1&1&0 \\ 1&0&0&1 \end{array}\right] \in \mathbb{F}_2^{2\times 4}, \quad \boldsymbol{S}_R = \left[\begin{array}{cc|cc} 1&0&0&1 \\ 0&1&1&1 \end{array}\right] \in \mathbb{F}_2^{2\times 4},$$

$$\boldsymbol{R}'_1 = \begin{bmatrix} 0&0 \\ 1&1 \\ 0&1 \\ 1&0 \end{bmatrix} \in \mathbb{F}_2^{4\times 2}, \quad \boldsymbol{R}'_2 = \begin{bmatrix} 0&0 \\ 1&0 \\ 1&1 \\ 0&1 \end{bmatrix} \in \mathbb{F}_2^{4\times 2},$$

$$\boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{R}_{p,00} \\ \boldsymbol{R}_{p,01} \\ \boldsymbol{R}_{p,10} \\ \boldsymbol{R}_{p,11} \end{bmatrix} = \left[\begin{array}{cc|cc} 0&0&1&0 \\ 0&1&0&0 \\ \hline 0&0&1&0 \\ 0&0&0&0 \\ \hline 0&0&0&0 \\ 0&1&0&0 \\ \hline 0&0&0&0 \\ 0&0&0&0 \end{array}\right] \in (\mathbb{F}_2^{2\times 4})^4 \equiv \mathbb{F}_2^{8\times 4}.$$

The distribution $\mathcal{R}_0$ is defined as sampling two uniform bits $\begin{bmatrix} r_L & r_R \end{bmatrix} \leftarrow \mathbb{F}_2^2$ and returning

$$\begin{aligned}
\boldsymbol{R}'_{\$} = &\begin{bmatrix} \boldsymbol{R}'_{\$,00} \\ \boldsymbol{R}'_{\$,01} \\ \boldsymbol{R}'_{\$,10} \\ \boldsymbol{R}'_{\$,11} \end{bmatrix} \\
:= &r_L \cdot \begin{bmatrix} 1&0 \\ 1&0 \\ 1&0 \\ 1&0 \end{bmatrix} \oplus r_R \cdot \begin{bmatrix} 0&1 \\ 0&1 \\ 0&1 \\ 0&1 \end{bmatrix} \in (\mathbb{F}_2^{1\times 2})^4 \equiv \mathbb{F}_2^{4\times 2}.
\end{aligned}$$

Appendix C. Separating npRPM from pRPM for Adaptive Garbling

Recall Definition 1 of adaptive garbling. In the npRPM, the simulator of adaptive garbling consists of two PPT algorithms: $\mathsf{SimF}^{\pi^{\pm 1}(\cdot)}$ and $\mathsf{SimIn}^{\pi^{\pm 1}(\cdot)}$, which have oracle access to a random permutation π and its inverse π^{-1}. The former simulates garbled circuit $\widehat{f}$ (without online input x) and outputs an internal state, while the latter simulates garbled input $\widehat{x}$ given the internal state and circuit output $f(x)$. Intuitively, since the simulated $\widehat{x}$ is the only message that can depend on $f(x)$, $\widehat{x}$ should consist of "adequate" information of $f(x)$. Otherwise, we cannot reconstruct $f(x)$ from the simulated $(\widehat{f}, \widehat{x})$ in the ideal world, making the real world and ideal world trivially distinguishable.

Following prior works [6], [33], we evaluate this "adequacy" based on the Yao entropy [48], [49], [50] of $f(x)$. This entropy is formalized as the minimal bit-length of an efficiently computable compressed form of $f(x)$, which can be decompressed to $f(x)$ with probability significantly more than $1/2$. Similar to the standard-model lower bound in the prior works, we can prove that the bit-length of $\widehat{x}$ should be at least the Yao entropy of $f(x)$ in the npRPM. The high-level idea is that the simulator can compress $f(x)$ into $\widehat{x}$, which can be decompressed using the evaluation algorithm of garbling scheme with overwhelming probability. Otherwise, the ideal world is trivially distinguishable from the real one. Recall that the simulator and the evaluation algorithm are PPT and the adversary also makes a polynomial number of queries to $\pi^{\pm 1}$. The total number of permutation queries in such compression and decompression is polynomial, and their responses can be approximated by uniform strings up to negligible statistical distance. Thus, the above compression and decompression have polynomial-size circuits as required by Yao entropy, where the circuits hardcode the uniform responses used to approximate the real ones from $\pi^{\pm 1}$.

More specifically, we define an intermediate hybrid by replacing a random permutation and its inverse with "coarse" approximation $\mathring{\pi}^{\pm 1}$, which output a fresh random string upon a fresh query but ensures the query-response consistency. Clearly, the advantage of any adaptive adversary to distinguish this hybrid and the ideal world is at most twice the birthday bound, which is negligible up to a polynomial number of queries to the permutation and its inverse. As the garbling scheme is adaptively secure in the npRPM, the ideal world is indistinguishable from the real one. Thus, $\mathsf{SimF}^{\mathring{\pi}^{\pm 1}(\cdot)}$ and $\mathsf{SimIn}^{\mathring{\pi}^{\pm 1}(\cdot)}$ give an approximate simulation indistinguishable from the real world.

This indistinguishability implies that the approximate simulation results in $(\widehat{f}, \widehat{x})$ that can be decoded to $f(x)$ with overwhelming probability. If the bit-length of this $\widehat{x}$ is less than the Yao entropy of $f(x)$, we show that the approximate simulation contradicts this entropy. We can construct a compression circuit that sequentially runs $\mathsf{SimF}^{\mathring{\pi}^{\pm 1}(\cdot)}$ and $\mathsf{SimIn}^{\mathring{\pi}^{\pm 1}(\cdot)}$ over some hardcoded random tape to output such an $\widehat{x}$. The decompression circuit uses the same random tape to recompute $\widehat{f}$ output by $\mathsf{SimF}^{\mathring{\pi}^{\pm 1}(\cdot)}$, and runs the deterministic evaluation algorithm to compute $f(x)$ from $(\widehat{f}, \widehat{x})$. According to the coarse approximate permutation (which is hardcoded in the circuits), both circuits are of polynomial sizes. Thus, these circuits contradict the Yao entropy of $f(x)$, leading to a lower bound of the online complexity in the npRPM.

However, this lower bound does not hold in the pRPM, where the simulator has another PPT algorithm $\mathsf{SimP}^{\pm 1}$ to emulate a random permutation and its inverse for queries (a) before $\widehat{f}$, (b) after $\widehat{f}$ but before $\widehat{x}$, and (c) after $\widehat{x}$. It is notable that SimF, SimIn, and $\mathsf{SimP}^{\pm 1}$ maintain an internal state. In particular, given $f(x)$ and the internal state, SimIn outputs simulated $\widehat{x}$ and *an updated internal state to be used in* $\mathsf{SimP}^{\pm 1}$ *for case (c)*. Although we can prove that the output bit-length of SimIn should exceed the Yao entropy of $f(x)$, a part of the information of $f(x)$ can be transferred into the updated internal state so that the bit-length of the simulated $\widehat{x}$ can be lower than the Yao entropy. As a result, the lower bound does not hold for a real-world $\widehat{x}$ since the real and ideal worlds are indistinguishable for the adaptive security in the pRPM. This result is confirmed by our proofs for the adaptive security of half-gates and three-halves in the pRPM, where programming is performed to embed the decoding consistency, e.g., $\mathsf{lsb}(X_c) = x_c \oplus d_c$, into the internal state of the simulator.