# Double-Block-Length Hash Function for Minimum Memory Size

Yusuke Naito[1], Yu Sasaki[2], and Takeshi Sugawara[3]

[1] Mitsubishi Electric Corporation, Kanagawa, Japan
Naito.Yusuke@ce.MitsubishiElectric.co.jp
[2] NTT Social Informatics Laboratories, Tokyo, Japan
yu.sasaki.sk@hco.ntt.co.jp
[3] The University of Electro-Communications, Tokyo, Japan
sugawara@uec.ac.jp

**Abstract.** Sharing a common primitive for multiple functionalities is essential for lightweight cryptography, and NIST's lightweight cryptography competition (LWC) considers the integration of hashing to AEAD. While permutations are natural primitive choices in such a goal, for design diversity, it is interesting to investigate how small block-cipher (BC) based and tweakable block-cipher (TBC) based schemes can be. Double-block-length (DBL) hash function modes are suitable to ensure the same security level for AEAD and hashing, but hard to achieve a small memory size. Romulus, a TBC-based finalist in NIST LWC, introduced the DBL hashing scheme Romulus-H, but it requires $3n + k$ bits of memory using an underlying primitive with an $n$-bit block and a $k$-bit (twea)key. Even the smallest DBL modes in the literature require $2n + k$ bits of memory. Addressing this issue, we present new DBL modes EXEX-NI and EXEX-I achieving $(n + k)$-bit state size, i.e., no extra memory in addition to $n + k$ bits needed within the primitive. EXEX-NI is indifferentiable from a random oracle up to $n - \log n$ bits. By instantiating it with SKINNY, we can provide hashing to Romulus with zero memory overhead. EXEX-I is an optimized mode with collision resistance. We finally compare the hardware performances of EXEX-NI, EXEX-I, and Romulus-H with SKINNY-128-384. EXEX-NI and EXEX-I achieve the circuit-area reduction by 2,000+ GE, yielding the total areas being smaller than 70% of that of Romulus-H.

**Keywords:** double-block-length hash · lightweight cryptography · low memory · indifferentiability · collision resistance · tweakable block cipher.

## 1 Introduction

Lightweight cryptography receives great attention in the field of symmetric-key cryptography. The National Institute of Standards and Technology (NIST) is now organizing a competition (NIST LWC) to standardize lightweight authenticated encryption with associated data (AEAD) schemes [29]. In particular, it is the final year of the competition, at the time of writing, and new knowledge on lightweight AEAD schemes is highly important. Sharing a common primitive for multiple functionalities is important for reducing hardware/software costs; it is an essential idea behind AEAD schemes that simultaneously realize both encryption and message authentication. For even higher efficiency, NIST LWC considers the integration of yet another functionality into AEAD: a hashing scheme.

Efficient integration of AEAD and hashing is a challenging task. Using a secret key is the central difference between AEAD and hashing, which results in different security levels for a given state size. Intuitively, to ensure the same level of security, hashing schemes require a larger state than AEAD schemes in order to resist collision attacks based on the birthday paradox. A naive approach to compensate for this difference is to use a larger primitive for hashing, but using a larger primitive for the optional functionality in NIST LWC is unreasonable for lightweight implementation. Compared to block ciphers (BCs) and tweakable block ciphers (TBCs), cryptographic permutations seem more suitable to support both AEAD and hashing by using the duplex construction for AEAD [5] and the sponge construction for hashing [4].

NIST LWC has recently proceeded into the final stage by selecting 10 schemes as finalists [30], where 6 of them are permutation-based schemes, and the rest have diversity; a BC-based scheme, a TBC-based scheme, a stream cipher-based scheme, and a keyed permutation-based scheme. Among the 10 finalists, the 4 schemes support both AEAD and hashing, and all of them adopt the duplex and the sponge constructions. NIST explicitly states that they consider design diversity during the selection [31], and exploring an efficient realization of hashing in other constructions is an important research challenge. The design team of the TBC-based scheme Romulus [12] has recently announced a hashing scheme called Romulus-H [13], which is an MDPH hashing mode [24] instantiated with SKINNY-128-384 [3] used in Romulus AEAD. This is an interesting direction, and the goal of this paper is to explore an optimal construction with respect to the memory size to realize a hashing scheme based on a BC or a TBC, particularly to satisfy the design requirements for NIST LWC.

We begin by recalling NIST's requirements for hash functions: (1) cryptanalytic attacks to find a collision, a second preimage, and a preimage shall require at least $2^{112}$ computations and (2) length extension attacks should be prevented. For example, if part of the message is a secret key, constructing a hash value corresponding to another message under the same key should be infeasible.

**DBL Modes.** Double block length (DBL) hashing modes construct a $2n$-bit hash function from an $n$-bit block cipher.[4] Given that hashing schemes require a larger state than AEAD schemes, DBL modes are very suitable to support both AEAD and hashing schemes by BCs. Moreover, NIST LWC requires at least 112-bit security for a hashing scheme, while most of the existing lightweight

[4] Instead of BCs, TBCs can be used in DBL. Hashing modes replace a key with some public value, hence BCs with a $k$-bit key and TBCs with a $k'$-bit key and a $t$-bit tweak generally play the same role as long as $k = k' + t$. For sake of simplicity, we denote underlying primitives by BCs.

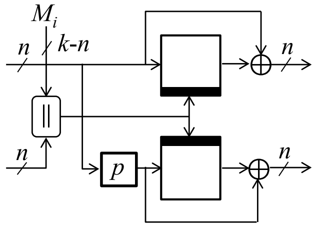

**Fig. 1.** Hirose's DBL CF. $p$ is a permutation without fixed points.

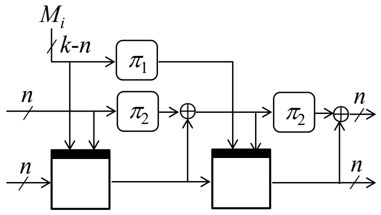

**Fig. 2.** EXEX CF. $\pi_1$ (resp. $\pi_2$) is an $(k-n)$-bit (resp. $n$-bit) linear function.

BCs and TBCs have a block size of 128 bits or shorter, which makes it natural to design a 256-bit hashing scheme from a 128-bit BC. Apart from NIST LWC, DBL modes are motivated by a practical demand; one may want to implement a 256-bit hash function from AES (128-bit block and a 128-bit or a 256-bit key) instead of implementing SHA-256 that has a completely different structure.

A popular approach to design DBL modes is to first design a DBL compression function (CF), which is then converted to a hash function (HF) by the Merkle-Damgård domain extension [9, 21]. There are many such modes, including MDC-2, MDC-4 [7], Tandem-DM, and Abreast-DM [16]. Hirose's DBL construction [10] is one of the most widely known DBL CFs, which is depicted in Fig. 1. Besides, many DBL constructions have been proposed for different goals, e.g., for improving security [1, 18, 24], relaxing the key size limitation for underlying BCs [17, 20], etc. In particular, there is a line of research works for reducing the required memory size, which determines the overall hardware cost in lightweight implementation. The goal of this paper is to optimize the memory size, and we explain below the previous works in this direction.

With an underlying BC having an $n$-bit block and a $k$-bit key, Hirose's DBL CF requires a $(3n+k)$-bit state. Bogdanov et al. proposed a lightweight hash function for RFID tags by instantiating Hirose's construction with a lightweight block cipher PRESENT [6]. Notably, they pointed out that the feed-forward operation in Hirose's DBL construction increases the memory size. Naito's MDPH, the mode used in Romulus-H, also requires a $(3n+k)$-bit state because it is an extension of Hirose's CF; MDPH replaces the standard Merkle-Damgård domain extension with Merkle-Damgård-Permutation (MDP) domain extension [11], thereby providing indifferentiability (from a random oracle) [19].

Özen and Stam broke the $(3n+k)$-bit barrier by proposing a synthetic approach to achieving the $(2n+k)$-bit state size [32]. Their generic construction includes Hirose's DBL CF without the feed-forward, which enables to reduce the state size to $2n+k$ bits. This is an interesting approach because the omission of the feed-forward makes the CF vulnerable, while the security (collision resistance) is guaranteed as a whole HF. Naito followed the same approach [23], and designed another mode achieving $2n+k$ bits of memory in which the CF is Hirose's construction without feed-forward, while the security (indifferentiabil-

**Table 1.** Comparison of DBL modes. $-\epsilon$ denotes that security is lost by a factor of $\log n$. coll resp. indiff denote the collision resistance resp. the indifferentiability.

| Scheme | Memory | Security in bits | Security goal | #BC calls | Message length | Key length | Parallel | Omit KSF$^{-1}$ | Ref. |
|---|---|---|---|---|---|---|---|---|---|
| Tandem-DM | $3n + k$ | $n$ | coll | 2 | $k - n$ | $n < k$ | Yes | No | [16] |
| Abreast-DM | $3n + k$ | $n$ | coll | 2 | $k - n$ | $2n \leq k$ | Yes | No | [16] |
| MDC-2 | $3n + k$ | $3n/5 - \epsilon$ | coll | 2 | $n$ | $n \leq k$ | Yes | No | [7] |
| MDC-4 | $3n + k$ | $5n/8 - \epsilon$ | coll | 4 | $n$ | $n \leq k$ | Yes | No | [7] |
| Mennink | $4n + k$ | $n - \epsilon$ | coll | 3 | $n$ | $n \leq k$ | Yes | No | [20] |
| MJH | $5n + k$ | $n - \epsilon$ | coll | 2 | $n$ | $n \leq k$ | Yes | No | [17] |
| Hirose | $3n + k$ | $n$ | coll | 2 | $k - n$ | $n < k$ | Yes | No | [10] |
| MDPH$^\dagger$ | $3n + k$ | $n - \epsilon$ | indiff | 2 | $k - n$ | $n < k$ | Yes | No | [24, 13] |
| Özen-Stam | $2n + k$ | $n - \epsilon$ | coll | 2 | $k - n$ | $n < k$ | Yes | No | [32] |
| Naito$^\ddagger$ | $2n + k$ | $n$ | indiff | 2 | $k - n$ | $2n \leq k$ | Yes | No | [23] |
| EXEX-NI$^\ddagger$ | $n + k$ | $n - \epsilon$ | indiff | 2 | $k - n$ | $2n \leq k$ | No | Yes | Ours |
| EXEX-I | $n + k$ | $n - \epsilon$ | coll | 2 | $k - n$ | $n < k$ | No | Yes | Ours |

$^\dagger$The mode used in Romulus-H. $^\ddagger$Requires 2 additional BC calls.

ity) for the HF is guaranteed by replacing the domain extension. Naito's mode imposes a stronger requirement on the underlying BC with $k \geq 2n$.

The memory size of existing DBL modes, along with their security and other associated parameters, is compared in Table 1. The smallest memory size in the literature is $2n + k$ bits by Özen-Stam [32] and Naito [23]. The collision resistance may not be sufficient to prevent the length extension attack, one of the requirements in NIST LWC, while indifferentiability suffices to prevent it. The lack of indifferentiability does not immediately imply that the length extension attack is feasible; however, the standard Merkle-Damgård domain extension is known to be vulnerable against the length extension attack.

**Our Contributions.** In this paper, we present new DBL hash function modes EXEX-NI (NI for <u>n</u>ested <u>i</u>terated) and EXEX-I (I for <u>i</u>terated) achieving $(n+k)$-bit state size using an underlying primitive having an $n$-bit block and a $k$-bit key. This is the smallest as compared in Table 1, and is optimal because we need $n + k$ bits just for implementing the BC. EXEX-NI and EXEX-I also accept a TBC of an $n$-bit block, a $k'$-bit key, and a $t$-bit tweak such that $k = k' + t$. Their instantiations with SKINNY can be efficient alternatives to Romulus-H; our modes provide hashing to Romulus (AEAD) with zero memory overhead.

Designing DBL modes with a small memory size is challenging. We first observe that many existing DBL modes call the underlying BC at least twice, and the results of the first BC call (or the value after the feed-forward) are stored on the memory, which will be used as a half of the compression function output after the second BC call. Our idea is to save such a memory. Namely, we use the result of the first BC call to compute an input to the second BC call, as shown in Fig. 2. [5] In this way, all the memory is always actively used to compute both

[5] Linear dependency between the message and the key does not degrade security. Intuitively, previous block-cipher's output spreads to next block-cipher's key. This makes two keys different, and block-cipher's outputs become different random strings.

BC calls, which enables us to reduce the memory size to $n + k$ bits. As a natural consequence of serialization, the new CF cannot run these consecutive BCs in parallel (see "Parallel" in Table 1), however, that has a negligible impact on low-area implementations.

Due to the strictly restricted memory size, we adopt the same approach as Özen-Stam and Naito that use a vulnerable CF [32, 23], which actually allows an attacker to easily find a preimage of the CF. This implies that there exists a preimage attack on HF by using the meet-in-the-middle approach with $O(2^n)$ computational cost, though the digest size is $2n$ bits. As required in NIST LWC, not all practical usage require higher security for the preimage resistance than the collision resistance. [6] EXEX-NI and EXEX-I are suitable for such demand. We prove the security of EXEX-NI with respect to indifferentiability up to $n - \log n$ bits, which ensures that EXEX-NI resists the length-extension attacks. Recall that NIST LWC requires 112-bit security. This can be satisfied by using a 128-bit BC or a 128-bit TBC, say AES or SKINNY-128, even by considering the security loss of $\log n$ bits. We believe that EXEX-NI is an attractive design, especially as an alternative to the MDPH mode in Romulus-H.

As mentioned above, our CF is invertible. In contrast, to design an indifferentiable HF, we need a non-invertible function somewhere in the computation. In EXEX-NI, we fill this gap by using Coron et al.'s NMAC hash [8]: we add a special finalization function with 2 BC calls, making the finalization function non-invertible with a constant-time overhead. EXEX-NI requires an underlying BC to support the key size $k \geq 2n$, and the requirement can be satisfied by many existing BC designs, e.g., by AES-256 and SKINNY-128-256.

Although it is required in NIST LWC, we observe that resisting the length-extension attack may be unnecessary in practice. This is because all the NIST LWC candidates support an AEAD scheme, thus keyed computations such as MACs can be done by using the AEAD scheme. This motivates us to consider relaxing the security goal to the collision resistance rather than the indifferentiability. EXEX-I is a design for this purpose. EXEX-I does not require the finalization function, which reduces the number of BC calls by 2. This has a significant impact on the performance for short messages. Moreover, the requirement of the key size of the underlying BC is relaxed compared to EXEX-NI.

As shown in Fig. 2, we prove the security of EXEX-NI and EXEX-I by assuming that some part of the key state can be updated by using any linear functions $\pi_1$ and $\pi_2$, which allows us interesting optimization regarding on-the-fly key scheduling (see "Omit KSF$^{-1}$" in Table 1). This idea was first introduced by Naito et al. [26] and known to reduce the hardware cost when a key schedule function (and a tweak schedule function) of the underlying BC is a state-wise linear update, which is particularly useful when the underlying BC is SKINNY. In short, by setting $\pi_1$ and $\pi_2$ to the key schedule function (KSF), the updated key state during the computation of BC can immediately be used to compute the next BC without applying the KSF$^{-1}$.

[6] HMAC [28] and Hash-then-MAC [15] are example use-cases whose security is reduced to coll of the hashing scheme, thus does not require higher security level than coll.

**Table 2.** Comparison between EXEX and the sponge construction. A check mark ✓ shows that the target mode is advantageous.

| Modes | Sponge | EXEX |
|---|---|---|
| Base primitive | Permutation | BC or TBC |
| Memory size for modes | $n+k$ | $n+k$ |
| Number of primitive calls | 1 | 2 |
| Construction simplicity | ✓ | — |
| Proof simplicity | ✓ | — |
| Memory size with TI | — | ✓ |
| Backward compatibility with AES | — | ✓ |

We finally compare the hardware performances of EXEX-NI, EXEX-I, and Romulus-H by instantiating them with SKINNY-128-384 and implementing them with the same design policy. Thanks to the smaller memory size and the optimized tweakey-schedule implementation, EXEX-NI and EXEX-I achieved the circuit-area reduction by 2,000+ GE, yielding the total areas being smaller than 70% of that of the Romulus-H.

**Related Work.** It is well-known that permutation-based schemes (the sponge construction) provide an excellent hardware performance. Although BC-based and TBC-based hashing schemes are important with respect to the design diversity, it is still interesting to compare the performance of those schemes.

Memory sizes for the sponge construction and EXEX are the same to achieve the same rate under the same security level. The sponge construction is advantageous with respect to the number of primitive calls, design simplicity, and proof simplicity, though the number of primitive calls is a low-priority criterion for small implementations. EXEX is advantageous with respect to other metrics. First, TBC-based designs are advantageous with masking countermeasures against side-channel attack, as discussed in [25]. Second, it has backward compatibility with AES. Many microprocessors have AES accelerators and EXEX provides efficient hashing to them. The comparison is summarized in Table 2. Overall, EXEX is competitive with the key performance metric memory size and offers new options for implementers depending on the criteria of their choice.

**Outline.** Section 2 introduces notations and fundamentals. Section 3 shows our general approach to obtain DBL HF only with $n+k$-bit memory. Section 4 shows our higher-security variant EXEX-NI that satisfies the requirements in NIST LWC, followed by its proof in Section 5. Section 6 shows our rigorously optimized variant EXEX-I that compromises the security goal to collision resistance but provides better performance, particularly for short messages. Finally, we make a hardware performance comparison by implementing EXEX-NI, EXEX-I, and Romulus-H instantiated with SKINNY-128-384 in Sect. 7. Section 8 is the conclusion.

## 2 Preliminaries

**Notation.** Let $\{0,1\}^*$ be the set of all bit strings. Let $\lambda$ be an empty string, and $\emptyset$ an empty set. For an integer $n \geq 0$, $\{0,1\}^n$ be the set of all $n$-bit strings, and $(\{0,1\}^n)^*$ be the set of all strings whose bit lengths are multiples of $n$. For an integer $i > 0$, let $[i]$ be the set of positive integers less than or equal to $i$. For an $m\ell$-bit string $M$, we write its partition into $m$-bit strings as $(M_1, M_2, \ldots, M_\ell) \xleftarrow{m} M$. For integers $r, s$ with $0 \leq s \leq r$ and an $r$-bit string $X$, the most (resp. least) significant $s$ bits of $X$ is denoted by $[X]^s$ (resp. $[X]_s$). For a bit string $Y$, $X \leftarrow Y$ means that $Y$ is assigned to $X$. $X \xleftarrow{\$} \mathcal{X}$ means that an element is sampled uniformly at random from a finite set $\mathcal{X}$ 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 $\mathcal{Y} \leftarrow \mathcal{X} \cup \mathcal{Y}$.

**Security Definition of Hash Function.** Our proofs are given in the ideal cipher model. For positive integers $k$ and $n$, let $E$ be (an encryption function of) a BC having a $k$-bit key and an $n$-bit plaintext block, which is a set of $n$-bit permutations indexed by keys. The decryption function of $E$ is denoted by $D : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. $\mathsf{BC}(k,n)$ denotes the set of all BCs with $k$-bit keys and $n$-bit blocks. An ideal cipher is defined as $E \xleftarrow{\$} \mathsf{BC}(k,n)$. In the ideal cipher model, an adversary has access to $E$ and $D$.

Let $\mathsf{H}^E : \{0,1\}^* \to \{0,1\}^{2n}$ be a DBL hash function using a BC $E$. We define security notions for $\mathsf{H}^E$.

Preimage Security. In the security game, a computationally unbounded adversary $\mathbf{A}$ is given oracle access to $(E,D)$, where $E \xleftarrow{\$} \mathsf{BC}(k,n)$. The goal of $\mathbf{A}$ is to find a massage $M$ of a given hash value $H$. The advantage function of an adversary $\mathbf{A}$ is defined as $\mathsf{Adv}^{\mathsf{pre}}_{\mathsf{H},H}(\mathbf{A}) = \mathsf{Pr}\left[\mathsf{H}^E(M) = H : M \leftarrow \mathbf{A}^{E,D}(H)\right]$, where the probabilities are taken over $\mathbf{A}$, $H$, and $E$.

Collision Security. In the security game, a computationally unbounded adversary $\mathbf{A}$ is given oracle access to $(E,D)$, where $E \xleftarrow{\$} \mathsf{BC}(k,n)$. The goal of $\mathbf{A}$ is to find a pair of messages $M$ and $M'$ such that the hash values are equal. The advantage function of an adversary $\mathbf{A}$ is defined as $\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{H}}(\mathbf{A}) = \mathsf{Pr}\left[\mathsf{H}^E(M) = \mathsf{H}^E(M') : (M,M') \leftarrow \mathbf{A}^{E,D}, M \neq M'\right]$, where the probabilities are taken over $\mathbf{A}$ and $E$.

Indifferentiability from a Random Oracle. The indifferentiability of $\mathsf{H}^E$ from a random oracle is indistinguishability between $\mathsf{H}^E$ (in the ideal cipher model) and a random oracle. $\mathsf{Func}(*, 2n)$ denotes the set of all functions from $\{0,1\}^*$ to $\{0,1\}^{2n}$, and a random oracle is defined as $\mathcal{RO} \xleftarrow{\$} \mathsf{Func}(*, 2n)$.

In the security game, an adversary $\mathbf{A}$ tries to distinguish a real world from an ideal world. $\mathbf{A}$ has access to a hash oracle $L$, an encryption oracle $R_E$, and a decryption oracle $R_D$. In the real world, these oracles are $(L, R_E, R_D) = (\mathsf{H}^E, E, D)$, where $E \xleftarrow{\$} \mathsf{BC}(k,n)$. In the ideal world, these oracles are $(L, R_E, R_D) = (\mathcal{RO}, \mathsf{S}^{\mathcal{RO}}_E, \mathsf{S}^{\mathcal{RO}}_D)$, where $\mathcal{RO} \xleftarrow{\$} \mathsf{Func}(*, 2n)$, and $\mathsf{S}^{\mathcal{RO}}_E$ and $\mathsf{S}^{\mathcal{RO}}_D$ are simulators

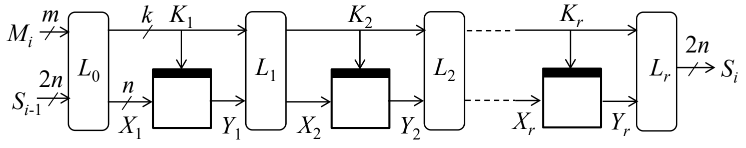

**Fig. 3.** $\mathsf{GLMCF}^E$: a general construction for low-memory compression function.

with access to $\mathcal{RO}$. After the interaction, $\mathbf{A}$ outputs a decision bit in $\{0,1\}$. An output of an adversary $\mathbf{A}$ with access to oracles $L, R_E, R_D$ is denoted by $\mathbf{A}^{L,R_E,R_D}$. For a simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$, the advantage function of an adversary $\mathbf{A}$ is defined as $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathsf{S}}(\mathbf{A}) = \mathsf{Pr}\left[\mathbf{A}^{\mathsf{H}^E,E,D} = 1\right] - \mathsf{Pr}\left[\mathbf{A}^{\mathcal{RO},\mathsf{S}_E^{\mathcal{RO}},\mathsf{S}_D^{\mathcal{RO}}} = 1\right]$, where the probabilities are taken over $\mathbf{A}$, $E$, $\mathcal{RO}$, and $\mathsf{S}$. $\mathsf{H}^E$ is indifferentiable from a random oracle if for any adversary $\mathbf{A}$, there exists a simulator $\mathsf{S}$ such that the advantage function is upper-bounded by a negligible probability. In this paper, we call queries to $L$, $R_E$, and $R_D$ hash queries, encryption queries, and decryption queries, respectively.

## 3 Conditions for Secure Low Memory DBL Hash Designs

In this section, we approach to DBL hash functions with the smallest memory size, which uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and uses a memory of $k+n$ bits. We first introduce a generic framework to construct a CF in Sect. 3.1. We then derive conditions for parameters to resist collision attacks in Sect. 3.2. We show that such a CF is always invertible, thus requires additional effort to be indifferentiable in Sect. 3.3. Finally, a generic framework to construct a HF is introduced in Sect. 3.4.

### 3.1 Generic Construction of Low-Memory DBLCF

To design a DBL hash function with a $(k+n)$-bit memory, we need to design a CF with a $(k+n)$-bit memory. In this subsection, we introduce $\mathsf{GLMCF}^E$: a generic construction of the low-memory CF with a block cipher $E$, which is depicted in Fig. 3.

Let $m$ be the bit-length of a message block. $\mathsf{GLMCF}^E$ takes as input a $2n$-bit state value $S_{i-1}$ and an $m$-bit message $M_i$, and generates a $2n$-bit output $S_i$. $\mathsf{GLMCF}^E$ calls a BC with $n$-bit block and a $k$-bit key, denoted by $E$. $E$ can be called multiple times even under the restriction of $(k+n)$-bit memory if all $E$ calls are sequentially processed. At this stage, we do not fix the number of calls of $E$, and let $r$ be this number. Since $E$ uses the entire $k+n$ bits, we cannot carry anything over the $E$ call. This restricts the design of $\mathsf{GLMCF}^E$ to be an iteration of a linear function $L_i$ and $E$. Namely, we first apply a linear function $L_0$ to map a $(2n+m)$-bit state to a $(k+n)$-bit state. Then, the state is updated to another $(k+n)$-bit state by $E$ and linear function $L_1$. This is iterated $r$ times

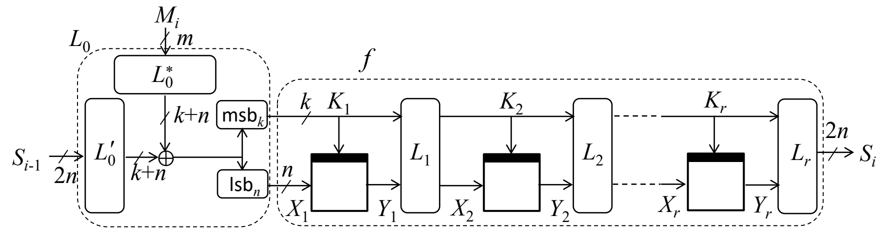

**Fig. 4.** Another formulation of $\mathsf{GLMCF}^E$. $\mathsf{msb}_k$ (resp. $\mathsf{lsb}_n$) returns the most (resp. least) significant $k$ (resp. $n$) bits of the input.

to apply BC $r$ times. Lastly, the $(k+n)$-bit state is transformed to $2n$-bit output by a linear function $L_r$. At this stage, we do not make any assumption on $m$, thus $L_0$ can be either an expanding or a contracting function and $L_r$ can be its opposite.

The formal description is as follows. Let $L_0 : \{0,1\}^{2n} \times \{0,1\}^m \to \{0,1\}^k \times \{0,1\}^n$, $L_1 : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k \times \{0,1\}^n, \ldots, L_{r-1} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k \times \{0,1\}^n$, and $L_r : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^{2n}$ be linear functions. We give the procedure of $\mathsf{GLMCF}^E : \{0,1\}^{2n} \times \{0,1\}^m \to \{0,1\}^{2n}$.

```
– $\mathsf{GLMCF}^E(S_{i-1}, M_i)$
    1. $(K_1, X_1) \leftarrow L_0(S_{i-1}, M_i)$
    2. **for** $j = 1, \ldots, r-1$ **do** $Y_j \leftarrow E(K_j, X_j)$; $(K_{j+1}, X_{j+1}) \leftarrow L_j(K_j, Y_j)$
    3. $Y_r \leftarrow E(K_r, X_r)$; $S_i \leftarrow L_r(K_r, Y_r)$; **return** $S_i$
```

## 3.2 Conditions of $m$, $k$, and $n$ for $n$-bit Collision Resistance

Now we are lifting the CF to a HF. Suppose that a DBL HF is constructed by iteratively applying $\mathsf{GLMCF}^E$, which is denoted by $i\mathsf{GLMCF}^E$. For sake of simplicity, an input message $M$ is in $(\{0,1\}^m)^*$. Let $IV$ be a $2n$-bit constant.

```
– $i\mathsf{GLMCF}^E(M)$
    1. $(M_1, M_2, \ldots, M_\ell) \xleftarrow{m} M$; $S_0 \leftarrow IV$
    2. **for** $i = 1, \ldots, \ell$ **do** $S_i \leftarrow \mathsf{GLMCF}^E(S_{i-1}, M_i)$
    3. **return** $S_\ell$
```

First, the construction needs to satisfy $n \leq k$. In fact, if $n > k$, $n$-bit security cannot be ensured due to the birthday attack on the $(n+k)$-bit state.

Second, the construction needs to satisfy $k-n \geq m$. In fact, there is an attack that breaks the collision security on $i\mathsf{GLMCF}^E$ with $O(2^{(k+n-m)/2})$ complexity, which we discuss in this section. In this attack, we split the linear function $L_0$ in $\mathsf{GLMCF}^E$ into two linear functions $L_0' : \{0,1\}^{2n} \to \{0,1\}^{k+n}$ and $L_0^* : \{0,1\}^m \to \{0,1\}^{k+n}$ that satisfy $L_0(S_{i-1}, M_i) = L_0'(S_{i-1}) \oplus L_0^*(M_i)$. $L_0^*$ must be injective, otherwise one can trivially find a collision. The compression function with some formulation is given in Fig. 4, where $f : \{0,1\}^{n+k} \to \{0,1\}^{2n}$ is a composed function covering from the first BC to the last linear function $L_r$. The security bound is given in the following lemma.

**Lemma 1.** *Let $f$ be an ideal function. For any $S_i \in \{0,1\}^{2n}$, there exists an adversary $\mathbf{A}$ making $Q \cdot \max\{1, \lceil n/m \rceil\}$ queries to $f$ such that $\mathsf{Adv}^{\mathsf{coll}}_{i\mathsf{GLMCF}}(\mathbf{A}) = \Omega\left(\frac{Q^2}{2^{k+n-m}}\right)$.*

*Proof.* We define an adversary $\mathbf{A}$ that finds a collision of $i\mathsf{GLMCF}^E$ where $f$ is ideal. Let $u := \lceil n/m \rceil$.

1. For $j = 1, \ldots, Q$ do the following steps.
   (a) Select a $j$-th message $(M_1^{(j)} \| \cdots \| M_u^{(j)})$ that is distinct from all previous messages $(M_1^{(1)} \| \cdots \| M_u^{(1)}), \ldots, (M_1^{(j-1)} \| \cdots \| M_u^{(j-1)})$.
   (b) Calculate the $u$-th state denoted by $S_u^{(j)}$ for the message $(M_1^{(j)} \| \cdots \| M_u^{(j)})$ by making queries to $f$.
   (c) For each $j' \in [j-1]$, check if there exist message blocks $M_{u+1}^{(j)}$ and $M_{u+1}^{(j')}$ such that $L'_0(S_u^{(j)}) \oplus L^*_0(M_{u+1}^{(j)}) = L'_0(S_u^{(j')}) \oplus L^*_0(M_{u+1}^{(j')})$ which causes a collision at the $(u+1)$-th CF call.
   (d) If such an index $j'$ exists then $M \leftarrow (M_1^{(j)} \| \cdots \| M_u^{(j)} \| M_{u+1}^{(j)})$, $M' \leftarrow (M_1^{(j')} \| \cdots \| M_u^{(j')} \| M_{u+1}^{(j')})$, and go to Step 3.
2. Choose messages $M \xleftarrow{\$} \{0,1\}^m$ and $M' \xleftarrow{\$} \{0,1\}^m$.
3. Return $(M, M')$.

The number of choices of the XOR value $L^*_0(M_{u+1}^{(j)}) \oplus L^*_0(M_{u+1}^{(j')})$ is $2^m$. Hence, for each pair $(j, j')$, the probability that there exist message blocks $M_{u+1}^{(j)}$ and $M_{u+1}^{(j')}$ such that $L'_0(S_u^{(j)}) \oplus L^*_0(M_{u+1}^{(j)}) = L'_0(S_u^{(j')}) \oplus L^*_0(M_{u+1}^{(j')})$ is $\Omega(1/2^{k+n-m})$. By summing the probability for each pair, we have $\mathsf{Adv}^{\mathsf{coll}}_{i\mathsf{GLMCF}^E}(\mathbf{A}) = \Omega(Q^2/2^{k+n-m})$. □

To ensure $n$-bit security against the collision attack, $k + n - m \geq 2n$, which results in $k - n \geq m$. This implies that $m$ can take any value between $k - n$ and 1. The memory size is $k+n$ bits for any $m$, while the number of bits processed in each invocation of $\mathsf{GLMCF}^E$ decreases when $m$ becomes small. In the rest of the paper, we fix $k - n = m$, which is the optimal choice in terms of the performance under the restriction of the $(k+n)$-bit memory.

### 3.3 Conditions for Indifferentiability: Invertibility of GLMCF

The conditions in Sect. 3.2 were derived for the collision resistance, which is insufficient for the indifferentiability (to ensure resistance against length-extension attacks required by NIST). Towards indifferentiable constructions, we first show that one can break the preimage security of $\mathsf{GLMCF}^E$, where $E$ is an ideal cipher.

**Lemma 2.** *Fix $r$. For any $S_i \in \{0,1\}^{2n}$, there exists an adversary $\mathbf{A}$ making $r$ queries such that $\mathsf{Adv}^{\mathsf{pre}}_{\mathsf{GLMCF}^E, S_i}(\mathbf{A}) = 1$.*

*Proof.* We define an adversary $\mathbf{A}$ that finds a preimage of a value $S_i$.

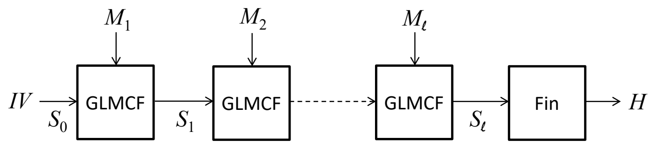

**Fig. 5.** Hash Function GLMHF.

```
1. Find $(K_r, Y_r)$ s.t. $S_i = L_r(K_r, Y_r)$; $X_r \leftarrow D(K_r, Y_r)$
2. for $j = r-1, \ldots, 1$ do
       Find $(K_j, Y_j)$ s.t. $(K_{j+1}, X_{j+1}) = L_j(K_j, Y_j)$; $X_j \leftarrow D(K_j, Y_j)$
3. Find $(S_{i-1}, M_i)$ s.t. $(K_1, X_1) = L_0(S_{i-1}, M_i)$; Return $(S_{i-1}, M_i)$
```

Since $L_0, L_1, \ldots, L_r$ are linear functions, for each $j = 0, 1, \ldots, r$, given an output of $L_j$, one can easily find the input. Hence, the adversary finds a preimage $(M_i, S_{i-1})$ by $r$ queries. $\square$

This analysis shows that only with $(k+n)$-bit memory satisfying $k - n = m$, $\mathsf{GLMCF}^E$ is easily invertible. To obtain an indifferentiable HF, we need a non-invertible part somewhere in HF.

### 3.4 Generic Construction of Low Memory DBL HF

The analysis in Sect. 3.3 motivates us to introduce a non-invertible finalization function $\mathsf{Fin} : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$. Here, we define $\mathsf{GLMHF} : \{0,1\}^* \rightarrow \{0,1\}^{2n}$, a generic construction of low memory hash function using the compression function $\mathsf{GLMCF}^E$ and a finalization function $\mathsf{Fin}$. Let $\mathsf{pad} : \{0,1\}^* \rightarrow (\{0,1\}^m)^*$ be an injective padding function.

```
– $\mathsf{GLMHF}^{E,\mathsf{Fin}}(M)$
    1. $(M_1, M_2, \ldots, M_\ell) \overset{m}{\leftarrow} \mathsf{pad}(M)$; $S_0 \leftarrow IV$
    2. for $i = 1, \ldots, \ell$ do $S_i \leftarrow \mathsf{GLMCF}^E(S_{i-1}, M_i)$
    3. $H \leftarrow \mathsf{Fin}(S_\ell)$; return $H$
```

Note that $IV$ is a $2n$-bit constant. Fig. 5 shows the structure of GLMHF. In the next section, we propose EXEX-NI by specifying details in GLMHF.

## 4 EXEX-NI: Low Memory Indifferentiable DBL HF

In this section, we specify every details of the general framework introduced in Sect. 3. In particular, a compression function EXEX and our indifferentiable DBF mode EXEX-NI are defined in Sect 4.1. An overview of its indifferentiability is given in Sect. 4.2 and Sect. 4.3.

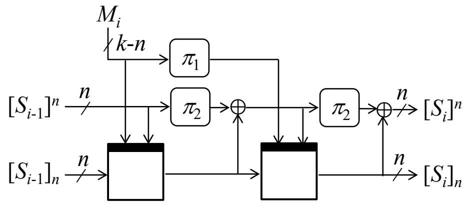

**Fig. 6.** Compression Function EXEX. The figure shows the structure of the CF block $S_{i-1} \xrightarrow{M_i} S_i$ in the proof of Theorem 1.

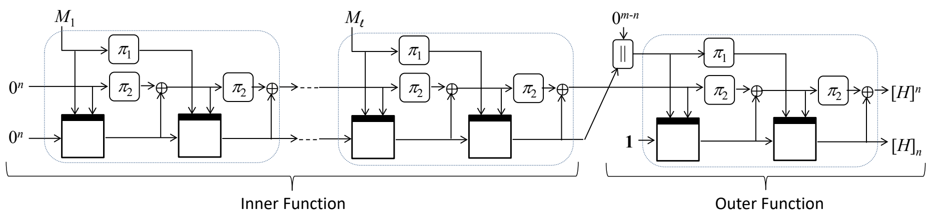

**Fig. 7.** Hash Function EXEX-NI. The inner function of this figure shows the structure of the path $0^{2n} \xrightarrow{M_1\|M_2\|\cdots\|M_\ell} S_\ell$ in the proof of Theorem 1.

## 4.1 Specifications of EXEX and EXEX-NI

To realize our modes using $\mathsf{GLMCF}^E$, we should specify the number of $E$ calls $r$ and the linear functions $L_0, \ldots, L_r$. Choosing the same linear function for $L_0$ to $L_{r-1}$ is a reasonable decision considering implementation efficiency. In each invocation of $\mathsf{GLMCF}^E$, the $2n$-bit state (chaining values) must be updated non-linearly. For this purpose, we XOR BC's output to the least significant $n$ bits of the key state. To provide $2n$-bit entropy from BC's output, we set $r = 2$.[7]

We can prove security only with the above configuration, but in addition, we assume that the $(k-n)$-bit and the remaining $n$-bit key states are independently updated by any linear function $\pi_1$ and $\pi_2$. As shown by Naito et al. [26], a proof over $\pi_1$ and $\pi_2$ provides a certain optimization of the memory size when a key (and a tweak) schedule function of $E$ is a state-wise linear update, which is particularly useful for SKINNY. The intuition behind is that $k$ bits of memory for the key state is updated by a key schedule function inside $E$, thus starting the next $E$ with the state after the key schedule function is more efficient. Indeed, if the next $E$ takes as input the key state before the key schedule function, the $k$ bits of memory must be reproduced by computing the inverse of the key schedule function. We will discuss this optimization later in Sect. 7. Note that $\pi_1$ and $\pi_2$

[7] For an EXEX-based hash function with $r = 1$ (single BC call), one can easily found a collision with $O(2^{n/2})$ complexity: Choosing distinct $2^{n/2}$ single-block messages, a collision of the BC outputs occurs with non-negligible probability, yielding a collision on the internal state. Note that when the linear layers of $\mathsf{GLMCF}^E$ are arbitrary, attacking $\mathsf{GLMCF}^E$ with $r = 1$ is non-trivial and an open problem.

can also be the identity map, hence if $E$ does not have such a structure, we use the identity map to avoid having extra computations.

$\mathsf{EXEX}^E$ uses a BC $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ with $k > n$. Let $m = k - n$. The formal definition of $\mathsf{EXEX}^E : \{0,1\}^{2n} \times \{0,1\}^m \to \{0,1\}^{2n}$ is given below, which is also depicted in Fig. 6.

```
– $\mathsf{EXEX}^E(S_{i-1}, M_i)$:
   1. $Y \leftarrow E(M_i \| [S_{i-1}]^n, [S_{i-1}]_n)$; $K \leftarrow \pi_1(M_i) \| (\pi_2([S_{i-1}]^n) \oplus Y)$;
   2. $[S_i]_n \leftarrow E(K, Y)$; $[S_i]^n \leftarrow \pi_2([K]_n) \oplus [S_i]_n$; return $S_i$
```

We next define a $\mathsf{EXEX\text{-}NI} : \{0,1\}^* \to \{0,1\}^{2n}$. Let $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^m)^*$ be an injective padding function, e.g., one-zero padding $\mathsf{pad}(M) = M \| 10^w$ where $w = m - 1 - |M| \bmod m$. We realize $\mathsf{Fin}$ in $\mathsf{GLMHF}$ by reusing $\mathsf{EXEX}$ to save the memory size. The nested-iterated construction enables us to achieve it. Let $\mathbf{i}$ be an $n$-bit representation of a positive integer $i$, e.g., $\mathbf{1} = 0^{n-1}1$. The definition is given below, which is also depicted in Fig. 7.

```
– $\mathsf{EXEX\text{-}NI}^E(M)$:
   1. $S_0 \leftarrow 0^{2n}$; $M_1, M_2, \cdots, M_\ell \xleftarrow{m} \mathsf{pad}(M)$
   2. for $i = 1, \ldots, \ell$ do $S_i \leftarrow \mathsf{EXEX}^E(S_{i-1}, M_i)$ // Inner Function
   3. $H \leftarrow \mathsf{EXEX}^E([S_\ell]^n \| \mathbf{1}, 0^{m-n} \| [S_\ell]_n)$ // Outer Function
   4. return $H$
```

## 4.2 Indifferentiability of EXEX-NI

We give an upper-bound of the indifferentiability of $\mathsf{EXEX\text{-}NI}$ below.

**Theorem 1.** *Let $\mu$ be any positive integer. There exists a simulator $\mathsf{S}$ such that for any adversary $\mathbf{A}$ running in time $t$ and making $q$ hash queries with $\sigma$ BC calls in total and $p$ encryption or decryption queries,*

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{EXEX\text{-}NI},\mathsf{S}}(\mathbf{A}) \le 2^{n+2} \cdot \binom{3Q}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu} + \frac{6\mu Q + 19Q}{2^n - 3Q} + \frac{22Q^2}{(2^n - 3Q)^2},$$

*where $Q = \sigma + p$. $\mathsf{S}$ runs in time $t + O(p)$ and makes at most $p$ queries.*

We next study the upper-bound.

– Putting $\mu = n$ and using Stirling's approximation ($x! \ge (x/e)^x$ for any $x$), we have $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{EXEX\text{-}NI},\mathsf{S}}(\mathbf{A}) \le 4 \cdot \left(\frac{3eQ}{n(2^n-2Q)}\right)^n + \frac{6nQ+19Q}{2^n-3Q} + \frac{22Q^2}{(2^n-3Q)^2}$. The upper-bound ensures that $\mathsf{EXEX\text{-}NI}$ is indifferentiable from a random oracle up to $O(2^n/n)$ query complexity.
– Consider a BC with $n = 128$. In this case, putting $\mu = 17$ and using Stirling's approximation, $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{EXEX\text{-}NI},\mathsf{S}}(\mathbf{A}) \le \left(\frac{97Q}{2^{128}-3Q}\right)^{17} + \frac{121Q}{2^{128}-3Q} + \frac{22Q^2}{(2^{128}-3Q)^2}$. The upper-bound is less than $1/2$ as long as $Q \le 2^{118}$. Thus, $\mathsf{EXEX\text{-}NI}$ is indifferentiable from a random oracle up to $2^{118}$ query complexity.

## 4.3 Overview of the Proof of Theorem 1

We briefly describe the proof of Theorem 1 along with some definitions. We give the full proof in Sect. 5. The goal of this proof is to construct a simulator $\mathsf{S} = (\mathsf{S}_E, \mathsf{S}_D)$ such that the real-world oracles are indistinguishable from the ideal-world oracles up to $O(2^n/n)$ query complexity. The real-world oracles are $(L, R_E, R_D) = (\mathsf{EXEX\text{-}NI}^E, E, D)$, and the ideal-world oracles are $(L, R_E, R_D) = (\mathcal{RO}, \mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$.

Firstly, we give several definitions to explain an outline of our simulator.

**Definition 1 (query-response of $R_E/R_D$).** *An encryption (resp. decryption) query is denoted by $(K, X) \in \{0,1\}^k \times \{0,1\}^n$ (resp. $(K, Y) \in \{0,1\}^k \times \{0,1\}^n$) and the response is denoted by $Y \in \{0,1\}^n$ (resp. $X \in \{0,1\}^n$). Hence, $Y = R_E(K, X)$ and $X = R_D(K, Y)$. Let $\mathcal{L}_{\mathsf{qr}}$ be a set of tuples $(K, X, Y)$ of $R_E$ or $R_D$. A tuple in $\mathcal{L}_{\mathsf{qr}}$ is called R block.*

**Definition 2 (block).** *A CF block is a tuple $(S_{i-1}, M_i, S_i)$ which is defined by two R blocks with the relation $S_i = \mathsf{EXEX}^{R_E}(S_{i-1}, M_i)$. The CF block is denoted by $S_{i-1} \xrightarrow{M_i} S_i$ (see Fig. 6). $\mathcal{L}_{\mathsf{block}}$ is a set of all CF blocks obtained from $\mathcal{L}_{\mathsf{qr}}$.*

**Definition 3 (Path).** *A path is a CF block or a concatenation of CF blocks which start from the initial value $0^{2n}$. For a sequence of CF blocks $0^{2n} \xrightarrow{M_1} S_1, S_1 \xrightarrow{M_2} S_2, \ldots, S_{\ell-1} \xrightarrow{M_\ell} S_\ell$, we denote the concatenation by $0^{2n} \xrightarrow{M_1\|M_2\|\cdots\|M_\ell} S_\ell$. Hence, the path represents the inner function of $\mathsf{EXEX\text{-}NI}^{R_E}(M_1\|M_2\|\cdots\|M_\ell)$ (see Fig. 7). $\mathcal{L}_{\mathsf{path}}$ is a set of all paths obtained from $\mathcal{L}_{\mathsf{block}}$.*

**Definition 4.** *For a path $0^{2n} \xrightarrow{M} S$ and an input $(K, X)$ to $R_E$, if $(K, X)$ is the first R block at the next CF block, i.e., $S = [K]_n\|X$, then the relation is denoted by $S \overset{in}{\rightsquigarrow} (K, X)$. If $(K, X)$ is the input of the first R block at the outer function connected with the path $0^{2n} \xrightarrow{M} S$, i.e., $[K]_{2n} = [S]_n\|[S]^n$, $X = \mathbf{1}$, and $[K]^{m-n} = 0^{m-n}$, then the relation is denoted by $S \overset{out}{\rightsquigarrow} (K, X)$. We abuse the notation for a CF block $S' \xrightarrow{M'} H$, i.e., if $[S']_n = \mathbf{1}$, $S = [S']^n\|[M']_n$, and $[M']^{m-n} = 0^{m-n}$ then the relation is denoted by $S \overset{out}{\rightsquigarrow} (S', M')$.*

We next specify a relation between $L$ and $R_E$ in the real world. In the real world, for each query $M$, the response $L(M)$ is defined as $L(M) = \mathsf{EXEX\text{-}NI}^{R_E}(M)$ where $R_E = E$, thus the following relation is satisfied.

$$\forall\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}, \left(S' \xrightarrow{M'} H\right) \in \mathcal{L}_{\mathsf{block}} \quad \text{s.t. } S \overset{out}{\rightsquigarrow} (S', M') : L(M) = H. \tag{1}$$

On the other hand, in the ideal world, $L = \mathcal{RO}$ is a monolithic function. We thus need to construct a simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$ so that Eq. (1) is satisfied and the simulator behaves like an ideal cipher.

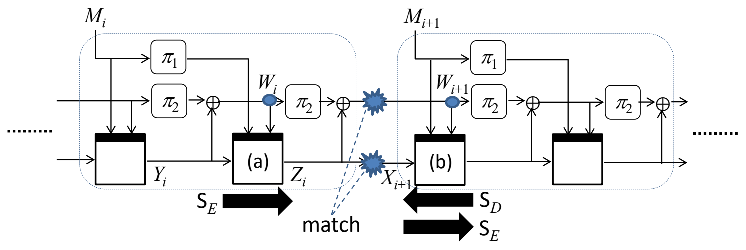

**Fig. 8.** Collision event $\mathsf{E1}$. The $R$ block (a) (resp. (b)) is defined by $\mathsf{S}_E$ (resp. $\mathsf{S}_E$ or $\mathsf{S}_D$). The $R$ block (b) by $\mathsf{S}_E$ is defined before the $R$ block (a).

We explain an outline of our simulator. For each query to $\mathsf{S}_E/\mathsf{S}_D$, to behave like an ideal cipher, the response is defined by lazy sampling.[8] In addition, in order to ensure the relation in Eq. (1), the simulator keeps paths in a table $\mathcal{T}_{path}$ that are constructed from $R$ blocks. Specifically, for each tuple $(K, X, Y)$ defined by $\mathsf{S}_E$, if there exists a path $\left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{T}_{path}$ such that $S' \overset{in}{\rightsquigarrow} (K, X)$, then a new path $\left(0^{2n} \xrightarrow{M} S\right)$ is added to $\mathcal{T}_{path}$, which is defined by appending the CF block $\left(S' \xrightarrow{M^*} S\right)$ having the $R$ block $(K, X, Y)$ to $\left(0^{2n} \xrightarrow{M'} S'\right)$, where $M = M' \| M^*$. Since the new path represents an inner function with the input $M$, $\mathsf{S}_E$ defines a CF block corresponding to the outer function by using $\mathcal{RO}(M)$. That ensures the relation in Eq. (1) as long as the following events do not occur.

**E1:** For a path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$, there exists an $i$-th CF block defined by $\mathsf{S}_E$ such that the $(i+1)$-th CF block is defined by $\mathsf{S}_D$ or defined by $\mathsf{S}_E$ before the $i$-th CF block. The collision event is depicted in Fig. 8. In this event, if the CF blocks from the $(i+2)$-th to the last (in the outer function) are already defined, then $\mathsf{S}_E$ cannot obtain the message $M$ when defining the last CF block. Thus, Eq. (1) cannot be satisfied.

**E2:** When the path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$ is defined, the last CF block (in the outer function) is already defined. In this event, $\mathsf{S}_E$ cannot obtain the message $M$ when defining the last CF block. Thus, Eq. (1) cannot be satisfied.

**E3:** There exist collision paths $\left(0^{2n} \xrightarrow{M} S\right), \left(0^{2n} \xrightarrow{M^\dagger} S^\dagger\right) \in \mathcal{L}_{\mathsf{path}}$ such that $S = S^\dagger$. In this case, $\mathcal{RO}(M) \neq \mathcal{RO}(M^\dagger)$ is satisfied with a high probability (on the other hand, $\mathsf{EXEX\text{-}NI}^E(M) = \mathsf{EXEX\text{-}NI}^E(M^\dagger)$ is satisfied in the real world). In this event, Eq. (1) cannot be satisfied.

Assume that the above events do not occur. By $\neg\mathsf{E1}$, for any path $\left(0^{2n} \xrightarrow{M} S\right)$, the internal CF blocks are defined in order from the first to the last. Since

[8] For a query $(K, X)$ (resp. $(K, Y)$) to $\mathsf{S}_E$ (resp. $\mathsf{S}_D$), the response $Y$ (resp. $X$) is chosen uniformly at random from $\{0,1\}^n$ excluding previous ciphertext (resp. plaintext) blocks associated with the key $K$.

no collision path exists by $\neg\mathsf{E3}$, $\mathsf{S}$ can obtain the message $M$ leading to $S$. By $\neg\mathsf{E2}$, the path is defined before the $R$ block in the outer function, thus the simulator can define the $R$ block by using $\mathcal{RO}(M)$ so that Eq. (1) is satisfied. Thus, we have the indifferentiable bound by summing the upper-bounds of the probabilities for these events.

First, we analyze the event $\mathsf{E1}$ by using Fig. 8. If the $R$ block (b) defined by $\mathsf{S}_D$ before (a), then using the multi-collision technique for $W_i$, the number of the $R$ block (b) resulting in the $W_i$ value can be $n$, that is, the number of candidates for $X_{i+1}$ is at most $n$. Thus, for each $R$ block (a), the probability that $\mathsf{E1}$ occurs is at most $O(n/2^n)$. Similarly, if the $R$ block (b) is defined by $\mathsf{S}_D$ after (a), then using the multi-collision technique for $W_{i+1}$, for each $R$ block (b), the probability that $\mathsf{E1}$ occurs is at most $O(n/2^n)$. If the $R$ block (b) is defined by $\mathsf{S}_E$ before (a), then by the randomnesses of $Y_i$ and $Z_i$, the probability that the $R$ block (a) connects with one of candidates for the $R$ block (b) is $O(Q/2^{2n})$. Using the upper-bounds, we have $\mathsf{Pr}[\mathsf{E1}] \leq O(nQ/2^n)$.

Second, we analyze $\mathsf{E2}$. For each path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$, since $S$ is a $2n$-bit (almost) random value by two $R$ blocks, the probability that $S$ hits some $R$ block (in the outer function) is at most $O(Q/2^{2n})$. Hence, we have $\mathsf{Pr}[\mathsf{E2}] \leq O(Q^2/2^{2n})$.

Third, we analyze $\mathsf{E3}$. For a path $\left(0^{2n} \xrightarrow{M} S\right)$, since $S$ is a $2n$-bit (almost) random value by two $R$ blocks, the probability that there exists a collision path $\left(0^{2n} \xrightarrow{M^\dagger} S^\dagger\right)$ with $S = S^\dagger$ is at most $O(Q/2^{2n})$. Hence, we have $\mathsf{Pr}[\mathsf{E3}] \leq O(Q^2/2^{2n})$.

Finally, by these upper-bounds, we have the indifferentiable bound $O(nQ/2^n)$.

## 5 Proof of Theorem 1

In this proof, for the sake of simplicity, the padding function $\mathsf{pad}$ in $\mathsf{EXEX\text{-}NI}$ is omitted. Hence, an adversary $\mathbf{A}$ makes hash queries whose lengths are multiples of $m$. Since $\mathbf{A}$ can select any padding rule, this setting does not reduce the advantage of $\mathbf{A}$.

This proof considers three games. Game 0 is the real world, Game 1 is defined later, and Game 2 is the ideal world. In each game, an adversary $\mathbf{A}$ interacts with three oracles $(L, R_E, R_D)$. $L$ is a hash oracle, $R_E$ is an encryption oracle, and $R_D$ is a decryption oracle.

In the following analysis, we use Definitions 1, 2, 3, and 4 in Subsection 4.3.

### 5.1 Simulator

We define a simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$, where $\mathsf{S}_E^{\mathcal{RO}} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ simulates an encryption oracle $E$, and $\mathsf{S}_D^{\mathcal{RO}} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ simulates a decryption oracle $D$. In the real world, for a hash query, the response is defined by using $E$ via the $\mathsf{EXEX\text{-}NI}$ structure, thus the relation in Eq. (1) is satisfied. On the other hand, in the ideal world, for a hash query, the hash value

**Algorithm 1** Simulator $\mathsf{S}^{\mathcal{RO}} = (\mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$

```
Simulator $\mathsf{S}_E^{\mathcal{RO}}(K, X)$
 1: if $\mathsf{E}(K,X) \neq \lambda$ then return $\mathsf{E}(K,X)$
 2: $Y \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{E}(K,*)$; $\mathsf{E}(K,X) \leftarrow Y$; $\mathsf{D}(K,Y) \leftarrow X$
 3: $X_2 \leftarrow Y$; $K_2 \leftarrow \pi_1([K]^m) \| (\pi_2([K]_n) \oplus Y)$
 4: if $\mathsf{E}(K_2, X_2) = \lambda$ then
 5:     $Y_2 \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{E}(K_2,*)$; $\mathsf{E}(K_2,X_2) \leftarrow Y_2$; $\mathsf{D}(K_2,Y_2) \leftarrow X_2$
 6: end if
 7: $Y_2 \leftarrow \mathsf{E}(K_2,X_2)$; $Y_0 \leftarrow X$; $K_0 \leftarrow \pi_1^{-1}([K]^m) \| \pi_2^{-1}([K]_n \oplus X)$
 8: if $\mathsf{D}(K_0,Y_0) = \lambda$ then
 9:     $X_0 \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{D}(K_0,*)$; $\mathsf{E}(K_0,X_0) \leftarrow Y_0$; $\mathsf{D}(K_0,Y_0) \leftarrow X_0$
10: end if
11: if $\exists \left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{T}_{path}$ s.t. $S' \overset{in}{\rightsquigarrow} (K,X)$ then
12:     $M \leftarrow M' \| [K]^m$; $S \leftarrow \left(\pi_2([K_2]_n) \oplus Y_2\right) \| Y_2$; $\mathcal{T}_{path} \xleftarrow{\cup} \left(0^{2n} \xrightarrow{M} S\right)$
13:     $H \leftarrow \mathcal{RO}(M)$; $K_1' \leftarrow 0^{m-n} \| [S]_n \| [S]^n$; $X_1' \leftarrow \mathbf{1}$
14:     $Y_2' \leftarrow [H]_n$; $K_2' \leftarrow \pi_1([K_1']^m) \| \pi_2^{-1}([H]^n \oplus [H]_n)$; $X_2' \leftarrow [K_2']_n \oplus \pi_2([K_1']_n)$
15:     $Y_1' \leftarrow X_2'$; if $Y_1' \in \mathsf{E}(K_1',*)$ then abort
16:     $\mathsf{E}(K_1',X_1') \leftarrow Y_1'$; $\mathsf{D}(K_1',Y_1') \leftarrow X_1'$; if $Y_2' \in \mathsf{E}(K_2',*)$ then abort
17:     $\mathsf{E}(K_2',X_2') \leftarrow Y_2'$; $\mathsf{D}(K_2',Y_2') \leftarrow X_2'$
18: end if
19: return $\mathsf{E}(K,X)$

Simulator $\mathsf{S}_D^{\mathcal{RO}}(K, Y)$
 1: if $\mathsf{D}(K,Y) \neq \lambda$ then return $\mathsf{D}(K,Y)$
 2: $X \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{D}(K,*)$; $\mathsf{E}(K,X) \leftarrow Y$; $\mathsf{D}(K,Y) \leftarrow X$
 3: return $\mathsf{D}(K,Y)$
```

is defined by a monolithic random function $\mathcal{RO}$. Hence, the goal of a simulator is to simulate an ideal cipher so that the query-responses of $\mathcal{RO}$ and of $\mathsf{S}$ satisfy the relation in Eq. (1).

$\mathsf{S}$ is defined in Algorithm 1. $\mathsf{S}$ keeps $R$ blocks in lists $\mathsf{E}$ and $\mathsf{D}$ whose entries are initially empty strings. If an $R$ block $(K,X,Y)$ is defined where $\mathsf{S}_E(K,X) = Y$ or $\mathsf{S}_D(K,Y) = X$, then $Y$ is stored in $\mathsf{E}(K,X)$ and $X$ is stored in $\mathsf{D}(K,Y)$. $\mathsf{S}$ also keeps paths in $\mathcal{T}_{path}$, which initially keeps only a path $0^{2n} \xrightarrow{\lambda} 0^{2n}$. For $K \in \{0,1\}^k$, let $\mathsf{E}(K,*) = \{\mathsf{E}(K,X) | X \in \{0,1\}^n \wedge \mathsf{E}(K,X) \neq \lambda\}$ a set of all entries associated with $K$ in $\mathsf{E}$ and $\mathsf{D}(K,*) = \{\mathsf{D}(K,Y) | Y \in \{0,1\}^n \wedge \mathsf{D}(K,Y) \neq \lambda\}$ a set of all entries associated with $K$ in $\mathsf{D}$. For a query $(K,X)$ to $\mathsf{S}_E$, two ciphertext blocks $Y, Y_2$ and a plaintext block $X_0$ are defined, where the three tuples $(K_0,X_0,Y_0), (K,X,Y), (K_2,X_2,Y_2)$ offer two CF blocks: the first (resp. second) CF block consists of $(K_0,X_0,Y_0)$ and $(K,X,Y)$ (resp. $(K,X,Y)$ and $(K_2,X_2,Y_2)$). See Fig. 9. If there exists a path $\left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{T}_{path}$ such that $S' \overset{in}{\rightsquigarrow} (K,X)$, then a new path $\left(0^{2n} \xrightarrow{M} S\right)$ is defined by appending the CF block with $(K,X,Y)$ and $(K_2,X_2,Y_2)$ to the path, and is added to $\mathcal{T}_{path}$. To ensure the relation in Eq. (1), $R$ blocks $(K_1',X_1',Y_1')$ and $(K_2',X_2',Y_2')$ in the

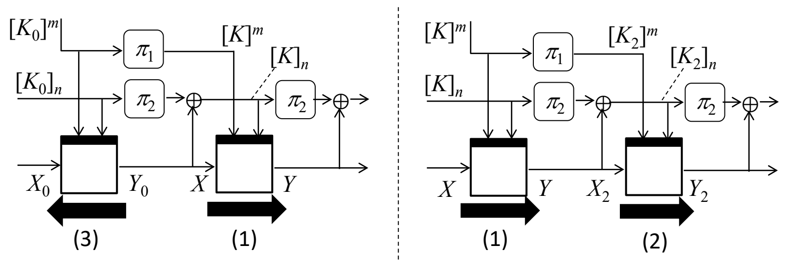

**Fig. 9.** Structures of $R$ blocks defined by $\mathsf{S}_E$. The $R$ block (1) is defined by a forward operation, then the $R$ block (2) is defined by a forward operation and the $R$ block (3) is defined by an inverse operation.

outer function that are connected with the path are defined by using $\mathcal{RO}(M)$. Since $Y_1'$ and $Y_2'$ are defined by $\mathcal{RO}$, there is a case that $\mathsf{E}(K_1', X_1')$ or $\mathsf{E}(K_2', X_2')$ is already defined, which does not occur in the real world. If this case occurs, then $\mathsf{S}$ aborts.

For each encryption query, $\mathsf{S}_E$ makes a query to $\mathcal{RO}$ at most once. Hence, the number of queries to $\mathcal{RO}$ is at most $p$. Regarding the running time, for each query, the number of steps of $\mathsf{S}$ is a constant. Hence, the running time is $t+O(p)$.

In $\mathsf{S}_E$, we call the operations to define $Y, Y_2, Y_1'$, and $Y_2'$ "forward operations," and the operation to define $X_0$ "inverse operation." In $\mathsf{S}_D$, we call the operation to define $X$ "inverse operation."

## 5.2 Main Part of the Proof

**Structure of the Proof.** As mentioned above, this proof consists of three games, Game 0, Game 1, and Game 2. Let $Gi$ be oracles in Game $i$. These oracles are defined as follows: Game 0: $G0 := (L, R_E, R_D) = (\mathsf{EXEX\text{-}NI}^E, E, D)$; Game 1: $G1 := (L, R_E, R_D) = (\mathsf{EXEX\text{-}NI}^{\mathsf{S}_E}, \mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$; Game 2: $G2 := (L, R_E, R_D) = (\mathcal{RO}, \mathsf{S}_E^{\mathcal{RO}}, \mathsf{S}_D^{\mathcal{RO}})$. Game 1 is a middle game between the real world (Game 0) and the ideal world (Game 2): for each hash query, the response is defined by using $\mathsf{S}_E$ via the structure of $\mathsf{EXEX\text{-}NI}$. In the following proof, in Game 2, after finishing all queries by $\mathbf{A}$, the procedure of $\mathsf{EXEX\text{-}NI}^{\mathsf{S}_E^{\mathcal{RO}}}(M)$ is performed for all hash queries $M$. Note that the additional procedure does not reduce the advantage of $\mathbf{A}$.

**Bad Events and Definitions.** We next define bad events in Game 1 and Game 2. Let $\mathcal{Q}_F$ (resp. $\mathcal{Q}_I$) be a list of $R$ blocks defined by forward (resp. inverse) operations in $\mathsf{S}$. Let $q_F = |\mathcal{Q}_F|$ and $q_I = |\mathcal{Q}_I|$. We assume that after an $R$ block $(K, X, Y)$ is stored in $\mathcal{Q}_F$ (resp. $\mathcal{Q}_I$), the $R$ block is not stored in $\mathcal{Q}_I$ (resp. $\mathcal{Q}_F$). Note that $\mathcal{L}_{\mathsf{block}} = \mathcal{Q}_F \cup \mathcal{Q}_I$ is satisfied.

– $\mathsf{mcoll_F}$: $\exists (K_1, X_1, Y_1), \ldots, (K_\mu, X_\mu, Y_\mu) \in \mathcal{Q}_F$ s.t. $\pi_2([K_1]_n) \oplus Y_1 = \cdots = \pi_2([K_\mu]_n) \oplus Y_\mu$.

– $\mathsf{mcoll}_\mathsf{I}$: $\exists (K_1, X_1, Y_1), \ldots, (K_\mu, X_\mu, Y_\mu) \in \mathcal{Q}_I$ s.t. $[K_1]_n \oplus X_1 = \cdots = [K_\mu]_n \oplus X_\mu$.
– $\mathsf{hit}_\mathsf{FI}$: $\exists (K, X, Y) \in \mathcal{Q}_F, (K', X', Y') \in \mathcal{Q}_I$ s.t. $\pi_2([K]_n) \oplus Y = [K']_n \wedge Y = X'$.
– $\mathsf{hit}_\mathsf{IV}$: $(\exists (K, X, Y) \in \mathcal{Q}_F$ s.t. $Y = 0^n \vee Y = \mathbf{1}) \vee (\exists (K, X, Y) \in \mathcal{Q}_I$ s.t. $X = 0^n \vee X = \mathbf{1})$.
– $\mathsf{hit}_\mathsf{XY}$: $\exists (K, X, Y) \in \mathcal{Q}_F \cup \mathcal{Q}_I$ s.t. $X = Y$.
– $\mathsf{coll}$: $\exists \left(0^{2n} \xrightarrow{M} S\right), \left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{L}_{path}$ s.t. $M \neq M' \wedge S = S'$.
– $\mathsf{hit}_\mathsf{Path}$: $\exists \left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_\mathsf{path}, (K, X, Y) \in \mathcal{Q}_F$ s.t. $S = [K]_n \| X$ or $[S]_n \| [S]^n = [K]_{2n}$, and the path is defined after the $R$ block.
– $\mathsf{Ecoll}$: $\mathsf{S}$ aborts.

For an event $\mathsf{e}$, the event in Game 1 (resp. Game 2) is denoted by $\mathsf{e}^1$ (resp. $\mathsf{e}^2$). Let $\mathsf{bad}^1 := \mathsf{mcoll}_\mathsf{F}^1 \vee \mathsf{mcoll}_\mathsf{I}^1 \vee \mathsf{hit}_\mathsf{FI}^1 \vee \mathsf{hit}_\mathsf{IV}^1 \vee \mathsf{hit}_\mathsf{XY}^1 \vee \mathsf{coll}^1 \vee \mathsf{hit}_\mathsf{Path}^1$. Let $\mathsf{bad}^2 := \mathsf{mcoll}_\mathsf{F}^2 \vee \mathsf{mcoll}_\mathsf{I}^2 \vee \mathsf{hit}_\mathsf{FI}^2 \vee \mathsf{hit}_\mathsf{IV}^2 \vee \mathsf{hit}_\mathsf{XY}^2 \vee \mathsf{coll}^2 \vee \mathsf{hit}_\mathsf{Path}^2 \vee \mathsf{Ecoll}^2$.

*Remark 1.* For the overview in Sec. 4.3, $\mathsf{hit}_\mathsf{FI}$ and $\mathsf{hit}_\mathsf{Path}$ (first condition) correspond to $\mathsf{E1}$, $\mathsf{hit}_\mathsf{Path}$ (second condition) corresponds to $\mathsf{E2}$, t $\mathsf{coll}$ corresponds to $\mathsf{E3}$, and $\mathsf{mcoll}_\mathsf{F}$ and $\mathsf{mcoll}_\mathsf{I}$ correspond to the multi-collision technique used in the analysis of $\mathsf{E1}$. Note that $\mathsf{hit}_\mathsf{IV}$, $\mathsf{hit}_\mathsf{XY}$, and $\mathsf{Ecoll}$ are not discussed in the overview. The following analyses show that these probabilities are negligible as long as other events do not occur.

**Upper-Bounding the Advantage Function.** To upper-bound the advantage, we use the following lemmas.

**Lemma 3.** *Let* $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_\mathsf{path}$ *be a path with* $\ell$ *CF blocks (i.e.,* $2\ell$ $R$ *blocks). In both Game 1 and Game 2, for any* $i \in [2\ell]$*, the* $i$*-th* $R$ *block is defined by forward operations and is defined after the* $(i-1)$*-th* $R$ *block as long as* $(\mathsf{hit}_\mathsf{FI} \vee \mathsf{hit}_\mathsf{IV} \vee \mathsf{hit}_\mathsf{XY} \vee \mathsf{hit}_\mathsf{Path})$ *does not occur.*

**Lemma 4.** *For* $V \in \{0,1\}^{2n}$ *and* $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_\mathsf{path}$ *such that* $V$ *is given before the path is defined, we have* $\mathsf{Pr}[S = V] \leq 1/(2^n - 3Q)^2$ *as long as* $(\mathsf{hit}_\mathsf{FI} \vee \mathsf{hit}_\mathsf{IV} \vee \mathsf{hit}_\mathsf{XY} \vee \mathsf{hit}_\mathsf{Path})$ *does not occur.*

**Lemma 5.** $\Pr[\mathbf{A}^{G0} = 1] = \Pr[\mathbf{A}^{G1} = 1 \mid \neg \mathsf{Ecoll}^1]$.

**Lemma 6.** $\mathsf{Pr}[\mathbf{A}^{G1} = 1 \mid \neg \mathsf{bad}^1 \wedge \neg \mathsf{Ecoll}^1] = \mathsf{Pr}[\mathbf{A}^{G2} = 1 \mid \neg \mathsf{bad}^2]$.

Lemma 3 is used in the proofs of Lemmas 4 and 6 and in the analyses of the bad events. Lemma 4 is used in the analyses of the bad events. Using Lemmas 5 and 6, we have

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathsf{S}}(\mathbf{A}) \leq \mathsf{Pr}[\mathsf{bad}^1 \mid \neg \mathsf{Ecoll}^1] + \mathsf{Pr}[\mathsf{bad}^2] \ ,[9]$$

where

$$\begin{aligned}\mathsf{Pr}[\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1] \le\ & \mathsf{Pr}[\mathsf{mcoll}^1_\mathsf{F} \mid \neg\mathsf{Ecoll}^1] + \mathsf{Pr}[\mathsf{mcoll}^1_\mathsf{I} \mid \neg\mathsf{Ecoll}^1] \\ & + \mathsf{Pr}[\mathsf{hit}^1_\mathsf{FI} \mid \neg\mathsf{mcoll}^1_\mathsf{F} \wedge \neg\mathsf{mcoll}^1_\mathsf{I} \wedge \neg\mathsf{Ecoll}^1] \\ & + \mathsf{Pr}[\mathsf{hit}^1_\mathsf{IV} \mid \neg\mathsf{Ecoll}^1] + \mathsf{Pr}[\mathsf{hit}^1_\mathsf{XY} \mid \neg\mathsf{Ecoll}^1] \\ & + \mathsf{Pr}[\mathsf{coll}^1 \mid \neg\mathsf{hit}^1_\mathsf{IV} \wedge \neg\mathsf{hit}^1_\mathsf{FI} \wedge \neg\mathsf{hit}^1_\mathsf{XY} \wedge \neg\mathsf{hit}^1_\mathsf{Path} \wedge \neg\mathsf{Ecoll}^1] \\ & + \mathsf{Pr}[\mathsf{hit}^1_\mathsf{Path} \mid \neg\mathsf{hit}^1_\mathsf{IV} \wedge \neg\mathsf{hit}^1_\mathsf{FI} \wedge \neg\mathsf{hit}^1_\mathsf{XY} \wedge \neg\mathsf{Ecoll}^1] \ ,\end{aligned}$$

and

$$\begin{aligned}\mathsf{Pr}[\mathsf{bad}^2] \le\ & \mathsf{Pr}[\mathsf{mcoll}^2_\mathsf{F}] + \mathsf{Pr}[\mathsf{mcoll}^2_\mathsf{I}] + \mathsf{Pr}[\mathsf{hit}^2_\mathsf{FI} \mid \neg\mathsf{mcoll}^2_\mathsf{F} \wedge \neg\mathsf{mcoll}^2_\mathsf{I}] \\ & + \mathsf{Pr}[\mathsf{hit}^2_\mathsf{IV}] + \mathsf{Pr}[\mathsf{hit}^2_\mathsf{XY}] + \mathsf{Pr}[\mathsf{coll}^2 \mid \neg\mathsf{hit}^2_\mathsf{IV} \wedge \neg\mathsf{hit}^2_\mathsf{FI} \wedge \neg\mathsf{hit}^2_\mathsf{XY} \wedge \neg\mathsf{hit}^2_\mathsf{Path}] \\ & + \mathsf{Pr}[\mathsf{hit}^2_\mathsf{Path} \mid \neg\mathsf{hit}^2_\mathsf{IV} \wedge \neg\mathsf{hit}^2_\mathsf{FI} \wedge \neg\mathsf{hit}^2_\mathsf{XY}] + \mathsf{Pr}[\mathsf{Ecoll}^2 \mid \neg\mathsf{hit}^2_\mathsf{Path}] \ .\end{aligned}$$

These upper-bounds are given in the following, which gives

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathsf{S}}(\mathbf{A}) \le 2^{n+2} \cdot \binom{3Q}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu} + \frac{6\mu Q + 19Q}{2^n - 3Q} + \frac{22Q^2}{(2^n - 3Q)^2} \ .$$

**Upper-Bounding $\mathsf{Pr}[\mathsf{mcoll}^1_\mathsf{F} \mid \neg\mathsf{Ecoll}^1]$.** For each $(K, X, Y) \in \mathcal{Q}_F$, since $Y$ is chosen uniformly at random from at least $2^n - 3Q$ elements in $\{0,1\}^n$, for some $V \in \{0,1\}^n$, we have $\mathsf{Pr}[\pi_2([K]_n) \oplus Y = V] \le 1/(2^n - 3Q)$. Hence, we have $\mathsf{Pr}[\mathsf{mcoll}^1_\mathsf{F} \mid \neg\mathsf{Ecoll}^1] \le 2^n \cdot \binom{q_F}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{mcoll}^2_\mathsf{F}]$.** By the same analysis as $\mathsf{Pr}[\mathsf{mcoll}^1_\mathsf{F} \mid \neg\mathsf{Ecoll}^1]$, we have $\mathsf{Pr}[\mathsf{mcoll}^2_\mathsf{F}] \le 2^n \cdot \binom{q_F}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu}$.

---

[9] The inequation is obtained by

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H},\mathsf{S}}(\mathbf{A}) = \Pr[\mathbf{A}^{G0} = 1] - \Pr[\mathbf{A}^{G2} = 1] = \Pr[\mathbf{A}^{G1} = 1 \mid \neg\mathsf{Ecoll}^1] - \Pr[\mathbf{A}^{G2} = 1] \\ &\le \Big(\Pr[\mathbf{A}^{G1} = 1 \wedge \mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1] + \Pr[\mathbf{A}^{G1} = 1 \wedge \neg\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1]\Big) \\ &\qquad - \Big(\Pr[\mathbf{A}^{G2} = 1 \wedge \mathsf{bad}^2] + \Pr[\mathbf{A}^{G2} = 1 \wedge \neg\mathsf{bad}^2]\Big) \\ &= \Big(\Pr[\mathbf{A}^{G1} = 1 \mid \mathsf{bad}^1 \wedge \neg\mathsf{Ecoll}^1] \cdot \mathsf{Pr}[\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1] \\ &\quad + \underbrace{\Pr[\mathbf{A}^{G1} = 1 \mid \neg\mathsf{bad}^1 \wedge \neg\mathsf{Ecoll}^1]}_{=\mathsf{Pr}[\mathbf{A}^{G2}=1 \mid \neg\mathsf{bad}^2]} \cdot \underbrace{\mathsf{Pr}[\neg\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1]}_{=1-\mathsf{Pr}[\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1]}\Big) \\ &\qquad - \Big(\Pr[\mathbf{A}^{G2} = 1 \mid \mathsf{bad}^2] \cdot \mathsf{Pr}[\mathsf{bad}^2] + \Pr[\mathbf{A}^{G2} = 1 \mid \neg\mathsf{bad}^2] \cdot \mathsf{Pr}[\neg\mathsf{bad}^2]\Big) \\ &= \Big(\Pr[\mathbf{A}^{G1} = 1 \mid \mathsf{bad}^1 \wedge \neg\mathsf{Ecoll}^1] - \Pr[\mathbf{A}^{G2} = 1 \mid \neg\mathsf{bad}^2]\Big) \cdot \mathsf{Pr}[\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1] \\ &\qquad + \Big(\Pr[\mathbf{A}^{G2} = 1 \mid \neg\mathsf{bad}^2] - \Pr[\mathbf{A}^{G2} = 1 \mid \mathsf{bad}^2]\Big) \cdot \mathsf{Pr}[\mathsf{bad}^2] \\ &\le \mathsf{Pr}[\mathsf{bad}^1 \mid \neg\mathsf{Ecoll}^1] + \mathsf{Pr}[\mathsf{bad}^2] \ ,\end{aligned}$$

**Upper-Bounding $\mathsf{Pr}[\mathsf{mcoll}_{\mathsf{I}}^1 \mid \neg\mathsf{Ecoll}^1]$.** For each $(K, X, Y) \in \mathcal{Q}_I$, since $X$ is chosen uniformly at random from at least $2^n - 3Q$ elements in $\{0,1\}^n$, for some $V \in \{0,1\}^n$, we have $\mathsf{Pr}[[K]_n \oplus X = V] \le 1/(2^n - 3Q)$. Hence, we have $\mathsf{Pr}[\mathsf{mcoll}_{\mathsf{I}}^1 \mid \neg\mathsf{Ecoll}^1] \le 2^n \cdot \binom{q_I}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{mcoll}_{\mathsf{I}}^2]$.** By the same analysis as $\mathsf{Pr}[\mathsf{mcoll}_{\mathsf{I}}^1 \mid \neg\mathsf{Ecoll}^1]$, we have $\mathsf{Pr}[\mathsf{mcoll}_{\mathsf{I}}^2] \le 2^n \cdot \binom{q_I}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^{\mu}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{FI}}^1 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^1 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^1 \wedge \neg\mathsf{Ecoll}^1]$.** For $R$ blocks $(K, X, Y)$ and $(K', X', Y')$, if $(K', X', Y')$ is defined after $(K, X, Y)$, then the relation is denoted by $(K, X, Y) \lhd (K', X', Y')$. We consider the following two cases.

- $\mathsf{hit}_{\overleftarrow{\mathsf{FI}}}^1$: $\exists (K, X, Y) \in \mathcal{Q}_F, (K', X', Y') \in \mathcal{Q}_I$ s.t. $(K, X, Y) \lhd (K', X', Y') \wedge \pi_2([K]_n) \oplus Y = [K']_n \wedge Y = X'$.
- $\mathsf{hit}_{\overrightarrow{\mathsf{FI}}}^1$: $\exists (K, X, Y) \in \mathcal{Q}_F, (K', X', Y') \in \mathcal{Q}_I$ s.t. $(K', X', Y') \lhd (K, X, Y) \wedge \pi_2([K]_n) \oplus Y = [K']_n \wedge Y = X'$.

We analyze $\mathsf{hit}_{\overleftarrow{\mathsf{FI}}}^1$. For an input $(K', Y')$ in $\mathcal{Q}_I$, by $\neg\mathsf{mcoll}_{\mathsf{F}}^1$, the number of tuples $(K, X, Y) \in \mathcal{Q}_F$ such that $\pi_2([K]_n) \oplus Y = [K']_n$ is satisfied is at most $\mu - 1$, thus the probability that $X'$ equals one of the $(\mu - 1)$ ciphertext blocks is at most $(\mu - 1)/(2^n - 3Q)$. Summing the probability for each element in $\mathcal{Q}_I$, we have $\mathsf{Pr}[\mathsf{hit}_{\overleftarrow{\mathsf{FI}}}^1 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^1 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^1 \wedge \neg\mathsf{Ecoll}^1] \le (\mu - 1)q_I/(2^n - 3Q)$.

For $\mathsf{hit}_{\overrightarrow{\mathsf{FI}}}^1$, the analysis is the same as that of $\mathsf{hit}_{\overleftarrow{\mathsf{FI}}}^1$. Using the condition $\neg\mathsf{mcoll}_{\mathsf{I}}^1$, we have $\mathsf{Pr}[\mathsf{hit}_{\overrightarrow{\mathsf{FI}}}^1 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^1 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^1 \wedge \neg\mathsf{Ecoll}^1] \le (\mu - 1)q_F/(2^n - 3Q)$.

By $q_F + q_I \le 3Q$, we have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{FI}}^1 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^1 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^1 \wedge \neg\mathsf{Ecoll}^1] \le \frac{3(\mu-1)Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{FI}}^2 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^2 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^2]$.** As the analysis of $\mathsf{Pr}[\mathsf{hit}_{\mathsf{FI}}^1 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^1 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^1 \wedge \neg\mathsf{Ecoll}^1]$, using the conditions $\neg\mathsf{mcoll}_{\mathsf{F}}^2$ and $\neg\mathsf{mcoll}_{\mathsf{I}}^2$, we have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{FI}}^2 \mid \neg\mathsf{mcoll}_{\mathsf{F}}^2 \wedge \neg\mathsf{mcoll}_{\mathsf{I}}^2] \le \frac{3(\mu-1)Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^1 \mid \neg\mathsf{Ecoll}^1]$** For each $(K, X, Y) \in \mathcal{Q}_F$ (resp. $(K, X, Y) \in \mathcal{Q}_I$), $Y$ (resp. $X$ ) is chosen uniformly at random from at least $2^n - 3Q$ elements in $\{0,1\}^n$. Thus, we have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^1 \mid \neg\mathsf{Ecoll}^1] \le \frac{6Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^2]$.** The analysis is the same as that of $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^1 \mid \neg\mathsf{Ecoll}^1]$. We have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^2] \le \frac{6Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{XY}}^1 \mid \neg\mathsf{Ecoll}^1]$** For each $(K, X, Y) \in \mathcal{Q}_F$ (resp. $(K, X, Y) \in \mathcal{Q}_I$), $Y$ (resp. $X$) is chosen uniformly at random from at least $2^n - 3Q$ elements in $\{0,1\}^n$. Thus, we have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{XY}}^1 \mid \neg\mathsf{Ecoll}^1] \le \frac{3Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{hit}_{\mathsf{XY}}^2]$.** The analysis is the same as that of $\mathsf{Pr}[\mathsf{hit}_{\mathsf{IV}}^1 \mid \neg\mathsf{Ecoll}^1]$. We have $\mathsf{Pr}[\mathsf{hit}_{\mathsf{XY}}^2] \le \frac{3Q}{2^n - 3Q}$.

**Upper-Bounding $\mathsf{Pr}[\mathsf{coll}^1 \mid \neg\mathsf{hit}_{\mathsf{IV}}^1 \wedge \neg\mathsf{hit}_{\mathsf{FI}}^1 \wedge \neg\mathsf{hit}_{\mathsf{XY}}^1 \wedge \neg\mathsf{hit}_{\mathsf{Path}}^1 \wedge \neg\mathsf{Ecoll}^1]$.** Assume that $(\mathsf{hit}_{\mathsf{IV}}^1 \vee \mathsf{hit}_{\mathsf{FI}}^1 \vee \mathsf{hit}_{\mathsf{XY}}^1 \vee \mathsf{hit}_{\mathsf{Path}}^1 \vee \mathsf{Ecoll}^1)$ is not satisfied. Fix a path $\left(0^{2n} \xrightarrow{M} S\right) \in$

$\mathcal{L}_{\mathsf{path}}$, and assume that $\mathsf{coll}^1$ has not occurred before the path. Then, for each path $\left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{L}_{\mathsf{path}}$ that was defined before $\left(0^{2n} \xrightarrow{M} S\right)$, we have $\mathsf{Pr}[S = S'] \leq 1/(2^n - 3Q)^2$ by Lemma 4. Summing the probability for each pair of paths where $|\mathcal{L}_{\mathsf{path}}| \leq 2Q$, we have $\mathsf{Pr}[\mathsf{coll}^1 \mid \neg\mathsf{hit}^1_{\mathsf{IV}} \wedge \neg\mathsf{hit}^1_{\mathsf{FI}} \wedge \neg\mathsf{hit}^1_{\mathsf{XY}} \wedge \neg\mathsf{hit}^1_{\mathsf{Path}} \wedge \neg\mathsf{Ecoll}^1] \leq \binom{2Q}{2} \cdot \frac{1}{(2^n-3Q)^2} \leq \frac{2Q^2}{(2^n-3Q)^2}$.

**Upper-Bounding** $\mathsf{Pr}[\mathsf{coll}^2 \mid \neg\mathsf{hit}^2_{\mathsf{IV}} \wedge \neg\mathsf{hit}^2_{\mathsf{FI}} \wedge \neg\mathsf{hit}^2_{\mathsf{XY}} \wedge \neg\mathsf{hit}^2_{\mathsf{Path}}]$ The analysis is the same as that of $\mathsf{Pr}[\mathsf{coll}^1 \mid \neg\mathsf{hit}^1_{\mathsf{IV}} \wedge \neg\mathsf{hit}^1_{\mathsf{FI}} \wedge \neg\mathsf{hit}^1_{\mathsf{XY}} \wedge \neg\mathsf{hit}^1_{\mathsf{Path}} \wedge \neg\mathsf{Ecoll}^1]$. By Lemma 4, we have $\mathsf{Pr}[\mathsf{coll}^2 \mid \neg\mathsf{hit}^2_{\mathsf{IV}} \wedge \neg\mathsf{hit}^2_{\mathsf{FI}} \wedge \neg\mathsf{hit}^2_{\mathsf{XY}} \wedge \neg\mathsf{hit}^2_{\mathsf{Path}}] \leq \frac{2Q^2}{(2^n-3Q)^2}$.

**Upper-Bounding** $\mathsf{Pr}[\mathsf{hit}^1_{\mathsf{Path}} \mid \neg\mathsf{hit}^1_{\mathsf{IV}} \wedge \neg\mathsf{hit}^1_{\mathsf{FI}} \wedge \neg\mathsf{hit}^1_{\mathsf{XY}} \wedge \neg\mathsf{Ecoll}^1]$**.** Fix $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$ and $(K, X, Y) \in \mathcal{Q}_F$ such that $\mathsf{hit}^1_{\mathsf{Path}}$ has not occurred and the $R$ block was defined before the path is defined. By Lemma 4, we have $\mathsf{Pr}[S = [K]_n \| X] \leq 1/(2^n - 3Q)^2$ and $\mathsf{Pr}[[S]_n \| [S]^n = [K]_{2n}] \leq 1/(2^n - 3Q)^2$. Summing the probabilities for each path and $R$ block, we have $\mathsf{Pr}[\mathsf{hit}^1_{\mathsf{Path}} \mid \neg\mathsf{hit}^1_{\mathsf{IV}} \wedge \neg\mathsf{hit}^1_{\mathsf{FI}} \wedge \neg\mathsf{hit}^1_{\mathsf{XY}} \wedge \neg\mathsf{Ecoll}^1] \leq 2Q \cdot 2Q \cdot \frac{2}{(2^n-3Q)^2} \leq \frac{8Q^2}{(2^n-3Q)^2}$.

**Upper-Bounding** $\mathsf{Pr}[\mathsf{hit}^2_{\mathsf{Path}} \mid \neg\mathsf{hit}^2_{\mathsf{IV}} \wedge \neg\mathsf{hit}^2_{\mathsf{FI}} \wedge \neg\mathsf{hit}^2_{\mathsf{XY}}]$**.** The analysis is the same as that of $\mathsf{Pr}[\mathsf{hit}^1_{\mathsf{Path}} \mid \neg\mathsf{hit}^1_{\mathsf{IV}} \wedge \neg\mathsf{hit}^1_{\mathsf{FI}} \wedge \neg\mathsf{hit}^1_{\mathsf{XY}} \wedge \neg\mathsf{Ecoll}^1]$. Thus we have $\mathsf{Pr}[\mathsf{hit}^2_{\mathsf{Path}} \mid \neg\mathsf{hit}^2_{\mathsf{IV}} \wedge \neg\mathsf{hit}^2_{\mathsf{FI}} \wedge \neg\mathsf{hit}^2_{\mathsf{XY}}] \leq \frac{8Q^2}{(2^n-3Q)^2}$.

**Upper-Bounding** $\mathsf{Pr}[\mathsf{Ecoll}^2 \mid \neg\mathsf{hit}^2_{\mathsf{Path}}]$**.** For each process of $\mathsf{S}^{\mathcal{RO}}_E(K, X)$, if there exists a path $\left(0^{2n} \xrightarrow{M'} S'\right) \in \mathcal{T}_{path}$ s.t. $S' \overset{in}{\rightsquigarrow} (K, X)$, then by $\neg\mathsf{hit}^2_{\mathsf{Path}}$, $\mathsf{E}[K'_1, *] = \lambda$ and $\mathsf{E}[K'_2, *] = \lambda$ are satisfied. On the other hand, there is a case that $K'_1 = K'_2$ and $X'_1 \neq X'_2$ are satisfied, but $Y'_1 = Y'_2$ is satisfied. For each path, the collision probability is $1/2^n$, since $Y'_1$ and $Y'_2$ are defined by $\mathcal{RO}$. We thus have $\mathsf{Pr}[\mathsf{Ecoll}^2 \mid \mathsf{hit}^2_{\mathsf{Path}}] \leq \frac{Q}{2^n}$.

## 5.3 Proof of Lemma 3

Consider Game $j$ where $j \in [2]$. Assume that $(\mathsf{hit}^j_{\mathsf{FI}} \vee \mathsf{hit}^j_{\mathsf{IV}} \vee \mathsf{hit}^j_{\mathsf{XY}} \vee \mathsf{hit}^j_{\mathsf{Path}})$ does not occur. Consider a path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$ with $\ell$ CF block ($2\ell$ $R$ blocks). By the definition of $\mathsf{S}_E$, after an $R$ block is defined by a forward operation, the next $R$ block is immediately defined by a forward operation, and the former $R$ block is defined by an inverse operation. All $R$ blocks in the path are defined by forward operations by $\neg\mathsf{hit}^j_{\mathsf{FI}}$ and $\neg\mathsf{hit}^j_{\mathsf{IV}}$, and for each $i \in [2\ell - 1]$, the $(i + 1)$-th $R$ block in the path is defined after the $i$-th $R$ block by $\neg\mathsf{hit}^j_{\mathsf{XY}}$ and $\neg\mathsf{hit}^j_{\mathsf{Path}}$.

## 5.4 Proof of Lemma 4

Consider Game $j$ where $j \in [2]$. Assume that $(\mathsf{hit}^j_{\mathsf{FI}} \vee \mathsf{hit}^j_{\mathsf{IV}} \vee \mathsf{hit}^j_{\mathsf{XY}} \vee \mathsf{hit}^j_{\mathsf{Path}})$ does not occur. Fix a value $V \in \{0, 1\}^{2n}$. For a path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$, by Lemma 3, the last CF block in the path consists of two $R$ blocks defined by forward operations and the outputs are differently sampled. Since the outputs are chosen uniformly at random from at least $2^n - 3Q$ elements in $\{0,1\}^n$, we have $\mathsf{Pr}[S = V] \leq 1/(2^n - 3Q)^2$.

### 5.5 Proof of Lemma 5

Since outputs of $\mathcal{RO}$ are chosen uniformly at random from $\{0,1\}^{2n}$, a collision occurs in entries with the same key. In other words, $\mathsf{S}$ behaves as an ideal cipher if and only if such collision does not occur, i.e., $\mathsf{Ecoll}^1 = \mathsf{false}$. Thus, we have $\Pr[\mathbf{A}^{G0} = 1] = \Pr[\mathbf{A}^{G1} = 1 \mid \neg\mathsf{Ecoll}_1]$.

### 5.6 Proof of Lemma 6

We show that $\mathsf{Pr}[\mathbf{A}^{G1} = 1 \mid \neg\mathsf{bad}^1 \wedge \neg\mathsf{Ecoll}^1] = \mathsf{Pr}[\mathbf{A}^{G2} = 1 \mid \neg\mathsf{bad}^2]$ (Game 1 and Game 2 are indistinguishable). In this proof, we need to show that the structural difference of $L$ gives no advantage to $\mathbf{A}$. The difference is: $L = \mathsf{EXEX\text{-}NI}^{R_E}$ (Game 1) and $L = \mathcal{RO}$ (Game 2), thus with the following two points, the equivalence is ensured.

1. In Game 1, for any hash query $M$, the response is equal to $\mathcal{RO}(M)$.
2. In Game 2, $L$ and $R$ are consistent as in Game 1 with respect to the structure of $\mathsf{EXEX\text{-}NI}$, that is, for any $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{path}$, $\left(S' \xrightarrow{M'} H\right) \in \mathcal{L}_{block}$ such that $S \overset{out}{\rightsquigarrow} (S', M')$, $H = L(M)$ is satisfied.

The following lemma ensures these two points. Hence, we have

$$\mathsf{Pr}[\mathbf{A}^{G1} = 1 \mid \neg\mathsf{bad}_1 \wedge \neg\mathsf{Ecoll}^1] = \mathsf{Pr}[\mathbf{A}^{G2} = 1 \mid \neg\mathsf{bad}_2] \ .$$

**Lemma 7.** *In Game $j$ $(= 1, 2)$, the following is satisfied as long as $\mathsf{bad}^j$ does not occur: $\forall\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}, \left(S' \xrightarrow{M'} H\right) \in \mathcal{L}_{\mathsf{block}}$ s.t. $S \overset{out}{\rightsquigarrow} (S', M')$: $H = \mathcal{RO}(M)$.*

*Proof (Lemma 7).* In Game $j$ $(j = 1, 2)$, for a path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$ and a CF block $\left(S' \xrightarrow{M'} H\right) \in \mathcal{L}_{\mathsf{block}}$ such that $S \overset{out}{\rightsquigarrow} (S', M')$, by $\neg\mathsf{hit}^j_{\mathsf{Path}}$, the CF block $\left(S' \xrightarrow{M'} H\right)$ is defined after the path $\left(0^{2n} \xrightarrow{M} S\right) \in \mathcal{L}_{\mathsf{path}}$ is defined. By Lemma 3, all $R$ blocks in the path are defined by forward operations in order from the first to the last. By $\neg\mathsf{coll}^j$, there is no collision path leading to $S$. Thus, by the definition of $\mathsf{S}_E$, $H$ is defined as $H = \mathcal{RO}(M)$. □

## 6 EXEX-I: Low Memory Collision Resistant DBL HF

In this section, we consider relaxing the security goal to the collision resistance rather than the indifferentiability. We design $\mathsf{EXEX\text{-}I}$ in Sect. 6.1. An overview of its collision resistance is given in Sect. 6.2.

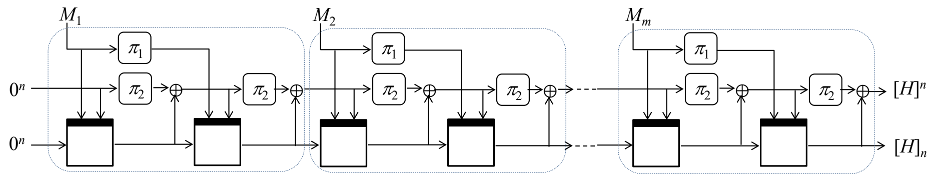

**Fig. 10.** Hash Function EXEX-I.

## 6.1 Specification of EXEX-I

Like EXEX-NI, we design EXEX-I by iterating EXEX defined in Sect. 4.1, but unlike EXEX-NI, we relax the security goal to be collision resistance. The change of the security goal enables us to remove the outer function of EXEX-NI, which was introduced to achieve indifferentiability. Removal of the outer function relaxes the condition for the key length and improves the efficiency.

We define EXEX-I : $\{0,1\}^* \to \{0,1\}^{2n}$. Let pad : $\{0,1\}^* \to (\{0,1\}^m)^*$ be an injective padding function, e.g., one-zero padding $\mathsf{pad}(M) = M\|10^w$ where $w = m - 1 - |M| \bmod m$. The construction of EXEX-I is depicted in Fig. 10.

– $\mathsf{EXEX\text{-}I}^E(M)$:
  1. $S_0 \leftarrow 0^{2n}$; $M_1, M_2, \cdots, M_\ell \xleftarrow{m} \mathsf{pad}(M)$
  2. for $i = 1, \ldots, \ell$ do $S_i \leftarrow \mathsf{EXEX}^E(S_{i-1}, M_i)$
  3. return $S_\ell$

## 6.2 Collision Resistance of EXEX-I

The following gives an upper-bound for the collision resistance of EXEX-I.

**Theorem 2.** *Let $\mu$ be any positive integer. For any adversary $\mathbf{A}$ making $Q$ queries, we have*

$$\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{EXEX\text{-}I}}(\mathbf{A}) \le \frac{0.5Q^2}{(2^n - 3Q)^2} + \frac{3\mu Q + 3Q}{2^n - 3Q} + 2^n \cdot \binom{3Q}{\mu} \cdot \left(\frac{1}{2^n - 3Q}\right)^\mu .$$

EXEX-I is equal to the inner function of EXEX-NI. Since a collision of the inner function breaks the indifferentiability of EXEX-NI, the proof of the collision resistance of EXEX-I is equal to (some part of) the proof of the indifferentiability of EXEX-NI. Using the proof, we can show that EXEX-I is collision resistant up to the bound given in the theorem.

## 7 Hardware Performance Evaluation

We compare the hardware performances of EXEX-NI, EXEX-I, and Romulus-H under the same design policy. We instantiate the candidates with SKINNY-128-384 used in the first literature on Romulus-H [13]. [10] All the instantiations achieve the same data rate, i.e., processing a 256-bit message block with two TBC calls. EXEX-NI and EXEX-I can choose SKINNY-128-256 for an even smaller implementation, which we will discuss later in Sect.7.4.

### 7.1 SKINNY-128-384 Implementation

The baseline design is Naito et al.'s conventional SKINNY-128-384 implementation [25] based on the common byte-serial architecture [3]. The state array is a 128-bit register arranged in a 2-dimensional array, which integrates the linear operations, i.e., MixColumns and ShiftRows. It finishes the round function in 21 cycles, and the entire SKINNY-128-384 in 1,176 (=21 $\times$ 56) cycles. We use a scan flip-flop, a special-purpose register with a built-in 2-way selector, for efficiently implementing the 2-dimensional array [3, 25].

The 384-bit tweakey comprises independently scheduled 128-bit chunks, referred to as $\mathsf{TK}_1$, $\mathsf{TK}_2$, and $\mathsf{TK}_3$ hereafter. We assign $\mathsf{TK}_1$ for storing the state mixed with TBC outputs and $\mathsf{TK}_2||\mathsf{TK}_3$ for storing 256-bit message blocks. By following the baseline implementation, we again use the 2-dimensional arrays for storing the tweakey, namely the $\mathsf{TK}_1$, $\mathsf{TK}_2$, and $\mathsf{TK}_3$ arrays. Those arrays efficiently realize a serial byte scanning and the byte-wise permutation for the tweakey schedule [25][11].

### 7.2 Hardware Implementation of EXEX-NI and EXEX-I

We decompose EXEX-NI into the four operations as shown in Fig. 11-(left): (C1) a TBC call, (C2) transferring a TBC output to $\mathsf{TK}_1$, (C3) feeding a 256-bit new message block to $\mathsf{TK}_2||\mathsf{TK}_3$, and (C4) organizing data for the nested processing. The circuit in Fig. 11-(right) implements the four basic operations and can hash a long message by dispatching the operations in an appropriate order. Starting from the baseline SKINNY-128-384 implementation, we added some 8-bit selector, XOR, and AND gates for managing data transmission between the arrays to support the (C2) and (C4) operations. The EXEX-I implementation is the EXEX-NI implementation without the (C4) operation, and we can remove some gates from the datapath in Fig. 11-(right).

**$\pi_1$ and $\pi_2$ for tweakey schedule.** We use the linear operations $\pi_1$ and $\pi_2$ (see Fig. 7) to achieve better performance by eliminating the inverse tweakey schedule. A lightweight TBC implementation commonly uses an on-the-fly tweakey schedule to avoid storing the round keys. As a drawback, we lose the original tweakey by updating it in place, which is a problem for a mode of operation that uses the same tweakey in the following operations. The previous SKINNY-128-384

[10] In the updates for the NIST LWC final round, the Romulus team decided to use a reduced-round variant called SKINNY-128-384+. We discuss the impact of this change in Sect. 7.4.

[11] For efficiency, we remove a built-in arithmetic counter in the previous tweakey arrays needed for an AEAD mode of operation [25].

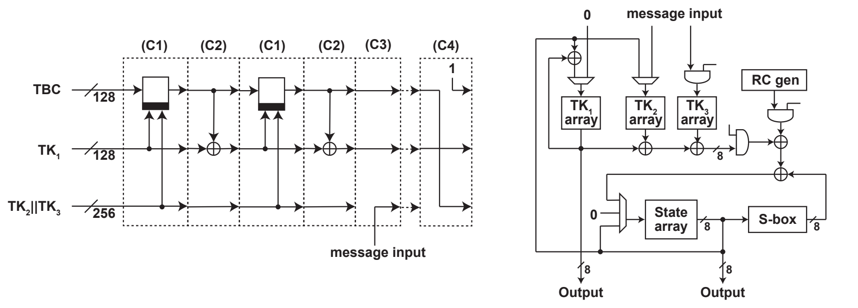

**Fig. 11.** (Left) Decomposition of EXEX-NI into basic operations and (Right) the datapath architecture realizing those operations

implementation addressed this issue by integrating the inverse tweakey schedule in the tweakey arrays at the cost of additional hardware resources [25].

A more sophisticated approach is integrating the tweakey schedule into a mode of operation so that we can continue without recovering the original tweakey [26]. EXEX-NI and EXEX-I support this optimization by assigning $\pi_1$ as $\mathsf{TK}_1$'s schedule and assigning $\pi_2$ as $\mathsf{TK}_2$'s and $\mathsf{TK}_3$'s schedule combined. As a result, by skipping the inverse operation after an on-the-fly key schedule, the (C2) operation implicitly executes $\pi_2$ and $\pi_3$ after a TBC call. This allows us to remove the circuits for the inverse tweakey operations from the tweakey arrays. [12]

## 7.3 Hardware Implementation of Romulus-H

Our Romulus-H design is based on the same SKINNY-128-384 implementation and has a similar architecture and components (the state and tweakey arrays). Romulus-H needs additional $2 \times 128$ bits of memory, and we realize them using a set of 128-bit shift registers, namely $\mathsf{SR}_0$ and $\mathsf{SR}_1$.

We decompose Romulus-H into several basic operations in Fig. 12-(left): (D1) a TBC call, (D2) processing the first TBC output, (D3) processing the second TBC output, and (D4) feeding a 256-bit new message block to $\mathsf{TK}_2||\mathsf{TK}_3$. Fig. 12-(left) also illustrates how we manage data between the memory elements: we use $\mathsf{SR}_0$ for storing the previous TBC input for feed-forward and use $\mathsf{SR}_1$ for preserving a derivative of the first TBC call during the second one.

Fig. 12-(right) shows the corresponding datapath. The major additions to the baseline SKINNY-128-384 implementation are $\mathsf{SR}_0$ and $\mathsf{SR}_1$ implemented as simple 8-bit width and 16-stage shift registers. We also added some 8-bit logic gates for enabling data transmission between the memory elements.

[12] The conventional $\mathsf{TK}_1$–$\mathsf{TK}_3$ arrays integrates circuits for inverse tweakey operations (the inverse LFSRs and inverse byte permutation) [25]. We remove these circuits along with selectors for switching between the datapaths. These circuits are intact in our Romulus-H implementation because it requires the inverse operations.

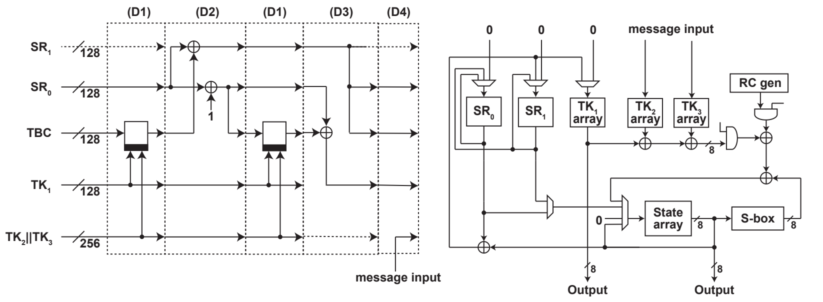

**Fig. 12.** (Left) Decomposition of Romulus-H into basic operations and (Right) the datapath architecture realizing those operations

## 7.4 Performance Evaluation and Comparison

**Implementation and Evaluation Procedure.** We implement the designs at the register-transfer level; we explicitly instantiate the standard cells only for the scan flip-flops [22]. We synthesize the designs using Synopsys Design Compiler with the NanGate 45-nm standard cell library [27]. Table 3 summarizes the circuit areas of our EXEX-NI, EXEX-I, and Romulus-H implementations in NAND gate equivalent (GE), along with its breakdown to major components that we preserved during the synthesis.

**Comparison.** The EXEX-NI and EXEX-I implementations are smaller than that of Romulus-H, as shown in Table 3. More specifically, the EXEX-NI implementation is smaller by 2,262 GE and is only 68% of the Romulus-H implementation. The main reason is the memory sizes: Romulus-H's additional memory, implemented as the shift registers $\mathsf{SR}_0$ and $\mathsf{SR}_1$, consumes roughly 1,500 GE, as shown in Table 3. [13] Another reason is the inverse tweakey schedule we discussed in Sect. 7.2. Eliminating the inverse tweakey schedule makes each tweakey array smaller by roughly 250 GE, resulting in the total reduction of 750 GE. The difference between the EXEX-NI and EXEX-I implementations is only 48 GE that corresponds to several 8-bit width logic gates for the (C4) operation; EXEX-I's main advantage is speed. The S-box circuit, composed of eight XORs and NORs, occupies roughly 30 GE which is negligible compared to the registers.

**Missing Parallelism.** The conventional schemes including Romulus-H can run up to two TBCs in parallel, which is impossible with EXEX-NI and EXEX-I that serializes the consecutive TBCs for smaller memory. Missing parallelism has a negligible impact on lightweight implementations because parallel execution needs double the hardware resources. On the other hand, in high-speed implementation with sufficiently many resources available, Romulus-H and other conventional schemes can have higher efficiency (i.e., throughput/area) by pipelining the consecutive TBCs.

[13] The per-bit cost of $\mathsf{SR}_0$ and $\mathsf{SR}_1$ is smaller than those of the state/tweakey arrays because $\mathsf{SR}_0$ and $\mathsf{SR}_1$ use a simple flip-flop instead of a scan flip-flop.

**Table 3.** Circuit-area comparison of EXEX-NI, EXEX-I, and Romulus-H instantiated with SKINNY-128-384 in gate-equivalent (GE)

| Target | Total | State array | $\mathsf{TK}_1$ array | $\mathsf{TK}_2$ array | $\mathsf{TK}_3$ array | $\mathsf{SR}_0$ | $\mathsf{SR}_1$ |
|---|---|---|---|---|---|---|---|
| EXEX-NI | 4,755 | 1,078 | 1,007 | 1,019 | 1,019 | — | — |
| EXEX-I | 4,707 | 1,077 | 1,007 | 1,019 | 1,019 | — | — |
| Romulus-H | 7,017 | 1,078 | 1,231 | 1,270 | 1,271 | 743 | 743 |

**Further optimization.** We can implement EXEX-NI even smaller at the cost of speed by choosing a TBC with a smaller tweakey. If we instantiate EXEX-NI with SKINNY-128-256 instead of SKINNY-128-384, we can eliminate the $\mathsf{TK}_3$ array and save roughly 1,000 GE, and the total circuit will be approximately 3,700 GE. Meanwhile, this modification comes at the cost of speed: the data rate is roughly halved because we can process only a 128-bit message block with a pair of TBC calls.

EXEX-NI and EXEX-I enjoy the conventional Romulus-H optimizations proposed by the Romulus team [14, 13]. The first optimization is to reduce the round number considering SKINNY's large security margin [13], i.e., SKINNY-128-384+. It will speed up each TBC call but its impact to area should be limited. Another optimization is to virtually reduce the tweakey size by using message blocks stuffed with zeros. For example, if we limit the message size to 128 bits in our tweakey configuration, the input to $\mathsf{TK}_3$ becomes always zero, and we can replace the $\mathsf{TK}_3$ array with a constant-value generator, which makes the circuit area similar to those instantiated with SKINNY-128-256.

## 8 Conclusion

In this paper, we proposed two DBL hash modes achieving minimum memory size. When an underlying BC supports an $n$-bit block and a $k$-bit key, our modes only require $n+k$-bit memory, which improves the previous smallest results of $2n+k$-bit memory. EXEX-NI mode is indifferentiable up to $n-\log n$ bits. Its instantiation with SKINNY-128 can be an efficient alternative to Romulus-H; our mode satisfies all the requirements in NIST LWC and provides hashing to Romulus with zero memory overhead, which significantly reduces the memory size of $3n+k$ bits for Romulus-H. EXEX-I mode focuses on the fact that indifferetiability may be unnecessary to be integrated with AEAD schemes, and we rigorously optimized its efficiency by focusing on the collision resistance.

There are several possible research directions, which includes relaxing the key size limitation $k \geq 2n$ of EXEX-NI, finding an attack rigorously matching the bound, a new mode without security loss of $\log n$ bits, integrated implementations of AEAD and hashing schemes. Application of our modes to the BC-based NIST finalist GIFT-COFB [2] is also interesting because it does not support hashing. Its underlying cipher GIFT128 supports $n = 128$ and $k = 128$, thus the block size fits perfectly, while the key size does not. Modifying GIFT128 to support a 256-bit key or 128-bit tweak is an interesting challenge.

## References

[1] Armknecht, F., Fleischmann, E., Krause, M., Lee, J., Stam, M., Steinberger, J.P.: The Preimage Security of Double-Block-Length Compression Functions. In: ASIACRYPT 2011. LNCS, vol. 7073, pp. 233–251. Springer (2011)

[2] Banik, S., Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT-COFB v1.0. Submitted to NIST LWC (2019)

[3] 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: CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer (2016)

[4] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the Indifferentiability of the Sponge Construction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer (2008)

[5] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2011)

[6] Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash Functions and RFID Tags: Mind the Gap. In: CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer (2008)

[7] Brachtl, B.O., Coppersmith, D., Hyden, M.M., Matyas Jr, S.M., Meyer, C.H.W., Oseas, J., Pilpel, S., Schilling, M.: Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function. U. S. Patent #4,908,861 (March 1990)

[8] Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer (2005)

[9] Damgård, I.: A Design Principle for Hash Functions. In: CRYPTO '89. LNCS, vol. 435, pp. 416–427. Springer (1989)

[10] Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer (2006)

[11] Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer (2007)

[12] 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)

[13] Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: New Results on Romulus. In: NIST Lightweight Cryptography Workshop 2020 (2020), https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/new-results-romulus-lwc2020.pdf

[14] Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus for Round 3 (2020), https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Romulus-for-Round-3.pdf

[15] Khurana, D.: MACs and Collision-Resistance (Section 6.3 Hash-then-MAC). https://courses.physics.illinois.edu/cs498ac3/fa2020/Files/Lecture_6_Scribe.pdf (2020)

[16] Lai, X., Massey, J.L.: Hash Function Based on Block Ciphers. In: EUROCRYPT '92. LNCS, vol. 658, pp. 55–70. Springer (1992)

[17] Lee, J., Stam, M.: MJH: A Faster Alternative to MDC-2. In: CT-RSA 2011. LNCS, vol. 6558, pp. 213–236. Springer (2011)

[18] Lee, J., Stam, M., Steinberger, J.P.: The Collision Security of Tandem-DM in the Ideal Cipher Model. In: CRYPTO 2011. LNCS, vol. 6841, pp. 561–577. Springer (2011)

[19] Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer (2004)

[20] Mennink, B.: Optimal Collision Security in Double Block Length Hashing with Single Length Key. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 526–543. Springer (2012)

[21] Merkle, R.C.: One Way Hash Functions and DES. In: CRYPTO '89. LNCS, vol. 435, pp. 428–446. Springer (1989)

[22] Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer (2011)

[23] Naito, Y.: Indifferentiability of Double-Block-Length Hash Function Without Feed-Forward Operations. In: ACISP 2017. LNCS, vol. 10343, pp. 38–57. Springer (2017)

[24] Naito, Y.: Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. In: LATINCRYPT 2019. LNCS, vol. 11774, pp. 65–85. Springer (2019)

[25] Naito, Y., Sasaki, Y., Sugawara, T.: Lightweight Authenticated Encryption Mode Suitable for Threshold Implementation. In: EUROCRYPT 2020. LNCS, vol. 12106, pp. 705–735 (2020)

[26] Naito, Y., Sasaki, Y., Sugawara, T.: LM-DAE: Low-Memory Deterministic Authenticated Encryption for 128-bit Security. IACR Trans. Symmetric Cryptol. **2020**(4), 1–38 (2020)

[27] NanGate: NanGate FreePDK45 Open Cell Library. https://si2.org/open-cell-library/ (2021), accessed: 2021-05-22

[28] NIST: The Keyed-Hash Message Authentication Code (HMAC). FIPS PUB 198-1 (2008)

[29] NIST: Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. https://csrc.nist.gov/Projects/lightweight-cryptography (2018)

[30] NIST: Lightweight Cryptography Standardization: Finalists Announced. https://csrc.nist.gov/News/2021/lightweight-crypto-finalists-announced (2021)

[31] NIST: Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process. https://csrc.nist.gov/publications/detail/nistir/8369/final (2021)

[32] Özen, O., Stam, M.: Another Glance at Double-Length Hashing. In: IMACC 2009. LNCS, vol. 5921, pp. 176–201. Springer (2009)