New Indifferentiability Security Proof of MDPH Hash Function

Chun Guo[1], Tetsu Iwata[2], and Kazuhiko Minematsu[3]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China
chun.guo@sdu.edu.cn
[2] Nagoya University, Nagoya, Japan
tetsu.iwata@nagoya-u.jp
[3] NEC Corporation, Kawasaki, Japan
k-minematsu@nec.com

Abstract. MDPH is a double-block-length hash function proposed by Naito at Latincrypt 2019. This is a combination of Hirose's compression function and the domain extender called Merkle-Damgård with permutation (MDP). When instantiated with an n-bit block cipher, Naito proved that this achieves the (nearly) optimal indifferentiable security bound of $O(n - \log n)$-bit security. In this paper, we first point out that the proof of the claim contains a gap, which is related to the definition of the simulator in simulating the decryption of the block cipher. We then show that the proof can be fixed. We introduce a new simulator that addresses the issue, showing that MDPH retains its (nearly) optimal indifferentiable security bound of $O(n - \log n)$-bit security.

Keywords: Cryptography · Hash function · MDPH · Indifferentiability

1 Introduction

Cryptographic hash functions have wide applications, and constructing an efficient and secure hash function is an important research problem. For their wide applications, hash functions have to satisfy various security notions, including collision resistance, preimage resistance, and 2nd-preimage resistance. The security notion called indifferentiability [12] from the random oracle, put forward by Maurer et al., implies these security notions. There are various approaches to construct hash functions, and we consider one of the approaches based on a block cipher.

At Latincrypt 2019, Naito [13] proposed a block cipher-based hash function. It is essentially a combination of a double-block-length (DBL) compression function proposed by Hirose [7] and Merkle-Damgård with permutation (MDP) domain extender by Hirose et al. [8]. Both schemes are well-known and their security is well understood. However, when combined, a dedicated security analysis is needed to evaluate the indifferentiability from the random oracle. Naito [13]

showed that this combination, dubbed MDPH, indeed achieves $O(n - \log n)$-bit indifferentiability if it is realized with an n-bit block cipher, which is close to the optimal indifferentiable security bound.

As a DBL hash function, MDPH is quite efficient in terms of the number of primitive calls since it does not need a dedicated message-less finalization step, and it requires small state memory in its implementation. It has been adopted as the hashing mode of Romulus [9, 5], one of the finalists of the NIST Lightweight Cryptography for lightweight authenticated encryption (AE)[4]. More precisely, MDPH is specified as Romulus-H in the latest (v1.3) specification of Romulus [5], using Skinny [2] as the underlying tweakable block cipher. It is also the hashing component of leakage-resilient AE mode of Romulus, called Romulus-T (which is based on TEDT [3]). Recently, List [10] proposed an improved variant of TEDT that uses MDPH/Romulus-H as a component.

This article first shows that the analysis of [13] contains a gap. In the indifferentiability security proof, we consider a distinguisher that tries to distinguish between the real world and the ideal world. In the real world, the distinguisher has oracle access to MDPH, and also E and E^{-1}, the encryption and decryption of the underlying block cipher which is modelled as an ideal cipher. In the ideal world, the distinguisher has oracle access to a random oracle and a simulator, where the simulator has to simulate the encryption and decryption of the ideal cipher.

This gap is in the definition of the simulator simulating the decryption of the underlying block cipher. In the original analysis [13], the decryption simulator always performs lazy sampling for determining the block cipher input-output tuples. However, it turns out that this easily allows the distinguisher to break the consistency by first making an oracle call to the hash function, which is either MDPH or a random oracle, and then making a decryption query.

We then present a refined simulator in order to fix the issue. The refined decryption simulator is more involved, and the main approach is to keep all the hash computation paths implied by the queries, and the simulator performs lazy sampling only if it is really needed, to keep consistency with a high probability. As a result, we show that MDPH retains its nearly optimal $O(n - \log n)$-bit indifferentiability security as originally claimed.

Organization. We specify notations and present the security definition in Sect. 2. The specification of MDPH and our provable security bound are presented in Sect. 3, where we also point out issues in the analysis of [13] and present the main idea for fixing the proof. In Sect. 4, we specify our refined simulator. Our full proof is presented in Sect. 5, and we conclude the paper in Sect. 6.

[4] `https://csrc.nist.gov/projects/lightweight-cryptography`

2 Preliminaries

2.1 Notations

We adopt some of the notations of [13] for consistency. Let $\{0,1\}^*$ be the set of all bit strings, and λ be the empty string. For an integer $n \geq 0$, let $\{0,1\}^n$ be the set of all n-bit strings, and $(\{0,1\}^n)^*$ be the set of all bit strings whose lengths in bits are a multiple of n. For a non-negative integer i, let $[i] = \{0,1,\ldots,i\}$, the set of non-negative integers at most i. For positive integers ℓ, ν and an $\ell\nu$-bit string M, we write its partition into ν-bit strings as $M_1, M_2, \ldots, M_\ell \xleftarrow{\nu} M$. For positive integers r, s with $s \leq r$ and an r-bit string X, $[X]^s$ (resp. $[X]_s$) denotes the most (resp. least) significant s bits of X. For $S = [S]^n \| [S]_n \in \{0,1\}^{2n}$, we write its swap as $\overline{S} = [S]_n \| [S]^n$, where $X \| Y$ denotes the concatenation of bit strings X and Y. For a bit string Y, $X \leftarrow Y$ means that Y is assigned to X. For a finite set $\mathcal{X}$, $X \xleftarrow{\$} \mathcal{X}$ means that an element is sampled uniformly at random from $\mathcal{X}$ and is assigned to X. $\mathcal{Y} \leftarrow \mathcal{X}$ means that a finite set $\mathcal{X}$ is assigned to $\mathcal{Y}$, and $\mathcal{Y} \xleftarrow{\cup} \mathcal{X}$ means that $\mathcal{Y} \leftarrow \mathcal{X} \cup \mathcal{Y}$. For a positive integer a, $\mathsf{Func}(*, a)$ denotes the set of all functions from $\{0,1\}^*$ to $\{0,1\}^a$. For positive integers k and n, $\mathsf{BC}(k,n)$ denotes the set of all block ciphers with k-bit keys and n-bit blocks. Hence, for each $E \in \mathsf{BC}(k,n)$, $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is a set of permutations over $\{0,1\}^n$ indexed by a key in $\{0,1\}^k$. The decryption function of E is denoted by $E^{-1} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. For simplicity we focus on the case $k = 2n$ but extension to the general case $k > n$ is possible. For a positive integer i, let $\boldsymbol{i}_n$ be the n-bit representation of i, e.g., $\mathbf{1}_n = 0^{n-1}1$, $\mathbf{2}_n = 0^{n-2}10$, and $\mathbf{3}_n = 0^{n-2}11$.

2.2 Indifferentiability of Hash Function

Let $\mathsf{H}^E : \{0,1\}^* \to \{0,1\}^{2n}$ be a hash function that uses a block cipher $E \in \mathsf{BC}(k,n)$ as an oracle. The indifferentiability of H^E from a random oracle $\mathcal{RO} \xleftarrow{\$} \mathsf{Func}(*, 2n)$ is the indistinguishability between the real and ideal worlds described as follows. In the real world, a distinguisher (an adversary) D interacts with H^E and a pair of encryption and decryption functions (E, E^{-1}) of E. Here, E is the ideal cipher, that is, we let $E \xleftarrow{\$} \mathsf{BC}(k,n)$. In the ideal world, for a simulator S that has oracle access to a random oracle $\mathcal{RO}$, a distinguisher D interacts with $\mathcal{RO}$ and S. After the interaction, D outputs a bit to indicate the decision. We write $D^{\mathcal{O}} = b$ if D returns a decision bit b after interacting with oracles $\mathcal{O}$. For a simulator S, the advantage function of a distinguisher D is defined as

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathcal{RO},\mathsf{S}}(D) = \mathsf{Pr}\left[E \xleftarrow{\$} \mathsf{BC}(k,n); D^{\mathsf{H}^E, E, E^{-1}} = 1\right] - \mathsf{Pr}\left[\mathcal{RO} \xleftarrow{\$} \mathsf{Func}(*, 2n); D^{\mathcal{RO}, \mathsf{S}^{\mathcal{RO}}} = 1\right].$$

In the above definition, the probabilities are taken over D, E, $\mathcal{RO}$, and S. We say that our target hash function H is indifferentiable from the random oracle

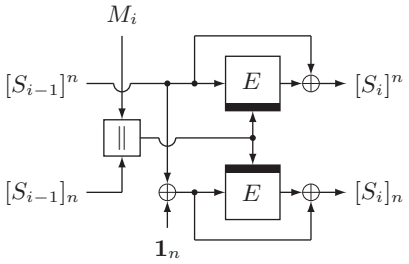

Fig. 1: Hirose compression function where $S_{i-1} \xrightarrow{M_i} S_i$.

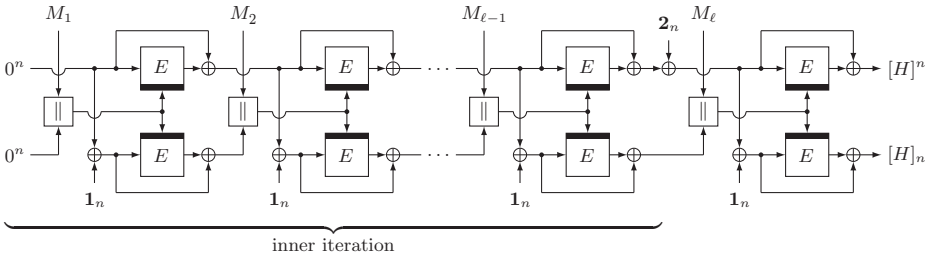

Fig. 2: MDPH hash function with block cipher E.

if for any D, there exists a simulator S such that $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathcal{RO},\mathsf{S}}(D)$ is negligibly small. Hereafter, H^E and $\mathcal{RO}$ are called hash oracles, queries to a hash oracle are called hash queries, (E, E^{-1}) and S are called primitive oracles, and queries to $(E, E^{-1})/\mathsf{S}$ are called primitive queries. Queries to E (resp. E^{-1}) are called encryption (resp. decryption) queries.

3 Specification of MDPH and Its Indifferentiability

As described earlier, the MDPH hash function was proposed by Naito [13]. It combines Merkle-Damgård with permutation (MDP) domain extender [8] and Hirose DBL compression function [7].

3.1 Specification

For $E \in \mathsf{BC}(k, n)$, let $\nu = k - n$. The Hirose compression function $\mathsf{Hirose}^E : \{0,1\}^{2n} \times \{0,1\}^\nu \to \{0,1\}^{2n}$ is defined as $\mathsf{Hirose}^E(S_{i-1}, M_i) = S_i$ for a ν-bit message M_i and a $2n$-bit state S_{i-1}, where

$$
\begin{aligned}
[S_i]^n &= E(M_i \,\|\, [S_{i-1}]_n, [S_{i-1}]^n) \oplus [S_{i-1}]^n && \text{(Top Part)}\,, \\
[S_i]_n &= E(M_i \,\|\, [S_{i-1}]_n, [S_{i-1}]^n \oplus \mathbf{1}_n) \oplus [S_{i-1}]^n \oplus \mathbf{1}_n && \text{(Bottom Part)}\,.
\end{aligned}
$$

In MDPH, Hirose^E is used as the compression function of a variant of the Merkle-Damgård construction. We use 0^{2n} as the initial state, and a simple

Algorithm $\mathsf{MDPH}^E(M)$

1. $S_0 \leftarrow 0^{2n}$
2. $M_1, M_2, \ldots, M_\ell \xleftarrow{\nu} \mathsf{pad}(M)$
3. **for** $i = 1, \ldots, \ell - 1$ **do** $S_i \leftarrow \mathsf{Hirose}^E(S_{i-1}, M_i)$
4. $[S_{\ell-1}]^n \leftarrow [S_{\ell-1}]^n \oplus \mathbf{2}_n$
5. $H \leftarrow \mathsf{Hirose}^E(S_{\ell-1}, M_\ell)$
6. **return** H

Fig. 3: MDPH hash function. The for-loop of line 3 is also called inner iteration in the proof sections.

permutation (XORing $\mathbf{2}_n$ to the top part) is applied before the final call of Hirose^E (See Figs. 2 and 3). The input message $M \in \{0,1\}^*$ is first padded with an injective padding function $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^\nu)^* \setminus \{\lambda\}$ so that the bit length of the padded message is always a multiple of ν. Originally, it is specified as $\mathsf{pad}(M) = M \,\|\, 10^p$, where $p = \nu - 1 - (|M| \mod \nu)$, but any injective padding function works.

3.2 Indifferentiablity Bound of MDPH

Our analysis fixes the flaws of the original analysis [13]. As a result, we present the following bound. The proof is given in Sect. 4 and 5.

Theorem 1. *Let D be a distinguisher that makes q hash queries of σ message blocks in total and p primitive queries, and runs in time t, which we call a (p, q, σ)-indifferentiability distinguisher. Let $Q = \sigma + p$, and assume $Q \le 2^n/6$ and $n \ge 3$. Then, there exists a simulator S such that*

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MDPH},\mathcal{RO},\mathsf{S}}(D) \le \frac{6(3Q+np)^2}{(2^n-2Q)^2} + \frac{11nQ+np}{2^n-2Q}\,.$$

The simulator makes no more than np queries to $\mathcal{RO}$ in total and runs in time $t + O(np)$.

Naito's bound [13] introduced the parameter μ for the maximum size of collisions, and it was optimized by setting $\mu = n$. This also applies to our analysis, and we present the case of $\mu = n$ for simplicity.

Implication to Classical Notions. To further derive the exact collision and preimage security of MDPH^E in the ideal cipher model, the influence of the np simulator query complexity from Theorem 1 has to be carefully analyzed. In detail, according to the concrete version of the indifferentiability composition theorem [14, Theorem 3.1], for any (p, q, σ)-collision adversary $\mathcal{A}^E$ against

MDPH^E, there exists a collision adversary $\mathcal{B}^{\mathcal{RO}}$ against $\mathcal{RO}$, such that

$$\begin{aligned} & \Pr\left[\mathcal{A}^E \Rightarrow (M_1, M_2) : \mathsf{MDPH}^E(M_1) = \mathsf{MDPH}^E(M_2)\right] \\ \leq\ & \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MDPH},\mathcal{RO},\mathsf{S}}(\mathcal{A}) + \Pr\left[\mathcal{B}^{\mathcal{RO}} \Rightarrow (M_1, M_2) : \mathcal{RO}(M_1) = \mathcal{RO}(M_2)\right] \\ \leq\ & \frac{6(3Q+np)^2}{(2^n-2Q)^2} + \frac{11nQ+np}{2^n-2Q} + \frac{(np+q)^2}{2^{2n}}, \end{aligned}$$

indicating collision security up to $2^n/n$ queries. Similarly, for any (p, q, σ)-preimage adversary $\mathcal{A}^E$ trying to find the preimage of any H on MDPH^E, there exists a preimage adversary $\mathcal{B}^{\mathcal{RO}}$ making nQ queries to find the preimage of H on $\mathcal{RO}$, such that

$$\begin{aligned} & \Pr\left[\mathcal{A}^E \Rightarrow M : \mathsf{MDPH}^E(M) = H\right] \\ \leq\ & \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MDPH},\mathcal{RO},\mathsf{S}}(\mathcal{A}) + \Pr\left[\mathcal{B}^{\mathcal{RO}} \Rightarrow M : \mathcal{RO}(M) = H\right] \\ \leq\ & \frac{6(3Q+np)^2}{(2^n-2Q)^2} + \frac{11nQ+np}{2^n-2Q} + \frac{np+q}{2^{2n}}, \end{aligned}$$

indicating everywhere preimage security up to $2^n/n$ queries. Fortunately, the np simulator query complexity turns out to be harmless in these settings.

3.3 Issues in the Original Proof

We describe the issues in the original proof [13]. We use some notations that appear in Sect. 4 and 5.

Flaw in the Original Simulator. In the original simulator in [13], assume that a distinguisher makes a decryption query $(\mathrm{Key}, \mathrm{Block}) = (K, Y)$. If the query was previously made and the simulator responded with X, then X is returned. Otherwise, the simulator randomly samples X from the set of possible plaintexts for the key K, and X is returned.

There exists the following attack for the simulator defined in [13].

1. Makes a (single block) hash query 0^ν and receives the response H;
2. Makes a decryption query $(\mathrm{Key}, \mathrm{Block}) = (0^\nu \| 0^n, [H]^n \oplus \mathbf{2}_n)$ and receives the response X;
3. If $X = \mathbf{2}_n$ then return 0; Else return 1.

In the ideal world, (H, X) is $(\mathcal{RO}(0^\nu), \mathsf{S}^{\mathcal{RO}}_{E^{-1}}(0^\nu \| 0^n, [H]^n \oplus \mathbf{2}_n))$, while in the real world, (H, X) is $(\mathsf{MDPH}^E(0^\nu), E^{-1}(0^\nu \| 0^n, [H]^n \oplus \mathbf{2}_n))$, where $\mathsf{S}^{\mathcal{RO}}_{E^{-1}}$ denotes the decryption simulator. In the real world, the above distinguisher returns 0, but in the ideal world, it returns 1 with high probability. This holds because the original simulator neglects the necessity to adapt simulated ideal cipher responses upon decryption queries and always samples X in the above case. We note that this attack is easily extended to using longer messages (namely, checking the consistency of the simulated decryption responses and the computed hash chain values).

Further issues in the Original Proof. The original proof [13] does not address the case where multiple hash chains are to-be-adapted within the same block. While this may be established, there is no argument for this. In the argument for the bad events, it was assumed that simulated ideal cipher entries are all defined "randomly". But since the simulator $(\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$ uses responses from $\mathcal{RO}$ to "adapt" a number of its ideal cipher entries, it is not all clear that every ideal cipher entry has a "random endpoint". In fact, in some cases (e.g., the event Hit_2 defined in [13]), the established probability for the bad events does not hold.

3.4 Fixing the Original Simulator

To facilitate understanding our new proof, we provide some intuitions here. As in the previous section, we use some notations from Sect. 4 and 5. Informally, a block is a triple (S_{i-1}, M_i, S_i) which is obtained from the distinguisher's queries/responses history and satisfies the relation $S_i = \mathsf{Hirose}^E(S_{i-1}, M_i)$. A path is a block or a concatenation of blocks which starts from the initial value 0^{2n}. Informally, a path captures the computations of a complete hash evaluation *before* the final block.

In the real world, if the distinguisher derives a path $\left(0^{2n} \xrightarrow{M} S\right)$ and a block $\left(S \oplus (\mathbf{2}_n \| 0^n)\right) \xrightarrow{M^*} H$ from its responses, then it necessarily holds $H = \mathsf{MDPH}^E(M \| M^*)$. In order to be consistent, in the ideal world the simulator shall "detect" such situations and define at least of the involved blocks such that $H = \mathcal{RO}(M \| M^*)$. Blocks that are defined to ensure consistency are called *adapted*.

It is easy to see that, if two paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S\right)$ with distinct messages though identical endpoints are created during the ideal world interaction, then the simulator has to define two blocks $\left(S \oplus (\mathbf{2}_n \| 0^n)\right) \xrightarrow{M^*} \mathcal{RO}(M \| M^*)$ and $\left(S \oplus (\mathbf{2}_n \| 0^n)\right) \xrightarrow{M^*} \mathcal{RO}(M' \| M^*)$, which are likely contradictory. Therefore, to prove indifferentiability, we at least have to argue that such colliding paths are unlikely to appear. To this end, a core lemma is that, *with high probability, paths only consist of blocks that are defined by sampling a pair of ciphertexts for an encryption query.* Actually this seems to be among the ideas of Naito [13], though the treatment has glitches. To bridge, we formally distinguish adapted and sampled blocks using the *explicit bookkeeping* technique of [1], and formally establish this lemma via induction on the adversarial queries. The reader is referred to Sect. 5.2 for a detailed description.

Finally, to rigorously reduce the indifferentiability claim to the non-occurrence of the "bad events", we employ the randomness mapping argument. Our formalism follows [6], though we bypass the intermediate system with some additional tricks.

Algorithm 1 Simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$

Initialization

1: $\mathsf{S}^{\mathcal{RO}}$ maintains a table PathT to keep track of encountered paths, indexed by their $2n$-bit endpoints. Initially, $\mathsf{PathT}[0^{2n}] = \left(0^{2n} \xrightarrow{\lambda} 0^{2n}\right)$ for the empty string λ, while $\mathsf{PathT}[S] = \perp$ for any other $S \in \{0,1\}^{2n} \backslash \{0^{2n}\}$.
2: $\mathsf{S}^{\mathcal{RO}}$ maintains another index structure Index on PathT, such that $\mathsf{Index}[R]$ returns the set $\left\{\left(0^{2n} \xrightarrow{M} S\right) : [S]_n = R\right\}$.
3: $\mathsf{S}^{\mathcal{RO}}$ maintains two tables ET and ET^{-1}, with entries initialized to $\perp$.
4: $\mathsf{S}^{\mathcal{RO}}$ maintains a set $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ for the already defined blocks. Elements in $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ are of the form $\left(S_1 \xrightarrow{M} S_2, d\right)$, $S_1, S_2 \in \{0,1\}^{2n}$, $M \in \{0,1\}^{\nu}$, $d \in \{\rightarrow, \leftarrow, \perp\}$. Initially, $\mathcal{L}_{\mathsf{block}}(\mathsf{ET}) = \left\{\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)\right\}$.

Simulator $\mathsf{S}_E^{\mathcal{RO}}(K, X)$

1: **if** $\mathsf{ET}[K](X) \neq \perp$ **then return** $\mathsf{ET}[K](X)$
2: **if** $\mathsf{PathT}[(X \| [K]_n) \oplus (\mathbf{2}_n \| 0^n)] \neq \perp \wedge \mathsf{PathT}[(X \oplus \mathbf{1}_n \| [K]_n) \oplus (\mathbf{2}_n \| 0^n)] \neq \perp$ **then**
3: **abort**
4: **end if**
5: $S \leftarrow X \| [K]_n$
6: **if** $\mathsf{PathT}[S \oplus (\mathbf{2}_n \| 0^n)] \neq \perp$ **then**
7: $\left(0^{2n} \xrightarrow{M} S \oplus (\mathbf{2}_n \| 0^n)\right) \leftarrow \mathsf{PathT}\left[S \oplus (\mathbf{2}_n \| 0^n)\right]$
8: $H \leftarrow \mathcal{RO}\left(M \| [K]^{\nu}\right)$
9: $Y_1 \leftarrow [H]^n \oplus X$; $Y_2 \leftarrow [H]_n \oplus X \oplus \mathbf{1}_n$
10: $\mathsf{CreateBlock}\left((K, X, Y_1), (K, X \oplus \mathbf{1}_n, Y_2), \perp\right)$
11: **end if**
12: $S \leftarrow X \oplus \mathbf{1}_n \| [K]_n$
13: **if** $\mathsf{PathT}[S \oplus (\mathbf{2}_n \| 0^n)] \neq \perp$ **then**
14: $\left(0^{2n} \xrightarrow{M} S \oplus (\mathbf{2}_n \| 0^n)\right) \leftarrow \mathsf{PathT}\left[S \oplus (\mathbf{2}_n \| 0^n)\right]$
15: $H \leftarrow \mathcal{RO}\left(M \| [K]^{\nu}\right)$
16: $Y_1 \leftarrow [H]^n \oplus X \oplus \mathbf{1}_n$; $Y_2 \leftarrow [H]_n \oplus X$
17: $\mathsf{CreateBlock}\left((K, X \oplus \mathbf{1}_n, Y_1), (K, X, Y_2), \perp\right)$
18: **end if**
19: **if** The conditions of lines 6 and 13 are not satisfied **then**
20: $Y_1 \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{Rng}[K]$; $Y_2 \xleftarrow{\$} \{0,1\}^n \backslash (\mathsf{Rng}[K] \cup \{Y_1\})$
21: $\mathsf{CreateBlock}\left((K, X \oplus \mathbf{1}_n, Y_1), (K, X, Y_2), \rightarrow\right)$
22: **end if**
23: **return** $\mathsf{ET}[K](X)$

4 Simulator

Formally, the simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$ is defined in Algorithm 1, where $\mathsf{S}_E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ simulates an encryption oracle E, and $\mathsf{S}_{E^{-1}} : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ simulates a decryption oracle E^{-1}. In order to work properly, the simulator S keeps multiple data structures: first, it keeps query-response triples in table ET and ET^{-1} whose entries are the special symbol $\perp$ indicating "undefined". If a query-response triple (K, X, Y) where $\mathsf{S}_E(K, X) = Y$

Algorithm 1 Simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$ (Cont'd)

Simulator $\mathsf{S}_{E^{-1}}^{\mathcal{RO}}(K, Y)$

if $\mathsf{ET}^{-1}[K](Y) \neq \perp$ **then return** $\mathsf{ET}^{-1}[K](Y)$
if $\left|\mathsf{Index}\left[[K]_n\right]\right| \geq n$ **then abort**
for all $\left(0^{2n} \xrightarrow{M} S\right) \in \mathsf{Index}\left[[K]_n\right]$ **do**
  $H \leftarrow \mathcal{RO}(M \| [K]^\nu)$
  // If $\mathsf{ET}^{-1}[K](Y)$ acts as the top part: check the chaining condition
  if $[H]^n \oplus Y = [S]^n \oplus \mathbf{2}_n$ **then**
    $X \leftarrow [H]^n \oplus Y$
    $Y' \leftarrow X \oplus \mathbf{1}_n \oplus [H]_n$
    $\mathsf{CreateBlock}\big((K, X, Y), (K, X \oplus \mathbf{1}_n, Y'), \perp\big)$
  end if
  // If $\mathsf{ET}^{-1}[K](Y)$ acts as the bottom part: check the chaining condition
  if $[H]_n \oplus Y \oplus \mathbf{1}_n = [S]^n \oplus \mathbf{2}_n$ **then**
    $X \leftarrow [H]_n \oplus Y$
    $Y' \leftarrow X \oplus \mathbf{1}_n \oplus [H]^n$
    $\mathsf{CreateBlock}\big((K, X \oplus \mathbf{1}_n, Y'), (K, X, Y), \perp\big)$
  end if
end for
if $\mathsf{ET}^{-1}[K](Y) = \perp$ **then**
  $X \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{Dom}[K]$; $Y' \xleftarrow{\$} \{0,1\}^n \backslash (\mathsf{Rng}[K] \cup \{Y\})$
  $\mathsf{CreateBlock}\big((K, X, Y), (K, X \oplus \mathbf{1}_n, Y'), \leftarrow\big)$
end if
return $\mathsf{ET}^{-1}[K](Y)$

or $\mathsf{S}_{E^{-1}}(K, Y) = X$ is defined, then the tables are defined such that $\mathsf{ET}[K](X) = Y$ and $\mathsf{ET}^{-1}[K](Y) = X$. The simulator S also keeps paths in $\mathcal{T}_{path}$ that are constructed from ET and ET^{-1}. We write $X \in \mathsf{ET}[K]$ if $\mathsf{ET}[K](X) \neq \perp$, $Y \in \mathsf{ET}^{-1}[K]$ if $\mathsf{ET}^{-1}[K](Y) \neq \perp$, and vice versa. We also write $\mathsf{Dom}[K] = \{X \in \{0,1\}^n : X \in \mathsf{ET}[K]\}$ and $\mathsf{Rng}[K] = \{Y \in \{0,1\}^n : Y \in \mathsf{ET}^{-1}[K]\}$.

A *block* is a triple (S_{i-1}, M_i, S_i) which is obtained from $\mathcal{L}_{\mathsf{qr}}$ and satisfies the relation $S_i = \mathsf{Hirose}^{R_E}(S_{i-1}, M_i)$. The block is denoted by $\left(S_{i-1} \xrightarrow{M_i} S_i\right)$ (see Fig. 1). We refer to $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$ as the *initial block*. $\mathsf{S}^{\mathcal{RO}}$ maintains an additional set $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ for the already defined blocks. Elements in $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ are of the form $\left(S_1 \xrightarrow{M} S_2, d\right)$, $S_1, S_2 \in \{0,1\}^{2n}$, $M \in \{0,1\}^m$, $d \in \{\rightarrow, \leftarrow, \perp\}$. Initially, $\mathcal{L}_{\mathsf{block}}(\mathsf{ET}) = \left\{\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)\right\}$.

A *path* is a block or a concatenation of blocks which starts from the initial value 0^{2n}. For a sequence of blocks $0^{2n} \xrightarrow{M_1} S_1, S_1 \xrightarrow{M_2} S_2, \ldots, S_{\ell-2} \xrightarrow{M_{\ell-1}} S_{\ell-1}$, we denote the concatenation by $0^{2n} \xrightarrow{M_1 \| M_2 \| \cdots \| M_{\ell-1}} S_{\ell-1}$. Hence, the path represents the inner iteration of MDPH^{R_E} (See Fig. 2). $\mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ is a set of all paths obtained from $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$. To ease access, $\mathsf{S}^{\mathcal{RO}}$ maintains a table PathT to keep track of encountered paths. In PathT, paths are indexed by their $2n$-bit end value, so that the time complexity for random access is constant.

Algorithm 1 Simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$ (Cont'd)

Subroutine $\mathsf{CreateBlock}((K, X, Y_1), (K, X \oplus \mathbf{1}_n, Y_2), dir)$

// Update the ideal cipher tables
if $dir = \perp$ **then** // Inconsistency only possible for adapted blocks
  if $X \in \mathsf{ET}[K]$ or $X \oplus \mathbf{1}_n \in \mathsf{ET}[K]$ **then abort**
  if $Y_1 \in \mathsf{ET}^{-1}[K]$ or $Y_2 \in \mathsf{ET}^{-1}[K]$ or $Y_1 = Y_2$ **then abort**
end if
$\mathsf{ET}[K](X) \leftarrow Y_1; \mathsf{ET}[K](X \oplus \mathbf{1}_n) \leftarrow Y_2; \mathsf{ET}^{-1}[K](Y_1) \leftarrow X; \mathsf{ET}^{-1}[K](Y_2) \leftarrow X \oplus \mathbf{1}_n$

$S^{(1)} \leftarrow X \| [K]_n$, $S^{(2)} \leftarrow (X \oplus Y_1) \| (X \oplus \mathbf{1}_n \oplus Y_2)$, $m \leftarrow [K]^\nu$
$S^{(3)} \leftarrow (X \oplus \mathbf{1}_n) \| [K]_n$, $S^{(4)} \leftarrow (X \oplus \mathbf{1}_n \oplus Y_2) \| (X \oplus Y_1)$
Add $\left(S^{(1)} \xrightarrow{m} S^{(2)}, dir\right)$ and $\left(S^{(3)} \xrightarrow{m} S^{(4)}, dir\right)$ to $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$

// Update the table PathT and check abort conditions
if $\mathsf{PathT}[S^{(1)}] \neq \perp$ **then**
  $\left(0^{2n} \xrightarrow{M'} S^{(1)}\right) \leftarrow \mathsf{PathT}[S^{(1)}]$
  if $\mathsf{PathT}[S^{(2)}] \neq \perp$ **then abort**
  $\mathsf{PathT}[S^{(2)}] \leftarrow \left(0^{2n} \xrightarrow{M' \| M} S^{(2)}\right)$
end if
if $\mathsf{PathT}[S^{(3)}] \neq \perp$ **then**
  $\left(0^{2n} \xrightarrow{M''} S^{(3)}\right) \leftarrow \mathsf{PathT}[S^{(3)}]$
  if $\mathsf{PathT}[S^{(4)}] \neq \perp$ **then abort**
  $\mathsf{PathT}[S^{(4)}] \leftarrow \left(0^{2n} \xrightarrow{M'' \| M} S^{(2)}\right)$
end if

Update the index Index

In the pseudocodes, $P \leftarrow \mathsf{PathT}[L]$ means a read of path P from $\mathsf{PathT}[L]$ for a label (an endpoint) $L \in \{0,1\}^{2n}$, and $\mathsf{PathT}[L] \leftarrow P$ means a write of path P to $\mathsf{PathT}[L]$. In principle $\mathsf{PathT}[L]$ for any L may contain multiple paths, and if so a path read picks arbitrary one and a path write adds one to the existing ones. However, the game is defined so that it aborts whenever such read/write operation is going to happen.

In the real world, for a hash query $\mathsf{MDPH}^E(M' \| M^*) \to H$, the response is defined by using E via the MDPH structure, and thus the following is satisfied:

$$\exists \left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET}), \left(S^* \xrightarrow{M^*} H\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET}) \text{ s.t. } S^* = S' \oplus (\mathbf{2}_n \| 0^n) : \mathsf{MDPH}^E(M' \| M^*) = H. \tag{1}$$

On the other hand, in the ideal world, for a hash query, the hash value is defined by a monolithic function $\mathcal{RO}$. Hence, the goal of simulator is to simulate an ideal cipher so that the query-responses of $\mathcal{RO}$ and of S satisfy the relation given in Eq. (1).

In order to ensure the relation in Eq. (1) in the ideal world, for an encryption query (K, X), if there exists a path $\left(0^{2n} \xrightarrow{M'} S'\right)$ in $\mathcal{T}_{path}$ to which the query connects, i.e., $S' = (X \oplus \mathbf{2}_n)\|[K]_n$ or $S' = (X \oplus \mathbf{1}_n \oplus \mathbf{2}_n)\|[K]_n$, then the response is defined by using $\mathcal{RO}$ (otherwise the response is defined by lazy sampling). For Hirose compression function, when a top (resp., bottom) part is defined, the bottom (resp., top) part can immediately be defined. In addition, switching top and bottom parts, the result also has Hirose's structure. Hence, for each (encryption or decryption) query, two outputs corresponding with the top and bottom parts are defined, and two blocks are defined. The initialization procedure is performed before the first query by D.

As explained earlier, S aborts upon certain bad situations. In particular, this includes the case where S fails to simulate an ideal cipher. In addition, S also aborts if it encounters certain bad situations, e.g., obtaining distinct paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S\right)$ with identical endpoint (see lines 15 and 20 of CreateBlock).

5 Indistinguishability of the Real and Ideal Worlds

5.1 Preparations

We assume that D makes no repeated query, and ignore the padding, that is, the length of hash queries made by D is always a multiple of ν.

We define two cryptographic games. In each game, a distinguisher D interacts with three oracles $(L, R_E, R_{E^{-1}})$, where L is a hash oracle, and $(R_E, R_{E^{-1}})$ are primitive oracles. Game 1 or G_1 captures the interaction between the distinguisher and the ideal world oracles $(L, R_E, R_{E^{-1}}) = (\mathcal{RO}, \mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$, where $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_{E^{-1}}^{\mathcal{RO}})$ is the simulator defined in Sect. 4. Game 2 or G_2 has $(L, R_E, R_{E^{-1}}) = (\mathsf{MDPH}^E, E, E^{-1})$, i.e., it captures the interaction with the real world oracles.

In the subsequent analyses in Sect. 5.4–5.6, D is permitted to make additional encryption queries after finishing the "normal" q hash and p primitive queries but before outputting a decision bit. There are two types of additional queries.

- The first type is that D can make queries to R_E according to the procedure of $\mathsf{MDPH}^{R_E}(M)$ for all hash queries $L(M)$, i.e., D can obtain all input-output triples of R_E to calculate $\mathsf{MDPH}^{R_E}(M)$.
- The second type is that D can make queries to R_E whose partner according to Hirose's structure has been defined by a primitive query. More precisely, if D obtains a triple (K, X, Y) by a primitive query but does not obtain the partner $(K, X \oplus \mathbf{1}_n, Y')$, then D can make the additional encryption query $(K, X \oplus \mathbf{1}_n)$ and obtain the response Y'.

With this added power, D makes at most $2Q$ primitive queries in total. Note that the additional queries do not reduce the advantage of D. Moreover, $|\mathcal{L}_{\mathsf{block}}(\mathsf{ET})| \leq 2Q$, i.e., these queries create at most $2Q$ blocks in total.

As additional notations:

– We imagine that the game G_1 maintains a set $\mathsf{HQueries}$ that records *all* random oracle queries appearing during the execution. Namely, every time the distinguisher D or the simulator $\mathsf{S}^{\mathcal{RO}}$ makes a random oracle query $\mathcal{RO}(M) \to H$, the pair (M, H) is added to the set $\mathsf{HQueries}$. Note that pairs added to $\mathsf{HQueries}$ due to simulator queries may be unknown to D.
– A block $\left(S \xrightarrow{m} T, d\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ is called *leftward* if $d = \leftarrow$; *rightward* if $d = \rightarrow$; and *adapted* if $d = \perp$. Note that the initial block $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ is also viewed as rightward even if it isn't created due to D querying $\mathsf{S}_E^{\mathcal{RO}}$.
– A *simulator cycle* consists of the execution period starting from when an adversary makes a query to when the adversary receives an answer. During a simulator cycle, if the simulator makes more than 1 call to $\mathsf{CreateBlock}(\star, \star, \perp)$, then it is an *adapted (simulator) cycle*. Otherwise,
  • it is a *rightward (simulator) cycle* if the simulator cycle was due to D querying $\mathsf{S}_E^{\mathcal{RO}}$;
  • it is a *leftward (simulator) cycle* if it was due to D querying $\mathsf{S}_{E^{-1}}^{\mathcal{RO}}$.

5.2 Outline of the Proof, and Intuitions

Indifferentiability requires to establish indistinguishability of G_1 and G_2. In addition, the simulator has to be efficient. In this respect, the remaining proof consists of four steps as follows.

1. First, we establish an upper bound on the simulator complexities in Sect. 5.3;
2. Second, in Sect. 5.4, we identify a number of bad events that could potentially render the simulation failed in G_1 executions, i.e., the interaction between D and the ideal world oracles;
3. Then, assuming absence of the aforementioned bad events, in Sect. 5.5 we show that the simulated oracle responses are indeed consistent with the random oracle responses (which is clearly a necessary condition for indifferentiability);
4. Finally, we conclude by bounding the distance between G_1 and G_2 in Sect. 5.6. The technique is a variant of Holenstein et al.'s randomness mapping argument [4] developed by Guo and Lin [6].

The simulator complexities are, in fact, somewhat obvious, since the simulator aborts once "out-of-control". It thus accounts for proving that these types of abortions are unlikely. More concretely, it means:

– The number of random oracle queries made during processing every decryption query is limited. Moreover, to achieve nearly optimal security, this number has to be bounded to a small constant;
– Despite the attempts, there is at most 1 hash chain that is completed during processing every decryption query.

Adaptations due to encryption queries: left hand side. As mentioned in Sect. 3.4, if distinct paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S\right)$ with the same endpoint are created during a G_1 execution, then the simulator has to define two contradictory blocks $\left(\left(S \oplus (\mathbf{2}_n \| 0^n)\right) \xrightarrow{M^*} \mathcal{RO}(M \| M^*)\right)$ and $\left(\left(S \oplus (\mathbf{2}_n \| 0^n)\right) \xrightarrow{M^*} \mathcal{RO}(M' \| M^*)\right)$. Therefore, to prove indifferentiability, we at least have to argue that such colliding paths are unlikely to appear. If all blocks are built upon ideal cipher query records that are defined via lazy sampling, then a straightforward argument following Hirose [7] could establish collision resistance and multi-half-collision resistance among the right hand sides of the blocks. However, an adapted block has its right hand side defined as the response of a random oracle query, and it is not clear how to argue that a random oracle query would not be used to define two (or more) adapted blocks. Moreover, (at first glance) adapted blocks may appear anywhere in the paths, which significantly complicates relevant arguments.

To see how this could be settled, let's summarize a number of (easy-to-see) observations as follows.

- After the distinguisher making an encryption query, if the simulator does not detect the aforementioned adapting situation, then it defines a pair of ideal cipher query records with randomly sampled ciphertexts. Consequently, the blocks $\left(S \xrightarrow{m} S'\right)$ and $\left(S \oplus (\mathbf{1}_n \| 0^n) \xrightarrow{m} \overline{S'}\right)$ formed by this pair of simulated queries have their right hand side random, and
  - they are unlikely to collide with existing blocks/paths. By this, the above contradictory paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S\right)$ won't appear after creating these records. This will be formalized as the event BCol in Sect. 5.4.
  - they are unlikely to "hit" existing blocks. By this, at most two paths are "extended" by this action:
    - The path $\left(0^{2n} \xrightarrow{M} S\right)$, if existed, has its length increased by 1, while the resulted path $\left(0^{2n} \xrightarrow{M \| m} S'\right)$ cannot further "extend";
    - The path $\left(0^{2n} \xrightarrow{M'} S \oplus (\mathbf{1}_n \| 0^n)\right)$, if existed, has its length increased by 1, while the resulted path $\left(0^{2n} \xrightarrow{M' \| m} \overline{S}\right)$ cannot further "extend".

    This idea will be formalized as the event Hit in Sect. 5.4.
- After the distinguisher making a decryption query, if the simulator does not detect adapting situation, then it defines a pair of ideal cipher query records with one plaintext sampled and the other ciphertext sampled. In this case, the blocks formed by this pair of simulated queries have their left hand side somewhat random, and they are unlikely to "hit" the endpoints of existing paths. By this, such *"leftward" blocks won't join in any path.* This will be formalized as the event DHit in Sect. 5.4.

By the above, before the first adapted block is created, *all paths are fully constituted by "rightward" blocks*, and thus various types of harmful collisions can be proved unlikely, including the aforementioned two paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S\right)$ and two paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S'\right)$ with $S' = S \oplus (\mathbf{2}_n \| 0^n)$. The first adapted block $\left(S^* \xrightarrow{M^*} H\right)$ necessarily has a corresponding path $\left(0^{2n} \xrightarrow{M} S\right)$ such that $S^* = S \oplus (\mathbf{2}_n \| 0^n)$. Since paths of the form $0^{2n} \xrightarrow{M'} S'$ with $S' = S \oplus (\mathbf{2}_n \| 0^n)$ didn't exist, this *adapted block won't join in any path either*, and the property that all paths are fully constituted by "rightward" blocks *is preserved*.

Following this intuition, we argue (via an induction) that:

- Paths are fully constituted by "rightward" blocks and are free of various collisions, and
- Without the bad collisions, adaptions always succeed.

The above intuition already restricted the event Hit (as well as other events originally introduced by Naito [13]) to lazily sampled blocks and resolved the aforementioned issues in bad events. For clarity, we employ the explicit bookkeeping method of [1], i.e., maintaining meta information (e.g., whether the record is defined via sampling) with the (simulated) blocks. In this way, bad events regarding lazily sampled records become unambiguous.

Adaptations due to encryption queries: right hand side. Regarding the right hand side of the to-be-adapted block, one may simply believe the involved $2n$-bit random oracle response H is uniform and thus collision events at this side are unlikely. Though, this is far from being precise: while the involved $2n$-bit random oracle response is uniform right after it's being sampled, the random oracle query may be made *before* this adapted simulator cycle.[5] More concretely, the distinguisher may first trigger the simulator to create a path $\left(0^{2n} \xrightarrow{M} S\right)$ (via issuing queries), and then query $\mathcal{RO}(M\|m) \to H$. After these preparations, the distinguisher then makes an appropriate encryption (or decryption) query to trigger the simulator completing the hash chain corresponding to $(M\|m, H)$. Similarly by symmetry, the distinguisher could prepare in a converse order, i.e., triggering creating the $\left(0^{2n} \xrightarrow{M} S\right)$ *after* querying $\mathcal{RO}(M\|x) \to H$. Since the query structure has been in the history *before* this simulator cycle, a bad event (the event HBCol in Sect. 5.4) has to be defined to exactly capture this situation, and a detailed case study is needed for this event. Moreover, the case that the pair of simulated ideal cipher query records share the same ciphertext (the abort condition $Y_1 = Y_2$ in line 4 of $\mathsf{CreateBlock}$) has to be in bad events as well, and this is captured by the event HCol in Sect. 5.4.

[5] In the weaker *sequential indifferentiability* setting [11], the distinguisher (roughly) cannot query the random oracle before querying the simulator, and the $2n$-bit response is indeed sampled in the current adapted cycle.

Adaptations due to decryption queries. Such adaptations enjoy another source of complexity. In detail, by inspecting the pseudocode of $\mathsf{S}^{\mathcal{RO}}_{E^{-1}}$, it can be seen that a crucial goal is to argue that the codes between lines 7–9 and 13–15 are executed only once upon every decryption query—as otherwise, the second time this part is executed, the corresponding call to $\mathsf{CreateBlock}(\star, \star, \perp)$ will find $Y \in \mathsf{ET}^{-1}[K]$ and abort. To elaborate on the concrete conditions, assume that the decryption query triggering this simulator cycle is $\mathsf{S}^{\mathcal{RO}}_{E^{-1}}(K, Y)$, and let's consider when the simulator would detect two distinct paths $\left(0^{2n} \xrightarrow{M} S\right)$ and $\left(0^{2n} \xrightarrow{M'} S'\right)$ in this simulator cycle. Then it can be seen:

1. It has to be $[S]_n = [S']_n$ which further equals $[K]_n$, as otherwise the two paths cannot be indexed in line 3 in $\mathsf{S}^{\mathcal{RO}}_{E^{-1}}$;
2. For the two random oracle queries $\mathcal{RO}(M\|[K]^\nu) \to H$ and $\mathcal{RO}(M'\|[K]^\nu) \to H'$, the path $\left(0^{2n} \xrightarrow{M} S\right)$ has to satisfy either $[H]^n \oplus Y = [S]^n \oplus \mathbf{2}_n$ or $[H]_n \oplus Y \oplus \mathbf{1}_n = [S]^n \oplus \mathbf{2}_n$, while the path $\left(0^{2n} \xrightarrow{M'} S'\right)$ has to satisfy either $[H']^n \oplus Y = [S']^n \oplus \mathbf{2}_n$ or $[H']_n \oplus Y \oplus \mathbf{1}_n = [S']^n \oplus \mathbf{2}_n$.

By rearranging, it can be seen the above is possible only if there exist two paths and two random oracle queries that satisfy some complicated cycle-like condition at some point during the G_1 execution. For example, if $\left(0^{2n} \xrightarrow{M} S\right)$ has $[H]^n \oplus Y = [S]^n \oplus \mathbf{2}_n$ and $\left(0^{2n} \xrightarrow{M'} S'\right)$ has $[H']^n \oplus Y = [S']^n \oplus \mathbf{2}_n$, then $[H]^n \oplus [S]^n = [H']^n \oplus [S']^n$. We formalize this as the event $\mathsf{PathLock}$ in Sect. 5.4.

5.3 Complexities of the Simulator

Consider D making q hash queries of σ message blocks and p primitive queries and running in time t. As long as the simulator doesn't abort, it can be seen:

- Upon each adversarial encryption query, the simulator detects at most 1 path (as otherwise it should have aborted due to line 2 in $\mathsf{S}^{\mathcal{RO}}_{E}(K, X)$), makes at most 1 query to $\mathcal{RO}$, and invokes $\mathsf{CreateBlock}$ once.
- Upon each adversarial decryption query, the simulator detects at most n paths (as otherwise it should have aborted due to line 2 in $\mathsf{S}^{\mathcal{RO}}_{E}(K, X)$), makes at most n queries to $\mathcal{RO}$, and invokes $\mathsf{CreateBlock}$ at most n times.

The running time of $\mathsf{CreateBlock}$ is roughly $O(1)$. Therefore, the simulator makes no more than np queries to $\mathcal{RO}$ in total and runs in time $t + O(np)$.

5.4 Bad Events in G_1 Executions

Size of the tables. Recall that the distinguisher making additional queries is considered in this section. In this respect, we first claim that $|\mathsf{HQueries}| \leq 2Q + np + q \leq 3Q + np$ during interaction with such distinguishers. For this, we make the following observations.

– The simulator makes at most 1 query to $\mathcal{RO}$ per adversarial encryption query. Since the number of adversarial encryption queries is at most $2Q$, such actions enlarge $|\mathsf{HQueries}|$ by at most $2Q$;
– The simulator makes at most n queries to $\mathcal{RO}$ per adversarial decryption query. Since the number of adversarial decryption queries cannot exceed p, such actions enlarge $|\mathsf{HQueries}|$ by at most np;
– Finally, D may directly make p queries to $\mathcal{RO}$ and increase $|\mathsf{HQueries}|$ by p.

We then claim $|\mathsf{ET}| \leq 2Q$ and $|\mathcal{L}_{\mathsf{block}}(\mathsf{ET})| \leq 2Q + 1$. For this, note that D makes at most Q non-redundant to $\mathsf{S}_E^{\mathcal{RO}}$ and $\mathsf{S}_{E^{-1}}^{\mathcal{RO}}$ in total. By inspecting the pseudocode, it can be seen:

– After processing every query to $\mathsf{S}_E^{\mathcal{RO}}$, $|\mathsf{ET}|$ increases by at most 2, and $|\mathcal{L}_{\mathsf{block}}(\mathsf{ET})|$ increases by at most 2. Note that these hold even if the simulator aborts in $\mathsf{S}_E^{\mathcal{RO}}$.
– After processing every query to $\mathsf{S}_{E^{-1}}^{\mathcal{RO}}$, $|\mathsf{ET}|$ increases by at most 2, and $|\mathcal{L}_{\mathsf{block}}(\mathsf{ET})|$ increases by at most 2.

Thus $|\mathsf{ET}| \leq 2Q$. Whereas $|\mathcal{L}_{\mathsf{block}}(\mathsf{ET})| \leq 2Q + 1$ since there is already one block $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$ at the beginning. Subsequent arguments typically distinguish between this initial block and the added ones.

Events on simulated ideal cipher queries. A table entry $\mathsf{ET}[K_1](X_1)$ is *lazily sampled*, if it is defined by line 20 in $\mathsf{S}_E^{\mathcal{RO}}$ or line 19 in $\mathsf{S}_{E^{-1}}^{\mathcal{RO}}$. We first put forward a collision event and a multi-collision event on such lazily sampled ideal cipher queries.

$$\begin{aligned}\mathsf{Hit0} \Leftrightarrow \exists \text{ a lazily sampled table entry } \mathsf{ET}[K](X) \text{ s.t. } \mathsf{ET}[K](X) \oplus X = 0^n,\\ \mathsf{MCol} \Leftrightarrow \exists \text{ distinct lazily sampled table entries } \mathsf{ET}[K_1](X_1), \ldots, \mathsf{ET}[K_n](X_n)\\ \text{s.t. } \mathsf{ET}[K_1](X_1) \oplus X_1 = \cdots = \mathsf{ET}[K_n](X_n) \oplus X_n.\end{aligned}$$

Events on blocks. The following events are defined with respect to $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$.

$$\begin{aligned}\mathsf{Hit} \Leftrightarrow \exists &\Big(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\Big), \Big(S^{(3)} \xrightarrow{M^*} S^{(4)}, d_2\Big) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET}) \text{ s.t.}\\ &d_1 = \rightarrow \wedge \Big(S^{(2)} = S^{(3)} \vee S^{(2)} = S^{(3)} \oplus (\mathbf{2}_n \| 0^n)\Big) \wedge\\ &\Big((S^{(3)} \xrightarrow{M^*} S^{(4)}) \text{ is defined before } (S^{(1)} \xrightarrow{M'} S^{(2)}) \text{ is defined}\\ &\vee (S^{(1)} \xrightarrow{M'} S^{(2)}) \text{ and } (S^{(3)} \xrightarrow{M^*} S^{(4)}) \text{ are defined}\\ &\qquad \text{in the same } \mathsf{CreateBlock} \text{ call}\Big)\\ \mathsf{DHit} \Leftrightarrow \exists &\Big(S^{(1)} \xrightarrow{M'} S^{(2)}, d\Big), \Big(S^{(3)} \xrightarrow{M^*} S^{(4)}, \leftarrow\Big) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET}) \text{ s.t. } d \neq \perp\\ &\Big(S^{(2)} = S^{(3)} \vee S^{(2)} = S^{(3)} \oplus (\mathbf{2}_n \| 0^n)\Big)\end{aligned}$$

$$\wedge \Big((S^{(1)} \xrightarrow{M'} S^{(2)}) \text{ is defined before } (S^{(3)} \xrightarrow{M^*} S^{(4)}) \text{ is defined} \Big)$$

$$\mathsf{BCol} \Leftrightarrow \exists \text{ distinct } \Big(S^{(1)} \xrightarrow{M'} S^{(2)}, \rightarrow\Big), \Big(S^{(3)} \xrightarrow{M^*} S^{(4)}, \rightarrow\Big) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET}) \text{ s.t.}$$
$$\Big(S^{(2)} = S^{(4)} \vee S^{(2)} = S^{(4)} \oplus (\mathbf{2}_n \| 0^n) \vee S^{(2)} = S^{(4)} \oplus (\mathbf{3}_n \| 0^n)\Big).$$

For convenience, define $\mathsf{BadBlock} := \mathsf{Hit0} \vee \mathsf{MCol} \vee \mathsf{Hit} \vee \mathsf{DHit} \vee \mathsf{BCol}$.

Events on paths. We next identify a subset $\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ of $\mathcal{L}_{\mathsf{path}}(\mathsf{ET})$, which consists of all the paths that are fully constituted by rightward blocks. We define two additional events on $\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$.

$$\mathsf{PathCol} \Leftrightarrow \exists \text{ distinct } \Big(0^{2n} \xrightarrow{M} S\Big), \Big(0^{2n} \xrightarrow{M'} S'\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET}) \text{ s.t.}$$
$$\Big(S = S' \vee S = S' \oplus (\mathbf{2}_n \| 0^n) \vee S = S' \oplus (\mathbf{3}_n \| 0^n)\Big),$$

$$\mathsf{PathMCol} \Leftrightarrow \exists n \text{ distinct paths } \Big(0^{2n} \xrightarrow{M_1} S_1\Big), \ldots, \Big(0^{2n} \xrightarrow{M_n} S_n\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET}) \text{ s.t.}$$
$$[S_1]_n = [S_2]_n = \cdots = [S_n]_n,$$

$$\mathsf{HBCol} \Leftrightarrow \exists (M\|m, H) \in \mathsf{HQueries},\ \Big(0^{2n} \xrightarrow{M} S\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET}), m \in \{0,1\}^{\nu} \text{ s.t.}$$
$$\Big([H]^n \oplus [S]^n \oplus \mathbf{2}_n \in \mathsf{ET}^{-1}\big[m\|[S]_n\big]$$
$$\bigvee\ [H]_n \oplus [S]^n \oplus \mathbf{3}_n \in \mathsf{ET}^{-1}\big[m\|[S]_n\big]\Big),$$

$$\mathsf{HCol} \Leftrightarrow \exists (M, H) \in \mathsf{HQueries} \text{ s.t. } [H]^n = [H]_n \oplus \mathbf{1}_n,$$

$$\mathsf{PathLock} \Leftrightarrow \exists \text{ distinct } \Big(0^{2n} \xrightarrow{M} S\Big), \Big(0^{2n} \xrightarrow{M'} S'\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET}),$$
$$m \in \{0,1\}^{\nu}, \big(M\|m, H\big), \big(M'\|m, H'\big) \in \mathsf{HQueries} \text{ s.t.}$$
$$\Big([S]_n = [S']_n\Big) \bigwedge \Big([H]_n \oplus \mathbf{1}_n \oplus [S]^n = [H']^n \oplus [S']^n\ \bigvee$$
$$[H]_n \oplus [S]^n = [H']_n \oplus [S']^n\ \bigvee\ [H]^n \oplus [S]^n = [H']^n \oplus [S']^n\Big).$$

Hereafter, these probabilities are upper-bounded.

Analyzing events on ET. Consider $\mathsf{Hit0}$ first. For any lazily sampled entry $\mathsf{ET}[K](X) = Y$, when it is defined, either X or Y is uniformly drawn from at least $2^n - 2Q$ values. Thus, the probability to have $X \oplus Y = 0^n$ does not exceed $1/(2^n - 2Q)$. Summing over the entries in ET, we have

$$\Pr\big[\mathsf{Hit0}\big] \leq \frac{2Q}{2^n - 2Q}. \tag{2}$$

For MCol, fix n distinct entries $\mathsf{ET}[K_1](X_1), \ldots, \mathsf{ET}[K_n](X_n)$. Since they are lazily sampled, for each i it holds either X_i or $\mathsf{ET}[K_i](X_i)$ is randomly drawn

from at least $2^n - 2Q$ values. Thus, the probability to have $\mathsf{ET}[K_1](X_1) \oplus X_1 = \cdots = \mathsf{ET}[K_n](X_n) \oplus X_n$ is at most $1/(2^n - 2Q)^{n-1}$. Hence, with the assumption $Q \le (2^n - 2Q)/4$ and $n \ge 3$, we have

$$\mathsf{Pr}[\mathsf{MCol}] \le \binom{2Q}{n} \cdot \left(\frac{1}{2^n - 2Q}\right)^{n-1} \le \frac{1}{n!} \cdot \left(\frac{4Q}{2^n - 2Q}\right)^n \le \frac{Q}{2^n - 2Q}. \quad (3)$$

Analyzing events on blocks. For Hit, fix two blocks $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\right)$ and $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, d_2\right)$ where these blocks are defined in the same $\mathsf{CreateBlock}$ call or the former block is defined after the latter. $S^{(2)}$ is defined as follows.

$$[S^{(2)}]^n = \mathsf{ET}\big[M' \| [S^{(1)}]_n\big]([S^{(1)}]^n) \oplus [S^{(1)}]^n,$$
$$[S^{(2)}]_n = \mathsf{ET}\big[M' \| [S^{(1)}]_n\big]([S^{(1)}]^n \oplus \mathbf{1}_n) \oplus [S^{(1)}]^n \oplus \mathbf{1}_n.$$

Since $d_1 = \rightarrow$, both $\mathsf{ET}\big[M' \| [S^{(1)}]_n\big]([S^{(1)}]^n)$ and $\mathsf{ET}\big[M' \| [S^{(1)}]_n\big]([S^{(1)}]^n \oplus \mathbf{1}_n)$ are randomly drawn from at least $2^n - 2Q$ values. Hence, the probability that $S^{(2)} = S^{(3)} \vee S^{(2)} = S^{(3)} \oplus (\mathbf{2}_n \| 0^n)$ is at most $2/(2^n - 2Q)^2$. Conditioned on $\neg\mathsf{Hit0}$, it necessarily be $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, d_2\right) \neq \left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$. Therefore, the number of choices for the two blocks is at most $2Q$, and thus

$$\Pr\big[\mathsf{Hit} \mid \neg\mathsf{Hit0}\big] \le (2Q)^2 \cdot \frac{2}{(2^n - 2Q)^2} \le \frac{8Q^2}{(2^n - 2Q)^2} \ . \quad (4)$$

For DHit, fix two blocks $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\right)$ and $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, \leftarrow\right)$ as desired by the condition. By construction, the leftward block was created due to D querying $\mathsf{S}^{\mathcal{RO}}_{E^{-1}}(M^* \| [S^{(3)}]_n, Y) \to X$ and then $[S^{(3)}]^n$ is set to either X or $X \oplus \mathbf{1}_n$. In either case, the derived $[S^{(3)}]^n$ is uniformly distributed in at least $2^n - 2Q$ possibilities. With these, we distinguish two cases as follows.

- Case 1: $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\right) = \left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$. In this case, the probability to have $S^{(3)} = S^{(2)} = 0^{2n}$ does not exceed $\Pr\big[[S^{(3)}]^n = 0^n\big] \le 1/(2^n - 2Q)$. Similarly, the probability to have $S^{(2)} = S^{(3)} \oplus (\mathbf{2}_n \| 0^n)$ does not exceed $\Pr\big[[S^{(3)}]^n = \mathbf{2}_n\big] \le 1/(2^n - 2Q)$. Summing over the at most $2Q$ leftward blocks, it can be seen the probability to have this Case 1 is at most $4Q/(2^n - 2Q)$.
- Case 2: $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\right) \neq \left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$. Then, since $d_1 \neq \perp$, its two corresponding cipher queries are lazily sampled. By this and $\neg\mathsf{MCol}$, the number of blocks $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, d_1\right)$ that have been defined such that $[S^{(2)}]_n = [S^{(3)}]_n$ is at most $n-1$. Therefore, the probability that DHit occurs due to the block $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, \leftarrow\right)$ is at most $2(n-1)/(2^n - 2Q)$. Summing over the at most $2Q$ blocks, it can be seen the probability to have this Case 1 is at most $4(n-1)Q/(2^n - 2Q)$.

Taking a union bound over the two cases yields

$$\mathsf{Pr}\big[\mathsf{DHit}|\neg\mathsf{MCol}\big] \leq \frac{4Q}{2^n-2Q} + \frac{4(n-1)Q}{2^n-2Q} = \frac{4nQ}{2^n-2Q}. \tag{5}$$

For BCol, fix two distinct blocks $\left(S^{(1)} \xrightarrow{M'} S^{(2)}, \rightarrow\right)$ and $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, \rightarrow\right)$. The following cases are considered.

- The first case is that $\left([S^{(1)}]^n = [S^{(3)}]^n \oplus \mathbf{1}_n\right) \wedge \left([S^{(1)}]_n = [S^{(3)}]_n\right) \wedge \left(M' = M^*\right)$, i.e., the blocks are defined by the same pair of queries. In this case, the condition $S^{(2)} = S^{(4)}$ implies $[S^{(2)}]^n = [S^{(2)}]_n$, i.e.,

  $$\mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n) \oplus [S^{(1)}]^n = \mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n \oplus \mathbf{1}_n) \oplus [S^{(1)}]^n \oplus \mathbf{1}_n \ .$$

  Since $\mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n)$ and $\mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n \oplus \mathbf{1}_n)$ are randomly drawn from at least $2^n - 2Q$ values, the probability to have $S^{(2)} = S^{(4)}$ is at most $1/(2^n - 2Q)$. On the other hand, it can be seen $S^{(2)} = S^{(4)} \oplus (\mathbf{2}_n \| 0^n)$ and $S^{(2)} = S^{(4)} \oplus (\mathbf{1}_n \| 0^n)$ are never fulfilled.
- The second case is that the blocks are defined by distinct pairs of queries. In this case, the condition $S^{(2)} = S^{(4)}$ implies

  $$\begin{aligned}\mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n) \oplus [S^{(1)}]^n &= \mathsf{ET}\big[M^*\|[S^{(3)}]_n\big]([S^{(3)}]^n) \oplus [S^{(3)}]^n \\ \mathsf{ET}\big[M'\|[S^{(1)}]_n\big]([S^{(1)}]^n \oplus \mathbf{1}_n) \oplus [S^{(1)}]^n \oplus \mathbf{1}_n & \\ = \mathsf{ET}\big[M^*\|[S^{(3)}]_n\big]([S^{(3)}]^n \oplus \mathbf{1}_n) \oplus [S^{(3)}]^n \oplus \mathbf{1}_n \ , &\end{aligned}$$

  where for $i = 1, 3$, $\mathsf{ET}\big[M'\|[S^{(i)}]_n\big]([S^{(i)}]^n)$ and $\mathsf{ET}\big[M'\|[S^{(i)}]_n\big]([S^{(i)}]^n \oplus \mathbf{1}_n)$ are randomly drawn from at least $2^n - 2Q$ values since the blocks are rightward. Moreover, these values are independently drawn, since the blocks are defined by distinct queries. Hence, the probability that $S^{(2)} = S^{(4)}$ is at most $1/(2^n - 2Q)^2$. Similarly, the probability that $S^{(2)} = S^{(4)} \oplus (\mathbf{2}_n \| 0^n)$ or $S^{(2)} = S^{(4)} \oplus (\mathbf{1}_n \| 0^n)$ is at most $2/(2^n - 2Q)^2$.

The number of chances for the first case is at most Q (to wit, $2Q$ blocks from Q pairs). For the second case, conditioned on $\neg\mathsf{Hit0}$, the involved blocks cannot be $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, d_1\right)$, and the number of choices is thus at most $\binom{2Q}{2}$. Hence, we have

$$\Pr\big[\mathsf{BCol} \mid \neg\mathsf{Hit0}\big] \leq \frac{Q}{2^n-2Q} + \binom{2Q}{2} \cdot \frac{3}{(2^n-2Q)^2} \leq \frac{Q}{2^n-2Q} + \frac{6Q^2}{(2^n-2Q)^2} \ . \tag{6}$$

Analyzing PathCol. We show that $\mathsf{PathCol}$ is not possible conditioned on $\neg\mathsf{BadBlock}$. Towards a contradiction, assume $\left(0^{2n} \xrightarrow{M} S\right), \left(0^{2n} \xrightarrow{M'} S'\right) \in$

$\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ with $M \neq M'$ and $S = S'$. If $M = \lambda$ then $S = 0^{2n}$ and it is not possible to have $S' = S = 0^{2n}$ by $\neg\mathsf{BadBlock}$, and vice versa. As such, assume that $M = h\|m\|t_1\|t_2\|\ldots\|t_\ell$ and $M' = h'\|m'\|t_1\|t_2\|\ldots\|t_\ell$, where $h, h' \in (\{0,1\}^\nu)^* \cup \{\lambda\}$, $m, m' \in \{0,1\}^\nu$, $t_1, t_2, \ldots, t_\ell \in \{0,1\}^\nu \cup \{\lambda\}$. Consider the last ℓ blocks

$$\Big(S_1^{\circ} \xrightarrow{t_1} S_2^{\circ}, \rightarrow\Big), \Big(S_2^{\circ} \xrightarrow{t_2} S_3^{\circ}, \rightarrow\Big), \ldots, \Big(S_\ell^{\circ} \xrightarrow{t_\ell} S_{\ell+1}^{\circ}, \rightarrow\Big);$$
$$\Big(S_1^{\circ\circ} \xrightarrow{t_1} S_2^{\circ\circ}, \rightarrow\Big), \Big(S_2^{\circ\circ} \xrightarrow{t_2} S_3^{\circ\circ}, \rightarrow\Big), \ldots, \Big(S_\ell^{\circ\circ} \xrightarrow{t_\ell} S_{\ell+1}^{\circ\circ} = S_{\ell+1}^{\circ}, \rightarrow\Big).$$

We distinguish two cases as follows.

- Case 1: there exists $1 \leq i \leq \ell$ such that $S_i^{\circ} \neq S_i^{\circ\circ}$, whereas $S_j^{\circ} = S_j^{\circ\circ}$ for all $j > i$. Then the existence of the two rightward blocks $\Big(S_i^{\circ} \xrightarrow{t_i} S_{i+1}^{\circ}, \rightarrow\Big)$ and $\Big(S_i^{\circ\circ} \xrightarrow{t_i} S_{i+1}^{\circ\circ}, \rightarrow\Big)$ indicates BCol and contradicts $\neg\mathsf{BadBlock}$;
- Case 2: $S_i^{\circ} = S_i^{\circ\circ}$ for all $1 \leq i \leq \ell$, i.e., the last ℓ blocks of the two paths are the same. Then the existence of the two rightward blocks $\Big(\star \xrightarrow{m} S_1^{\circ}, \rightarrow\Big)$ and $\Big(\star \xrightarrow{m'} S_1^{\circ\circ}, \rightarrow\Big)$ indicates BCol and contradicts $\neg\mathsf{BadBlock}$.

On the other hand, the existence of two paths $\exists\Big(0^{2n} \xrightarrow{M} S\Big), \Big(0^{2n} \xrightarrow{M'} S'\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ with $M \neq M'$ and $S = S' \oplus (\mathbf{2}_n\|0^n)$ ($S = S' \oplus (\mathbf{3}_n\|0^n)$, resp.) immediately indicates the existence of two rightward blocks $\Big(\star \xrightarrow{\star} S, \rightarrow\Big)$ and $\Big(\star \xrightarrow{\star} S', \rightarrow\Big)$ with $S = S' \oplus (\mathbf{2}_n\|0^n)$ ($S = S' \oplus (\mathbf{3}_n\|0^n)$, resp.) and contradicts $\neg\mathsf{BadBlock}$. By the above, we have $\mathsf{Pr}[\mathsf{PathCol}|\neg\mathsf{BadBlock}] = 0$.

Analyzing PathMCol. We also show that the event $\mathsf{PathMCol}$ is not possible conditioned on $\neg\mathsf{BadBlock}$ and $\neg\mathsf{PathCol}$. Consider any such n distinct paths $\Big(0^{2n} \xrightarrow{M_1} S_1\Big), \Big(0^{2n} \xrightarrow{M_2} S_2\Big), \ldots, \Big(0^{2n} \xrightarrow{M_n} S_n\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$.

First, note that $M_1, \ldots, M_n \neq \lambda$. Assume otherwise, and wlog assume $M_1 = \lambda$, then $S_1 = 0^{2n}$, and $M_2, \ldots, M_n \neq \lambda$. Let $\Big(T_2 \xrightarrow{m_2} S_2, \rightarrow\Big)$ be the last block of the path $\Big(0^{2n} \xrightarrow{M_2} S_2\Big)$. Then, the equality $[S_2]_n = 0^n$ implies $\mathsf{ET}[m_2\|[T_2]_n](X_2) \oplus X_2 = 0^n$ for $X_2 = [T_2]^n \oplus \mathbf{3}_n$. Since $\Big(0^{2n} \xrightarrow{M_2} S_2\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$, the entry $\mathsf{ET}[m_2\|[T_2]_n](X_2)$ is lazily sampled, and the above contradicts $\neg\mathsf{BadBlock}$ (more concretely, $\neg\mathsf{Hit0}$).

With the above, let $\Big(T_1 \xrightarrow{m_1} S_1, \rightarrow\Big), \ldots, \Big(T_n \xrightarrow{m_n} S_n, \rightarrow\Big)$ be the last blocks of the paths $\Big(0^{2n} \xrightarrow{M_1} S_1\Big), \Big(T_2 \xrightarrow{m_2} S_2, \rightarrow\Big), \ldots, \Big(0^{2n} \xrightarrow{M_n} S_n\Big)$ respectively. Conditioned on $\neg\mathsf{PathCol}$, we have $S_1 \neq S_2 \neq \cdots \neq S_n$. Then, the equality $[S_1]_n = [S_2]_n = \cdots = [S_n]_n$ implies

$$\mathsf{ET}\big[m_1\|[T_1]_n\big](X_1) \oplus X_1 = \mathsf{ET}\big[m_2\|[T_2]_n\big](X_2) \oplus X_2 = \cdots = \mathsf{ET}\big[m_n\|[T_n]_n\big](X_n) \oplus X_n$$

for $X_i = [T_i]^n \oplus \mathbf{3}_n$, $i = 1, \ldots, n$. This contradicts $\neg\mathsf{BadBlock}$ (more concretely, $\neg\mathsf{MCol}$). The above concludes $\mathsf{Pr}[\mathsf{PathMCol}|\neg\mathsf{BadBlock} \wedge \neg\mathsf{PathCol}] = 0$.

Analyzing HBCol and HCol. The event HBCol is divided into two sub-events as follows.

- ROHit: upon a random oracle query $\mathcal{RO}(M\|m) \to H$ is made (by either D or the simulator), there exists a path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}^{\to}_{\mathsf{path}}(\mathsf{ET})$ such that $[H]^n \oplus [S]^n \oplus \mathbf{2}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$ or $[H]_n \oplus [S]^n \oplus \mathbf{3}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$;
- $\mathsf{BlockHit}$: upon the simulator creating a rightward block $\left(S' \xrightarrow{m} S, \to\right)$, there exists a path $\left(0^{2n} \xrightarrow{M} S'\right) \in \mathcal{L}^{\to}_{\mathsf{path}}(\mathsf{ET})$ and a random oracle query $(M\|m\|m', H) \in \mathsf{HQueries}$ such that $[H]^n \oplus [S]^n \oplus \mathbf{2}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$ or $[H]_n \oplus [S]^n \oplus \mathbf{3}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$.

Below we analyze the probabilities in turn.

Probability of ROHit. Consider any random oracle query $\mathcal{RO}(M\|m) \to H$. Define a function

$$\mathsf{sval}\big(M\big) := s, \text{ where } s = [S]_n \text{ for } \left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}^{\to}_{\mathsf{path}}(\mathsf{ET}).$$

Then the probability to have $[H]^n \oplus [S]^n \oplus \mathbf{2}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$ is

$$\frac{\left|\mathsf{ET}^{-1}\big[m\|\mathsf{sval}\big(M\big)\big]\right|}{2^n},$$

since $[H]^n$ is uniform in $\{0,1\}^n$. Similarly, the probability to have $[H]_n \oplus [S]^n \oplus \mathbf{3}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$ is $\left|\mathsf{ET}^{-1}\big[m\|\mathsf{sval}\big(M\big)\big]\right|/2^n$. Conditioned on $\neg\mathsf{PathMCol}$, the number of paths $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}^{\to}_{\mathsf{path}}(\mathsf{ET})$ such that $[S]_n = s$ for any fixed $s \in \{0,1\}^n$ is at most n. By these, we have

$$\begin{aligned}
&\Pr[\mathsf{ROHit} \mid \neg\mathsf{BadBlock} \wedge \neg\mathsf{PathMCol}] \\
\leq\ & \sum\nolimits_{(M\|m,H)\in\mathsf{HQueries}} \frac{2\left|\mathsf{ET}^{-1}\big[m\|\mathsf{sval}\big(M\big)\big]\right|}{2^n} \\
\leq\ & \sum\nolimits_{\left(0^{2n}\xrightarrow{M}S\right)\in\mathcal{L}^{\to}_{\mathsf{path}}(\mathsf{ET})} \frac{2\left|\mathsf{ET}^{-1}\big[m\|[S]_n\big]\right|}{2^n} \\
\leq\ & n \cdot \sum\nolimits_{s\in\{0,1\}^n} \frac{2\left|\mathsf{ET}^{-1}\big[m\|s\big]\right|}{2^n} \\
\leq\ & \frac{4nQ}{2^n} \quad (\text{since } |\mathsf{ET}^{-1}| \leq 2Q). \qquad (7)
\end{aligned}$$

Probability of BlockHit. For this argument, define

$$\mathsf{HQueries}[M] := \left\{(M\|m, H) \in \mathsf{HQueries} \text{ for some } m \in \{0,1\}^{\nu}, H \in \{0,1\}^{2n}\right\}. \quad (8)$$

Consider any rightward block $\left(S' \xrightarrow{m} S, \rightarrow\right)$, where the underlying pair of ideal cipher queries are $\mathsf{ET}[m\|[S']_n]([S']^n) = Y = [S']^n \oplus [S]^n$ and $\mathsf{ET}[m\|[S']_n]([S']^n \oplus \mathbf{1}_n) = Y' = [S']^n \oplus \mathbf{1}_n \oplus [S]_n$. Conditioned on $\neg\mathsf{PathCol}$, there exists at most 1 path of the form $\left(0^{2n} \xrightarrow{M} S'\right)$. The number of random oracle queries of the form $(M\|m\|m', H)$, $m' \in \{0,1\}^{\nu}$, is $\left|\mathsf{HQueries}[M\|m]\right|$ (see Eq. (8)). Since $Y = [S']^n \oplus [S]^n$ and $Y' = [S']^n \oplus \mathbf{1}_n \oplus [S]_n$ are lazily sampled, the resulted $[S]^n$ and $[S]_n$ are uniform in at least $2^n - 2Q$ possibilities. By this, for any $(M\|m\|m', H) \in \mathsf{HQueries}[M\|m]$, the probability to have an entry $\mathsf{ET}[K](X) = Y$ such that $m\|[S]_n = K$ and $[H]^n \oplus [S]^n = Y$ is at most $2Q/(2^n - 2Q)^2$. Similarly, the probability to have $[H]_n \oplus [S]^n \oplus \mathbf{1}_n \in \mathsf{ET}^{-1}[m\|[S]_n]$ is at most $2Q/(2^n - 2Q)^2$.

Summing over all the rightward blocks, we have

$$\begin{aligned}&\Pr[\mathsf{BlockHit} \mid \neg\mathsf{BadBlock} \wedge \neg\mathsf{PathMCol}]\\ \le\ &\sum\nolimits_{\left(S' \xrightarrow{m} S, \rightarrow\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET}) : \exists \left(0^{2n} \xrightarrow{M} S'\right) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})} \frac{4Q\left|\mathsf{HQueries}[M\|m]\right|}{(2^n - 2Q)^2}. \end{aligned} \quad (9)$$

Note that for distinct blocks $\left(S_1 \xrightarrow{m_1} S_2, \rightarrow\right), \left(S_3 \xrightarrow{m_2} S_4, \rightarrow\right)$, the induced corresponding prefixes $M_1\|m_1$ and $M_2\|m_2$ are distinct: either $m_1 \neq m_2$ which immediately indicates the distinctness, or $S_1 \neq S_3$ indicating distinct corresponding paths $\left(0^{2n} \xrightarrow{M_1} S_1\right)$ and $\left(0^{2n} \xrightarrow{M_2} S_3\right)$ with $M_1 \neq M_2$. Therefore, using $\sum_{M \in \{0,1\}^*, m \in \{0,1\}^{\nu}} \left|\mathsf{HQueries}[M\|m]\right| = |\mathsf{HQueries}| \le 3Q + np$ (see Sect. 5.3), we obtain

$$\text{Eq. (9)} \le \sum\nolimits_{M \in \{0,1\}^*, m \in \{0,1\}^{\nu}} \left|\mathsf{HQueries}[M\|m]\right| \cdot \frac{4Q}{(2^n - 2Q)^2} \le \frac{12Q^2 + 4npQ}{(2^n - 2Q)^2}.$$

Summing over the above, we reach

$$\Pr[\mathsf{HBCol} | \neg\mathsf{BadBlock} \wedge \neg\mathsf{PathMCol}] \le \frac{4nQ}{2^n - 2Q} + \frac{12Q^2 + 4npQ}{(2^n - 2Q)^2}. \quad (10)$$

On the other hand, it is easy to see that

$$\Pr\left[\mathsf{HCol}\right] \le \frac{3Q + np}{2^n}. \quad (11)$$

Analyzing PathLock. Consider any pair of distinct paths $\left(0^{2n} \xrightarrow{M} S\right), \left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ and any $m \in \{0,1\}^{\nu}$ such that the two corresponding random oracle queries $\left(M\|m, H\right)$ and $\left(M'\|m, H'\right)$ indeed appeared during the interaction. To address the probabilities of the conditions, we distinguish several cases as follows.

Case 1: $M = \lambda$ or $M' = \lambda$. Note that they cannot both be λ since the two paths are distinct. Wlog consider the case of $M' = \lambda$ and $S' = 0^{2n}$. Let $\left(T \xrightarrow{m} S, d\right)$ be the last block of $\left(0^{2n} \xrightarrow{M} S\right)$. Since $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$, it holds $d = \rightarrow$, and there exists $K = m \| [T]_n, X = [T]^n$ such that $\mathsf{ET}[K](X) \oplus X = [S]^n$ and $\mathsf{ET}[K](X \oplus \mathbf{1}_n) \oplus X \oplus \mathbf{1}_n = [S]_n$, and that $\mathsf{ET}[K](X)$ and $\mathsf{ET}[K](X \oplus \mathbf{1}_n)$ are both lazily sampled. Then:

- The condition $[S]_n = [S']_n$ translates into $\mathsf{ET}[K](X \oplus \mathbf{1}_n) \oplus X \oplus \mathbf{1}_n = 0^n$, the probability of which is at most $1/(2^n - 2Q)$;
- The condition $[H]_n \oplus \mathbf{1}_n \oplus [S]^n = [H']^n \oplus [S']^n$ translates into $[H]_n \oplus [H']^n \oplus [S]^n = \mathbf{1}_n$ and further $[H]_n \oplus [H']^n \oplus \mathsf{ET}[K](X) \oplus X = \mathbf{1}_n$, the probability of which is at most $1/(2^n - 2Q)$ regardless of the order of sampling $[H]_n$, $[H']^n$, and $\mathsf{ET}[K](X)$ during the execution;
- Similarly, the probability to have $[H]_n \oplus [S]^n = [H']_n \oplus [S']^n$ is at most $1/(2^n - 2Q)$; the probability to have $[H]^n \oplus [S]^n = [H']^n \oplus [S']^n$ is at most $1/(2^n - 2Q)$.

In all, in this case, the probability to have the whole event $([S]_n = [S']_n) \wedge ([H]_n \oplus \mathbf{1}_n \oplus [S]^n = [H']^n \oplus [S']^n \vee [H]_n \oplus [S]^n = [H']_n \oplus [S']^n \vee [H]^n \oplus [S]^n = [H']^n \oplus [S']^n)$ is at most $3/(2^n - 2Q)^2$.

Case 2: $M, M' \neq \lambda$. Similarly to Case 1, there exists K, X, K', X' such that

$$\mathsf{ET}[K](X) \oplus X = [S]^n, \quad \mathsf{ET}[K](X \oplus \mathbf{1}_n) \oplus X \oplus \mathbf{1}_n = [S]_n,$$
$$\mathsf{ET}[K'](X') \oplus X' = [S']^n, \quad \mathsf{ET}[K'](X' \oplus \mathbf{1}_n) \oplus X' \oplus \mathbf{1}_n = [S']_n,$$

and that $\mathsf{ET}[K](X)$, $\mathsf{ET}[K](X \oplus \mathbf{1}_n)$, $\mathsf{ET}[K'](X')$, and $\mathsf{ET}[K'](X' \oplus \mathbf{1}_n)$ are all lazily sampled. Conditioned on $\neg\mathsf{PathCol}$, we have $S \neq S'$, and thus either $(K, X) \neq (K', X')$ or $(K, X \oplus \mathbf{1}_n) \neq (K', X' \oplus \mathbf{1}_n)$ (or both). Then:

- The condition $[S]_n = [S']_n$ translates into $\mathsf{ET}[K](X \oplus \mathbf{1}_n) \oplus X = \mathsf{ET}[K'](X' \oplus \mathbf{1}_n) \oplus X'$, the probability of which is always at most $1/(2^n - 2Q)$ regardless of the order of sampling $\mathsf{ET}[K](X \oplus \mathbf{1}_n)$ and $\mathsf{ET}[K'](X' \oplus \mathbf{1}_n)$ during the execution;
- The condition $[H]_n \oplus \mathbf{1}_n \oplus [S]^n = [H']^n \oplus [S']^n$ translates into $[H]_n \oplus \mathsf{ET}[K](X) \oplus X \oplus [H']^n \oplus \mathsf{ET}[K'](X') \oplus X' = \mathbf{1}_n$, the probability of which is always at most $1/(2^n - 2Q)$ regardless of the order of sampling $[H]_n$, $[H']_n$, $\mathsf{ET}[K](X)$, and $\mathsf{ET}[K'](X')$ during the execution. Similarly, the probability to have the condition $[H]_n \oplus [S]^n = [H']_n \oplus [S']^n$ is always at most $1/(2^n - 2Q)$; the probability to have the condition $[H]^n \oplus [S]^n = [H']^n \oplus [S']^n$ is always at most $1/(2^n - 2Q)$.

In all, in this case, the probability to have the whole event is at most $3/(2^n - 2Q)^2$.

Summarizing. We conclude by counting the number of choices as above. For any $\left(0^{2n} \xrightarrow{M} S\right), \left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$, the number of random oracle queries

of the form $\left(M\|m,H\right)$ appeared during the execution is $\left|\mathsf{HQueries}[M]\right|$, and the number of queries $\left(M'\|m,H'\right)$ is at most $\left|\mathsf{HQueries}[M']\right|$. For every such four queries, the probability to have the whole event is at most $3/(2^n-2Q)^2$ as argued. Therefore,

$$\begin{aligned}
&\mathsf{Pr}[\mathsf{PathLock}|\neg\mathsf{BadBlock}\wedge\neg\mathsf{PathCol}]\\
\leq\ &\sum_{\left(0^{2n}\xrightarrow{M}S\right)\in\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})}\ \sum_{\left(0^{2n}\xrightarrow{M'}S'\right)\in\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})}\frac{3\left|\mathsf{HQueries}[M]\right|\cdot\left|\mathsf{HQueries}[M']\right|}{(2^n-2Q)^2}\\
=\ &\sum_{\left(0^{2n}\xrightarrow{M}S\right)\in\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})}\left|\mathsf{HQueries}[M]\right|\sum_{\left(0^{2n}\xrightarrow{M'}S'\right)\in\mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})}\left|\mathsf{HQueries}[M']\right|\cdot\frac{3}{(2^n-2Q)^2}\\
\leq\ &\frac{3(3Q+np)^2}{(2^n-2Q)^2}. \qquad (12)
\end{aligned}$$

A G_1 execution is *good*, if none of the bad events occurred during this execution. Summing over the above concludes the probability to observe good G_1 executions.

Lemma 1. *The probability to have a bad G_1 execution is at most*

$$\frac{6(3Q+np)^2}{(2^n-2Q)^2}+\frac{11nQ+np}{2^n-2Q}.$$

Proof. Gathering Eqs. (2), (3), (4), (5), (6), (10), (11), and (12) yields

$$\begin{aligned}
&\frac{2Q}{2^n-2Q}+\frac{Q}{2^n-2Q}+\frac{8Q^2}{(2^n-2Q)^2}+\frac{4nQ}{2^n-2Q}+\frac{Q}{2^n-2Q}+\frac{6Q^2}{(2^n-2Q)^2}\\
&+\underbrace{\frac{4nQ}{2^n}}_{\leq\frac{4nQ}{2^n-2Q}}+\frac{12Q^2+4npQ}{(2^n-2Q)^2}+\underbrace{\frac{3Q+np}{2^n}}_{\leq\frac{3Q+np}{2^n-2Q}}+\frac{3(3Q+np)^2}{(2^n-2Q)^2}\\
\leq\ &\underbrace{\frac{26Q^2+4npQ}{(2^n-2Q)^2}}_{\leq\frac{3(3Q+np)^2}{(2^n-2Q)^2}}+\underbrace{\frac{8nQ+7Q+np}{2^n-2Q}}_{\leq\frac{8nQ+3nQ+np}{2^n-2Q}}+\frac{3(3Q+np)^2}{(2^n-2Q)^2}\\
\leq\ &\frac{6(3Q+np)^2}{(2^n-2Q)^2}+\frac{11nQ+np}{2^n-2Q},
\end{aligned}$$

as claimed. □

5.5 Consistency of good G_1 executions

As mentioned, we next establish consistency of the simulation. Formally, in Game 1, for any hash query M, the response $\mathcal{RO}(M)$ is equal to $\mathsf{MDPH}^{\mathsf{S}^{\mathcal{RO}}_E}$, and the simulator never aborts due to adaptations. To this end, we show the following (crucial) lemma regarding paths.

Lemma 2. *In a good* G_1 *execution, it holds* $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$*, i.e., all paths are fully constituted by rightward blocks.*

Proof. The claim $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ clearly holds before the first simulator cycle, since there was only 1 path $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}\right)$ due to the initial block $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$. Now consider the j-th simulator cycle, and assume that $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ holds before this cycle. Depending on the type of the cycle, we distinguish three cases.

Case 1: the j-th cycle is rightward. Assume that the block created in this cycle is $\left(S^{(1)} \xrightarrow{M} S^{(2)}, \rightarrow\right)$. Clearly, this block may "extend" existing paths. Though, if $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) \neq \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ after creating this block, then there necessarily exists another block $\left(S^{(3)} \xrightarrow{M'} S^{(4)}, d_2\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ with $d_2 = \leftarrow$ or $\perp$ before this cycle, and the newly created block fulfills $S^{(2)} = S^{(3)}$. This contradicts $\neg\mathsf{Hit}$, meaning that $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ *does hold* after this cycle.

Case 2: the j-th cycle is leftward. In this case, if $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) \neq \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ after this cycle, then for the leftward block $\left(S^{(3)} \xrightarrow{M^*} S^{(4)}, \leftarrow\right)$ created in this cycle, there necessarily exists another block $\left(S^{(1)} \xrightarrow{M} S^{(2)}, \rightarrow\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ such that

- $\left(S^{(1)} \xrightarrow{M} S^{(2)}, \rightarrow\right)$ was the last block of a path that already existed before this simulator cycle (it might be $\left(0^{2n} \xrightarrow{\lambda} 0^{2n}, \rightarrow\right)$), and
- $S^{(2)} = S^{(3)}$.

This contradicts $\neg\mathsf{DHit}$, showing that $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ *does hold* after this cycle.

Case 3: the j-th cycle is adapted. In this case, we consider the detailed steps of the cycle depending on the direction of the adversarial query. Ignoring the possibilities of abortions, we note that:

- If the cycle was triggered by D querying $\mathsf{S}_E^{\mathcal{R}\mathcal{O}}$, then the simulator creates *at most* 1 pair of adapted blocks;
- If the cycle was triggered by D querying $\mathsf{S}_{E^{-1}}^{\mathcal{R}\mathcal{O}}$, then the simulator creates *at most* 1 pair of adapted blocks for each detected path.

Regardless of the details, let $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ be the first path detected during this cycle, and let $(M\|m, H)$ be the corresponding random oracle query. The pair of adapted blocks to-be-created for this path would be $\left(S \oplus (\mathbf{2}_n\|0^n) \xrightarrow{m}\right.$

$H, \bot\Big)$ and $\Big(S \oplus (\mathbf{3}_n \| 0^n) \xrightarrow{m} \overline{H}, \bot\Big)$. Since $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ before this simulator cycle, it holds $\Big(0^{2n} \xrightarrow{M} S\Big) \in \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$. By this and by $\neg\mathsf{BadBlock}$, there does not exist any path of the form $\Big(0^{2n} \xrightarrow{\star} S \oplus (\mathbf{2}_n \| 0^n)\Big)$ or $\Big(0^{2n} \xrightarrow{\star} S \oplus (\mathbf{3}_n \| 0^n)\Big)$ before this simulator cycle (this in particular includes $\Big(0^{2n} \xrightarrow{\lambda} 0^{2n}\Big)$). By this, if successfully created, the two adapted blocks won't be "appended" to any existing paths, and thus $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ holds after creating this pair of adapted blocks. Applying this argument iteratively, it can be seen $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ holds after the simulator creates all adapted blocks and completes this simulator cycle. Note that the conclusion holds even if this cycle aborts when trying to create the adapted blocks. □

With the help of Lemma 2, we are able to show that various types of abortions cannot occur. We first consider those due to maintaining the table PathT.

Lemma 3. *In a good* G_1 *execution, the simulator never aborts due to lines 15 and 20 in* $\mathsf{CreateBlock}$ *(i.e., maintaining the table* PathT*).*

Proof. By construction, the simulator aborts due to lines 15 or 20 only if distinct paths $\Big(0^{2n} \xrightarrow{M} S\Big)$ and $\Big(0^{2n} \xrightarrow{M'} S\Big)$ with identical endpoint S are encountered. Since the execution is good, it holds $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ by Lemma 2. Then, such two colliding paths cannot exist by $\neg\mathsf{PathCol}$, and thus the claim. □

We then address the abortions due to adaptations.

Lemma 4. *In a good* G_1 *execution, the simulator never aborts in adapted simulator cycles.*

Proof. Consider the j-th adapted simulator cycle, and consider the detailed steps of the cycle depending on the direction of the adversarial query.

Case 1: the simulator cycle was triggered by D querying $\mathsf{S}^{\mathcal{R},\mathcal{O}}_{E}$. Then the simulator would first check the abort condition at line 2. By Lemma 2, it holds $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ right before this cycle. Thus, this condition cannot be fulfilled by $\neg\mathsf{PathCol}$. Since we assumed that the current simulator cycle is an adapted cycle, either the branch in lines 7–10 or the branch in lines 14–17 is executed. Regardless of which branch is executed, let $\Big(0^{2n} \xrightarrow{M} S\Big) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ be the path detected during this cycle, and let $(M \| m, H)$ be the corresponding random oracle query. The simulator would then try to create two blocks $\Big(S \oplus (\mathbf{2}_n \| 0^n) \xrightarrow{m} H, \bot\Big)$ and $\Big(S \oplus (\mathbf{3}_n \| 0^n) \xrightarrow{m} \overline{H}, \bot\Big)$, via making a call to $\mathsf{CreateBlock}\big((K, X, Y_1), (K, X \oplus \mathbf{1}_n, Y_2), \bot\big)$ with $K = m \| [S]_n$, $X = [S]^n \oplus \mathbf{2}_n$, $Y_1 = [H]^n \oplus X$ and $Y_2 = [H]_n \oplus X \oplus \mathbf{1}_n$.

We next step into the call to $\mathsf{CreateBlock}\big((K, X, Y_1), (K, X \oplus \mathbf{1}_n, Y_2), \bot\big)$. By the pseudocode, this call would check if the to-be-defined adapted entry causes inconsistency at line 3. On the input side, it necessarily holds $X \notin \mathsf{ET}[K]$ and $X \oplus \mathbf{1}_n \notin \mathsf{ET}[K]$:

– they held before this cycle (otherwise this cycle won't happen at all), and
– the only possibility for earlier actions in this cycle to define $\mathsf{ET}[K](X)$ and $\mathsf{ET}[K](X \oplus \mathbf{1}_n)$ is that, the branch in lines 7–10 is executed and then the branch in lines 14–17 is executed. But this, as we argued before, is not possible.

By the above, line 3 won't cause abort.

On the output side, we argue that it also holds $Y_1 \notin \mathsf{ET}^{-1}[K]$, $Y_2 \notin \mathsf{ET}^{-1}[K]$, and $Y_1 \neq Y_2$. Recall that $(M\|m, H) \in \mathsf{HQueries}$ is the corresponding random oracle query. We begin by arguing $Y_1 \notin \mathsf{ET}^{-1}[K]$ and $Y_2 \notin \mathsf{ET}^{-1}[K]$ before this simulator cycle. First, note that right after $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ and $(M\|m, H) \in \mathsf{HQueries}$ both hold, it holds $Y_1 \notin \mathsf{ET}^{-1}[K]$ and $Y_2 \notin \mathsf{ET}^{-1}[K]$ by $\neg\mathsf{HBCol}$. During the period between this point and the beginning of the current simulator cycle, the only possibilities to render $Y_1 \in \mathsf{ET}^{-1}[K]$ or $Y_2 \in \mathsf{ET}^{-1}[K]$ are:

– D queries $\mathsf{S}_E^{\mathcal{R}\mathcal{O}}([S]_n\|m, X)$ or $\mathsf{S}_E^{\mathcal{R}\mathcal{O}}([S]_n\|m, X \oplus \mathbf{1}_n)$, forcing the simulator to detect the path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ and define $\mathsf{ET}^{-1}[K](Y_1)$ and $\mathsf{ET}^{-1}[K](Y_2)$. But in this case, after S completes this purported detection, it already holds $\mathsf{ET}^{-1}[K](Y_1) = X$ and $\mathsf{ET}^{-1}[K](Y_2) = X \oplus \mathbf{1}_n$ which is consistent with the computations. And later when D makes its j-th query (to $\mathsf{S}_E^{\mathcal{R}\mathcal{O}}$), it holds $X, X \oplus \mathbf{1}_n \in \mathsf{ET}[K]$ and the purported detection and adaption wouldn't have happened, which contradicts.
– D queries $\mathsf{S}_{E^{-1}}^{\mathcal{R}\mathcal{O}}([S]_n\|m, Y_1)$ or $\mathsf{S}_{E^{-1}}^{\mathcal{R}\mathcal{O}}([S]_n\|m, Y_2)$, forcing the simulator to detect the path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$. Since this previous adapted simulator cycle was completed without abortion, it already holds $\mathsf{ET}^{-1}[K](Y_1) = X$ and $\mathsf{ET}^{-1}[K](Y_2) = X \oplus \mathbf{1}_n$ after that cycle and the purported detection in this j-th cycle wouldn't have happened.

Finally, as $Y_1 = [H]^n \oplus X$ and $Y_2 = [H]_n \oplus X \oplus \mathbf{1}_n$, the condition $Y_1 = Y_2$ translates into $[H]^n = [H]_n \oplus \mathbf{1}_n$, which isn't possible conditioned on $\neg\mathsf{HCol}$. By Lemma 3, the remaining abortions in $\mathsf{CreateBlock}\big((K, X, Y_1), (K, X \oplus \mathbf{1}_n, Y_2), \bot\big)$ won't happen either. We thus conclude this adapted cycle won't cause abort.

Case 2: the simulator cycle was triggered by D querying $\mathsf{S}_{E^{-1}}^{\mathcal{R}\mathcal{O}}$. Assume that the adversarial decryption query is $\mathsf{S}_{E^{-1}}^{\mathcal{R}\mathcal{O}}(K, Y)$. Since the execution is good, it holds $\mathcal{L}_{\mathsf{path}}(\mathsf{ET}) = \mathcal{L}^{\rightarrow}_{\mathsf{path}}(\mathsf{ET})$ by Lemma 2. Then, line 2 would not cause abortion, as otherwise it indicates the occurrence of $\mathsf{PathMCol}$ and contradicts.

It remains to argue that the codes between lines 7–9 and 13–15 is only executed once—as otherwise, the second time this part is executed, the corresponding call to $\mathsf{CreateBlock}(\star, \star, \bot)$ will find $Y \in \mathsf{ET}^{-1}[K]$ and abort. By construction, it can be seen that if the detection conditions at line 6 and 12 are fulfilled twice or more (for several paths and random oracle queries), then the event $\mathsf{PathLock}$ occurred before this j-th simulator cycle, contradicting our assumption. By this,

the codes between lines 7–9 and 13–15 is only executed once, i.e., the simulator detects only 1 path in this j-th adapted cycle. The remaining argument for the non-abortion of the subsequent call to $\mathsf{CreateBlock}(\star, \star, \perp)$ is similar to the above argument in Case 1. By these, in this case, this j-th adapted cycle won't abort either. □

5.6 From Consistency to Indistinguishability

As mentioned, this section presents the reduction from indistinguishability to non-abortion. To this end, we focus on a fixed and deterministic distinguisher D rather than an arbitrary one, since the advantage of a probabilistic distinguisher cannot exceed the corresponding deterministic version with the best random coins. With respect to D, we introduce some terminology first.

Recall that a G_1 execution is *good* if none of the bad events was observed. Whether a certain G_1 execution is good depends on the randomness ϵ used by the simulator and the random oracle $\mathcal{RO}$, and we write $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ for the G_1 execution that uses the (explicit) randomness $(\epsilon, \mathcal{RO})$. For a good G_1 execution, consider the tables $(\mathsf{ET}, \mathsf{ET}^{-1})$ of the simulator standing at the end of D^{G_1}. These tables essentially reflect the "footprints" of the execution. By Lemma 4, good G_1 executions never aborts. By the pseudocode, in non-aborting G_1 executions, information kept in the two tables are actually the same, and thus we only consider the table ET in the subsequent argument. Denote by $\mathcal{T}_{\mathsf{e}}$ the set of all possible tables ET that can be generated by the simulator during good executions. For randomness $(\epsilon, \mathcal{RO})$, if the table ET standing at the end of $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ has exactly the same entries as $\mathsf{ET}^{\circ} \in \mathcal{T}_{\mathsf{e}}$, then we write $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET}^{\circ}$.

Now, consider a table $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$. For an ideal cipher E, if for any $K \in \{0,1\}^k$ and any $X \in \{0,1\}^n$ such that $\mathsf{ET}[K](X) \neq \perp$ it holds $E(K, X) = \mathsf{ET}[K](X)$, then we say that E *extends* ET and denote $E \vdash \mathsf{ET}$.

Then, we have the following lemma.

Lemma 5. *At the end of a good* G_1 *execution, for every random oracle query* $(M\|m, H) \in \mathsf{HQueries}$*,* $M \in (\{0,1\}^{\nu})^*$*,* $m \in \{0,1\}^{\nu}$*, there exist* $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ *and* $\left(S^* \xrightarrow{m} H, \perp\right) \in \mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ *such that* $S = S^* \oplus (\mathbf{2}_n \| 0^n)$*.*

Proof. By construction, when $\mathsf{S}^{\mathcal{RO}}$ makes a random oracle query $\mathcal{RO}(M\|m) \to H$, it is to complete the corresponding hash chain, and the aforementioned path and block must be in $\mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ and $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$ respectively. On the other hand, if the query $\mathcal{RO}(M\|m) \to H$ is made by D, $M = m'_1\| \ldots \|m'_{\ell}$, then by our assumption, D will eventually issue queries to $\mathsf{S}_E^{\mathcal{RO}}$ to evaluate as defined by MDPH. When D issues the last pair of encryption queries to $\mathsf{S}_E^{\mathcal{RO}}$, $\mathsf{S}_E^{\mathcal{RO}}$ will detect the path and make a call to $\mathsf{CreateBlock}(\star, \star, \perp)$. After this, the aforementioned path and block must be in $\mathcal{L}_{\mathsf{path}}(\mathsf{ET})$ and $\mathcal{L}_{\mathsf{block}}(\mathsf{ET})$. □

Then, we have: the G_1 and G_2 executions that are "linked" by the tables of the simulator behave the same in the view of D, and are indistinguishable.

Lemma 6. *Consider a pair of randomness $(\epsilon, \mathcal{RO})$ such that $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$. Then for any block cipher $E \vdash \mathsf{ET}$, the transcripts of queries and answers of D in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ and $D^{\mathsf{G}_2(E)}$ are the same, and $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = D^{\mathsf{G}_2(E)}$.*

Proof. The idea is that the random values used during the three executions are consistent. See Appendix A.1 for the (somewhat standard) proof. □

For any $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$, the probabilities of the following two events are close:

1. a G_1 execution with a random tuple $(\epsilon, \mathcal{RO})$ generates ET;
2. $E \vdash \mathsf{ET}$ for an ideal cipher E.

Lemma 7. *With respect to a fixed distinguisher D that makes q hash queries of σ message blocks in total and p primitive queries, for any $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$, it holds*

$$\frac{\Pr_E[E \vdash \mathsf{ET}]}{\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET}]} \geq 1.$$

Proof. See Appendix A.3. □

Then the following lemma completes the indistinguishability of G_1 and G_2.

Lemma 8. *For any distinguisher D making q hash queries of σ message blocks in total and p primitive queries, when $Q = \sigma + p \leq 2^n/2$, it holds*

$$\left|\Pr\left[D^{\mathsf{G}_1} = 1\right] - \Pr\left[D^{\mathsf{G}_2} = 1\right]\right| \leq \frac{6(3Q + np)^2}{(2^n - 2Q)^2} + \frac{11nQ + np}{2^n - 2Q}.$$

Proof. The full proof is deferred to Appendix A.4. □

6 Conclusions

This article has shown a gap in the indifferentiability analysis of a hash function proposed by Naito [13], and presented a patch for the proof. As a result, MDPH maintains the original security bound, namely $O(n - \log n)$-bit indifferentiability security from the random oracle. The gap was about the behaviour of the decryption simulator, and we corrected it by clarifying what to be kept for consistency in responding to a decryption query.

Acknowledgments. The authors thank Yusuke Naito and Jooyoung Lee for helpful comments and discussions. We also thank Thomas Peyrin and Mustafa Khairallah for discussions. Chun Guo was partly supported by the Program of Taishan Young Scholars of the Shandong Province, the Program of Qilu Young Scholars (Grant No. 61580089963177) of Shandong University, the National Natural Science Foundation of China (Grant No. 62002202), and the Shandong Nature Science Foundation of China (Grant No. ZR2020MF053).

References

1. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indifferentiability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 531–550. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_29
2. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53008-5_5
3. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.X.: TEDT: a leakage-resistant AEAD mode. IACR TCHES **2020**(1), 256–320 (2019). https://doi.org/10.13154/tches.v2020.i1.256-320, `https://tches.iacr.org/index.php/TCHES/article/view/8400`
4. Coron, J.S., Holenstein, T., Künzler, R., Patarin, J., Seurin, Y., Tessaro, S.: How to build an ideal cipher: The indifferentiability of the Feistel construction. Journal of Cryptology **29**(1), 61–114 (Jan 2016). https://doi.org/10.1007/s00145-014-9189-6
5. Guo, C., Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus v1.3. Submission to NIST Lightweight Cryptography Project (2021), `https://csrc.nist.gov/Projects/lightweight-cryptography/`
6. Guo, C., Lin, D.: Separating invertible key derivations from non-invertible ones: sequential indifferentiability of 3-round even-mansour. Des. Codes Cryptogr. **81**(1), 109–129 (2016). https://doi.org/10.1007/s10623-015-0132-0, `https://doi.org/10.1007/s10623-015-0132-0`
7. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg, Germany, Graz, Austria (Mar 15–17, 2006). https://doi.org/10.1007/11799313_14
8. Hirose, S., Park, J.H., Yun, A.: A simple variant of the Merkle-Damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg, Germany, Kuching, Malaysia (Dec 2–6, 2007). https://doi.org/10.1007/978-3-540-76900-2_7
9. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the titans: The romulus and remus families of lightweight AEAD algorithms. IACR Trans. Symmetric Cryptol. **2020**(1), 43–120 (2020). https://doi.org/10.13154/tosc.v2020.i1.43-120, `https://doi.org/10.13154/tosc.v2020.i1.43-120`
10. List, E.: TEDT2 - highly secure leakage-resilient tbc-based authenticated encryption. In: LATINCRYPT. Lecture Notes in Computer Science, vol. 12912, pp. 275–295. Springer (2021)
11. Mandal, A., Patarin, J., Seurin, Y.: On the public indifferentiability and correlation intractability of the 6-round Feistel construction. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 285–302. Springer, Heidelberg, Germany, Taormina, Sicily, Italy (Mar 19–21, 2012). https://doi.org/10.1007/978-3-642-28914-9_16
12. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg, Germany, Cambridge, MA, USA (Feb 19–21, 2004). https://doi.org/10.1007/978-3-540-24638-1_2

13. Naito, Y.: Optimally indifferentiable double-block-length hashing without post-processing and with support for longer key than single block. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 65–85. Springer, Heidelberg, Germany (2019). https://doi.org/10.1007/978-3-030-30530-7_4
14. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011). https://doi.org/10.1007/978-3-642-20465-4_27

A Deferred Proofs for Randomness Mapping

A.1 Proof of Lemma 6

By an induction, assume that the transcripts obtained by D are the same up to some point in the two executions, and consider the next query of D. Since D is deterministic, the next query in the two executions are the same. We argue that the answers obtained in the two executions are the same as well. Depending on the type of this query, we distinguish two cases:

1. the query is to S_E/E or $\mathsf{S}_{E^{-1}}/E^{-1}$: the answer obtained in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ are consistent with the values in ET. Then, since $E \vdash \mathsf{ET}$, the answer obtained in $D^{\mathsf{G}_2(E)}$, which is provided by E, is the same as that in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$;
2. the query is to the hash: then the answer obtained in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ is provided by $\mathcal{RO}$. By Lemma 5, this answer is consistent with the hash value $\mathsf{MDPH}^{\mathsf{ET}}$. Then, since $E \vdash \mathsf{ET}$, the answer MDPH^{E} obtained in $D^{\mathsf{G}_2(E)}$ is also the same as $\mathsf{MDPH}^{\mathsf{ET}}$ in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$.

Therefore, the transcripts of D in the two executions are the same. Since D is deterministic, the two outputs of D are also the same.

A.2 Two Additional Helper Lemmas

We will rely on a useful inequality. It uses a new notation Θ_1, which is based on a corollary of Lemma 6. For this, consider a table $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$, and assume that the following holds for a pair of randomness $(\epsilon, \mathcal{RO})$:

- $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET}$;
- D outputs 1 in $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$, say, $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = 1$.

Then by Lemma 6, for any $(\epsilon', \mathcal{RO}')$, once $D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')} \to \mathsf{ET}$, $D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')} = 1$— to this end, consider a block cipher $E \vdash \mathsf{ET}$, then $1 = D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = D^{\mathsf{G}_2(E)} = D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')}$. With this in mind, we denote by Θ_1 the subset of $\mathcal{T}_{\mathsf{e}}$ such that for any $(\epsilon, \mathcal{RO})$ such that $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET} \in \Theta_1$, it holds $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = 1$.

Lemma 9. $\Pr_E[D^{\mathsf{G}_2(E)} = 1] \geq \sum_{\mathsf{ET}\in\Theta_1} \Pr_E[E \vdash \mathsf{ET}]$.

Proof. We show that for any E, there exists at most one $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$ such that $E \vdash \mathsf{ET}$. Assume otherwise, i.e. $\exists \mathsf{ET}' \in \mathcal{T}_{\mathsf{e}}$ such that $\mathsf{ET} \neq \mathsf{ET}' \wedge E \vdash \mathsf{ET} \wedge E \vdash \mathsf{ET}'$. Assume that for two good pairs $(\epsilon, \mathcal{RO})$ and $(\epsilon', \mathcal{RO}')$, it holds $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \rightarrow \mathsf{ET}$ and $D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')} \rightarrow \mathsf{ET}'$. Then, consider any query of the combination (D, S) in the two executions $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ and $D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')}$: (i) the answers to the query to $\mathsf{S}^{\mathcal{RO}}(\epsilon)/\mathsf{S}^{\mathcal{RO}'}(\epsilon')$ are the same, since they both equal the values eventually kept in ET/ET', which further equal the values defined by E; (ii) the answers to the query to $\mathcal{RO}/\mathcal{RO}'$ are also the same, since they equal $\mathsf{MDPH}^{\mathsf{ET}}/\mathsf{MDPH}^{\mathsf{ET}'}$ respectively (by Lemma 5), and $\mathsf{MDPH}^{\mathsf{ET}} = \mathsf{MDPH}^{E} = \mathsf{MDPH}^{\mathsf{ET}'}$. Then, following the same line as the proof of Lemma 6, we have that the transcripts of the combination (D, S) in the two executions $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ and $D^{\mathsf{G}_1(\epsilon',\mathcal{RO}')}$ are the same, so that the two tables ET and ET' should be the same, a contradiction. After this, we have

$$\begin{aligned}\Pr_E[D^{\mathsf{G}_2(E)} = 1] &\geq \Pr_E[D^{\mathsf{G}_2(E)} = 1 \wedge \exists \mathsf{ET} \in \mathcal{T}_{\mathsf{e}} \text{ s.t. } E \vdash \mathsf{ET}] \\ &= \sum_{\mathsf{ET} \in \Theta_1} \Pr_E[E \vdash \mathsf{ET}] \qquad \text{(by Lemma 6)},\end{aligned}$$

as claimed. □

We will rely on another helper lemma.

Lemma 10. *Consider a good execution D^{G_1} in which the distinguisher evaluates all hash queries (as we assumed). Then, at the end of the execution, the number of calls to $\mathsf{CreateBlock}(\star, \star, \perp)$ by the simulator equals the size of the set $\mathsf{HQueries}$ (i.e., the total number of distinct random oracle queries occurred during the execution).*

Proof. For every call to $\mathsf{CreateBlock}(\star, \star, \perp)$, there is a corresponding random oracle query record in $\mathsf{HQueries}$, which was just made inside the call. Furthermore, for no other call to $\mathsf{CreateBlock}(\star, \star, \perp)$ the same random oracle query was made, as otherwise the later call to $\mathsf{CreateBlock}(\star, \star, \perp)$ would have caused abort, contradicting Lemma 4.

On the other hand, for each record in $\mathsf{HQueries}$ there was a corresponding call to $\mathsf{CreateBlock}(\star, \star, \perp)$, as shown in Lemma 5. □

A.3 Proof of Lemma 7

Fix $\mathsf{ET} \in \mathcal{T}_{\mathsf{e}}$. Then clearly

$$\Pr_E[E \vdash \mathsf{ET}] = \prod_{K \in \{0,1\}^k} \frac{1}{(2^n)_{|\mathsf{ET}[K]|}}. \tag{13}$$

For $\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \rightarrow \mathsf{ET}]$, we consider the table $\mathsf{ET}[K]$ for an arbitrary key $K \in \{0,1\}^k$, and list the entries $\mathsf{ET}[K](X_1) = Y_1, \mathsf{ET}[K](X_2) = Y_2, \ldots, \mathsf{ET}[K](X_{|\mathsf{ET}[K]|}) = Y_{|\mathsf{ET}[K]|}$ in the order that they are defined during the

execution $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$. Assume that β_K out of the $|\mathsf{ET}[K]|$ entries were defined in calls to $\mathsf{CreateBlock}(\star,\star,\bot)$ (i.e., "adapted"), and their corresponding indices are $i_{K,1},\ldots,i_{K,\beta_K}$. Then it can be seen the probability that $\mathsf{S}^{\mathcal{RO}}$ samples the remaining $|\mathsf{ET}[K]|-\beta_K$ responses is exactly

$$\frac{(2^n-i_{K,1}+1)\cdots(2^n-i_{K,\beta_K}+1)}{(2^n)_{|\mathsf{ET}[K]|}} \quad \left(\text{its order is } O\Big(\frac{1}{(2^n)^{|\mathsf{ET}[K]|-\beta_K}}\Big)\right).$$

Further assume that at the end of $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$, the random oracle query set $\mathsf{HQueries}$ has γ records. Then, the probability that $\mathcal{RO}$ gives rise to the responses as in $\mathsf{HQueries}$ is exactly $1/2^{2\gamma n}$. We note that the randomness underlying the execution $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ is fully determined by the aforementioned $\sum_{K\in\{0,1\}^k}\big(|\mathsf{ET}[K]|-\beta_K\big)$ lazily sampled entries in ET and the γ records in $\mathsf{HQueries}$. By these,

$$\begin{aligned}\frac{\Pr_E[E\vdash\mathsf{ET}]}{\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\to\mathsf{ET}]} &= \frac{\prod_{K\in\{0,1\}^k}\frac{1}{(2^n)_{|\mathsf{ET}[K]|}}}{\frac{1}{2^{2\gamma n}}\cdot\prod_{K\in\{0,1\}^k}\frac{(2^n-i_{K,1}+1)\cdots(2^n-i_{K,\beta_K}+1)}{(2^n)_{|\mathsf{ET}[K]|}}}\\ &= \frac{2^{2\gamma n}}{\prod_{K\in\{0,1\}^k}(2^n-i_{K,1}+1)\cdots(2^n-i_{K,\beta_K}+1)}.\end{aligned}$$

The number of terms in the denominator is $\sum_{K\in\{0,1\}^k}\beta_K$, i.e., the total number of adapted entries in ET.

Let θ be the total number of calls to $\mathsf{CreateBlock}(\star,\star,\bot)$ during the execution $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$. Then, $\sum_{K\in\{0,1\}^k}\beta_K=2\theta$ since each such (successful) call defines two adapted entries in ET. On the other hand, it holds $\gamma=|\mathsf{HQueries}|=\theta$ by Lemma 10. By the above, $2^{2\gamma n}>\prod_{K\in\{0,1\}^k}(2^n-i_{K,1}+1)\cdots(2^n-i_{K,\beta_K}+1)$, and thus

$$\frac{\Pr_E[E\vdash\mathsf{ET}]}{\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\to\mathsf{ET}]}\geq 1.$$

A.4 Proof of Lemma 8

Recall from Sect. 5.4 that the G_1 execution $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}$ is good, if and only if the bad events never occur during the execution (and thus $\mathsf{S}^{\mathcal{RO}}(\epsilon)$ never aborts by Lemmas 3 and 4). Furthermore, let Θ_1 be the subset of $\mathcal{T}_{\mathsf{e}}$ such that for any $(\epsilon,\mathcal{RO})$ such that $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\to\mathsf{ET}\in\Theta_1$ it holds $D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}=1$ (as defined in Appendix A.2). Then, wlog assume that $\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}=1]\geq\Pr_E[D^{\mathsf{G}_2(E)}=1]$, it holds

$$\begin{aligned}&\Big|\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}=1]-\Pr_E[D^{\mathsf{G}_2(E)}=1]\Big|\\ =\ &\underbrace{\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\text{ is bad}\wedge D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}=1]}_{\leq\Pr[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\text{ is bad}]\leq\frac{6(3Q+np)^2}{(2^n-2Q)^2}+\frac{11nQ+np}{2^n-2Q}\ \text{(Lemma 1)}}\\ &+\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}\text{ is good}\wedge D^{\mathsf{G}_1(\epsilon,\mathcal{RO})}=1]-\underbrace{\Pr_E[D^{\mathsf{G}_2(E)}=1]}_{\geq\sum_{\mathsf{ET}\in\Theta_1}\Pr_E[E\vdash\mathsf{ET}]\ \text{(Lemma 9)}}.\end{aligned}$$

For the remaining probability, we have

$$\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \text{ is good} \wedge D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = 1] = \sum_{\mathsf{ET}\in\Theta_1} \Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET}].$$

Thus

$$\begin{aligned}
&\left|\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} = 1] - \Pr_E[D^{\mathsf{G}_2(E)} = 1]\right| \\
\leq &\frac{6(3Q+np)^2}{(2^n-2Q)^2} + \frac{11nQ+np}{2^n-2Q} + \sum_{\mathsf{ET}\in\Theta_1} \Big(\underbrace{\Pr_{\epsilon,\mathcal{RO}}[D^{\mathsf{G}_1(\epsilon,\mathcal{RO})} \to \mathsf{ET}] - \Pr_E[E \vdash \mathsf{ET}]}_{\leq 0 \quad \text{(Lemma 7)}}\Big) \\
\leq &\frac{6(3Q+np)^2}{(2^n-2Q)^2} + \frac{11nQ+np}{2^n-2Q},
\end{aligned}$$

as claimed.