Blockcipher-Based Double-Length Hash Functions for Pseudorandom Oracles

Yusuke Naito

Mitsubishi Electric Corporation

Abstract. PRO (Pseudorandom Oracle) is an important security of hash functions because it ensures that the hash function inherits all properties of a random oracle up to the PRO bound (e.g., security against length extension attack, collision resistant security, preimage resistant security and so on). In this paper, we propose new blockcipher-based double-length hash functions, which are PROs up to $\mathcal{O}(2^n)$ query complexity in the ideal cipher model. Our hash functions use a single blockcipher, which encrypts an n-bit string using a $2n$-bit key, and maps an input of arbitrary length to an n-bit output. Since many blockciphers supports a $2n$-bit key (e.g. AES supports a 256-bit key), the assumption to use the $2n$-bit key length blockcipher is acceptable. To our knowledge, this is the first time double-length hash functions based on a single (practical size) blockcipher with birthday PRO security.

Keywords: Double-length hash function, pseudorandom oracle, ideal cipher model.

1 Introduction

The blockcipher-based design (e.g. [21, 28]) is the most popular method for constructing a cryptographic hash function. A hash function is designed by the following two steps: (1) designing a blockcipher and (2) designing a mode of operation. MD-family [29, 30], SHA-family [25] and SHA-3 candidates follow the design method. Another design method is to utilize a practical blockcipher such as AES. Such hash functions are useful in size restricted devices such as RFID tags and smart cards: when implementing both a hash function and a blockcipher, one has only to implement a blockcipher. However, the output length of practical blockciphers is far too short for a collision resistant hash function, e.g., 128 bits for AES. Thus the design of collision resistant double length hash functions (CR-DLHFs) is an interesting topic. The core of the design of CR-DLHFs is to design a collision resistant double-length compression functions (CR-DLCFs) which maps an input of fixed length (more than $2n$-bits) to an output of $2n$-bit length when using an n-bit output length blockcipher. Then a hash function combined a domain extension (e.g. strengthened Merkle-Damgård (SMD) [6, 23]), which preserves CR security, with a CR-DLCF yield a CR-DLHF. Many DLCFs, e,g,. [2, 24, 13, 16, 26, 18, 20], have been designed and the security is proven in the ideal cipher (IC) model [9, 13, 19, 11, 26, 17, 31].

The indifferentiability framework was introduced by Maurer et al. [22], which considers the reducibility of one system to another system. Roughly speaking, if a system F is indifferentiable from another system G up to q query complexity, we can use F instead of G up to q query complexity. So any cryptosystem is at least as secure under F as under G up to q query complexity. Recent proposed hash functions, e.g. SHA-3 candidates, considered the security of the indifferentiability from a random oracle (RO) (or Pseudorandom Oracle (PRO)). It ensures that the hash function has no structural design flows in composition and has security against any generic attacks up to the PRO query complexity (e.g., length extension attack, collision attack, preimage attack and so on). So it is important to consider PRO security when a DLHF is designed.

Hereafter a blockcipher which encrypts an n-bit string using a k-bit key is denoted by (k,n)-BC. Gong et al. [12] proved that the prefix-free Merkle-Damgård using the PBGV compression function [27] is PRO up to $\mathcal{O}(2^{n/2})$ query complexity as long as the ($2n$,n)-BC is IC. The PRO security is not enough because the query complexity is $\mathcal{O}(2^{64})$ when $n = 128$. Chang et al. [4] and Hirose et al. [14] proposed $2n$-bit output length DLHFs using a compression function $h : \{0,1\}^d \to \{0,1\}^n$ where $d > 2n$. Their proposals are PROs up to $\mathcal{O}(2^n)$ query complexity as long as h is a fixed input length RO (FILRO). Since IC where the plain text element is fixed by a constant is FILRO, these hash functions can be modified to blockcipher-based schemes which use a (d,n)-BC. However, practical blockciphers (such as AES) don't support d-bit key where $d > 2n$. Many other practical size[1] blockcipher-based DLHFs were proposed, e.g., [2, 24, 13, 16, 26, 18, 20], while none

[1] "Practical size" is the size supported by practical blockciphers e.g. AES.

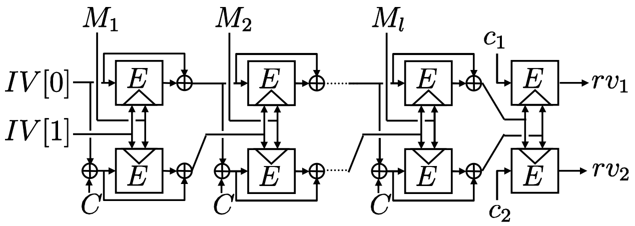

Fig. 1. Our DLHF using Hirose's compression function

of them achieves PRO security.[2] There is no hash function with birthday PRO security, and thus, we rise the following question:

Can we construct a DLHF from a "single practical size blockcipher" with "birthday PRO security"?

In this paper, we propose DLHFs using a single ($2n$,n)-BC, which are PROs up to $\mathcal{O}(2^n)$ query complexity in the IC model. Since many blockciphers support $2n$-bit key length, e.g., AES supports 256-bit key length, and the existing DLCFs (e.g. Hirose's compression function [13], Tandem-DM [16], Abreast-DM [16], and generalized DLCF [26]) use a ($2n$,n)-BC, the assumption to use a ($2n$,n)-BC is acceptable. To our knowledge, our hash functions are the first time DLHFs based on a practical size blockcipher with birthday PRO security. When $n = 128$ which is supported by AES, our hash functions have $\mathcal{O}(2^{128})$ security. Since our hash functions use only a *single* blockcipher, it is useful on size restricted devices when implementing both a hash function and a blockcipher. (the hybrid encryption schemes use both a blockcipher and a hash function (used in a key derivation function), for example.)

Our DLHF. Our DLHFs, which use Hirose's compression function [13], Tandem-DM [16] or Abreast-DM [16], iterate the compression function and use a new post-processing function f at the last iteration which calls a $(2n, n)$-BC twice. Our DLHFs are slightly lesser for speed than existing CR-DLHFs but have higher security (birthday PRO security).

Let $\mathsf{BC}_{2n,n} = (E, D)$ be a ($2n$,n)-BC where E is an encryption function and D is a decryption function. Let $\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}$ be a DLCF: Hirose's compression function, Tandem-DM, or Abreast-DM. Let $\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}} : \{0,1\}^* \rightarrow \{0,1\}^{2n}$ be the SMD hash function using the compression function $\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}$. Our DLHF is defined as follows:

$$F^{\mathsf{BC}_{2n,n}}(M) = f^{\mathsf{BC}_{2n,n}}(\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}}(M))$$

where $f^{\mathsf{BC}_{2n,n}}(x) = E(x, c_1)||E(x, c_2)$ and c_1 and c_2 are n-bit constant values. Note that the first element of the encryption function is the key element and the second element is the plain text element. The DLHF using Hirose's compression function is illustrated in Fig. 1 where each line is n bits and $IV[0], IV[1], C, c_1$ and c_2 are constant values. Note that in this figure we omit the suffix free padding function sfpad. So the hash function takes as its input a message M, $\mathsf{sfpad}(M) = M_1||M_2|| \cdots ||M_l$ with each block of n bits, and outputs the final value $rv_1||rv_2$. We use the DLHF $\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}}$ to compress an arbitrary length input into an fixed input length value. Since SMD hash functions cannot be used as ROs [5], the post-processing function $f^{\mathsf{BC}_{2n,n}}$ is used to guarantee PRO security.

The use of the constant values c_1 and c_2 in the post-processing function is inspired by the design technique of EMD proposed by Bellare and Ristenpart [1]. This realizes the fact that the post-processing function behaves like RO. So we can treat our hash function as a NMAC-like hash function. Note that the security of EMD is proven when the compression function is FILRO, while the security of our hash functions is proven when the compression function is the DLCF in the IC model. So additional analyses are needed due to the invertible property of IC and the structures of DLCFs.[3] We thus prove the PRO security of $F^{\mathsf{BC}_{2n,n}}$ by using

[2] Since PRO security is stronger security than CR security. CR security does not guarantee PRO security.

[3] One may think that there is an attack based on a decryption (inversion) function of the blockcipher. But our hash functions avoid the attack from the PRO security proof. For confirmation, we consider the attack in Appendix A.

three techniques: the PrA (Preimage Aware) design framework of Dodis et al. [7], PRO for a small function [5], and *indifferentiability from a hash function*. The first two techniques are existing techniques and the last technique is a new application of the indifferentiability framework [22].

First, we prove that the DLCFs are PrA up to $\mathcal{O}(2^n)$ query complexity. The PrA design framework offers the hash functions which are PROs up to $\mathcal{O}(2^n)$ query complexity where FILRO is used as the post-processing function. Second, we convert FILRO into the blockcipher-based post-processing function. We prove that the post-processing function is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model (PRO for a small function). Then, we prove that the PRO security of the post-processing function and the first PRO result ensure that the converted hash functions are PROs up to $\mathcal{O}(2^n)$ query complexity. Note that the hash functions use two blockciphers.[4] Finally, we consider the single-blockcipher-based hash functions $F^{\mathsf{BC}_{2n,n}}$. We prove that the single blockcipher-based hash functions are indifferentiable from the two-blockciphers-based hash functions in the IC model up to $\mathcal{O}(2^n)$ query complexity (indifferentiability from a hash function). Then we show that the indifferentiable security result and the second PRO result ensure that our hash functions are PROs up to $\mathcal{O}(2^n)$ query complexity in the IC model.

2 Preliminaris

Notation. For two values x, y, $x||y$ is the concatenated value of x and y. For some value y, $x \leftarrow y$ means assigning y to x. $\oplus$ is bitwise exclusive or. $|x|$ is the bit length of x. For a set (list) $\mathcal{T}$ and an element W, $\mathcal{T} \leftarrow W$ means to insert W into $\mathcal{T}$ and $\mathcal{T} \xleftarrow{\cup} W$ means $\mathcal{T} \leftarrow \mathcal{T} \cup \{W\}$. For some $2n$-bit value x, $x[0]$ is the first n bit value and $x[1]$ is the last n-bit value. $\mathsf{BC}_{d,n} = (E, D)$ be a blockcipher where $E : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^n$ is an encryption function, $D : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a decryption function, the key size is d bits and the cipher text size is n bits. $\mathcal{C}_{d,n} = (E_I, D_I)$ be a ideal cipher (IC) where $E_I : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^n$ is an encryption oracle, $D_I : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a decryption oracle, the key size is d bits and the cipher text size is n bits. $\mathcal{F}_{a,b} : \{0,1\}^a \rightarrow \{0,1\}^b$ is a random oracle (RO). An arbitrary input length random oracle is denoted by $\mathcal{F}_b : \{0,1\}^* \rightarrow \{0,1\}^b$. For any algorithm A, we write $\mathsf{Time}(A)$ to mean the sum of its description length and the worst-case number of steps.

Merkle-Damgård [6, 23]. Let $h : \{0,1\}^{2n} \times \{0,1\}^d \rightarrow \{0,1\}^{2n}$ be a compression function using a primitive P (more strictly h^P) and $\mathsf{pad} : \{0,1\}^* \rightarrow (\{0,1\}^d)^*$ be a padding function. The Merkle-Damgård hash function MD^h is described as follows where IV is a $2n$-bit initial value.

```
$\underline{\mathrm{MD}^h(M)}$
$z \leftarrow IV$;
Break $\mathsf{pad}(M)$ into $d$-bit blocks, $\mathsf{pad}(N) = M_1||\cdots||M_l$;
for $i = 1, \ldots, l$ do $z \leftarrow h(z, M_i)$;
Ret $z$;
```

We denote MD^h, when padding pad is a suffix-free padding sfpad, by SMD^h, called strengthened Merkle-Damgård. We assume that it is easy to strip padding, namely that there exists an efficiently computable function $\mathsf{unpad} : (\{0,1\}^d)^* \rightarrow \{0,1\}^* \cup \{\perp\}$ such that $x = \mathsf{unpad}(\mathsf{pad}(x))$ for all $x \in \{0,1\}^*$. Inputs to unpad that are not valid outputs of pad are mapped to $\perp$ by unpad.

Pseudorandom Oracle [22]. Let $H^P : \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function that utilizes an ideal primitive P. We say that H^P is PRO if there exists an efficient simulator S that simulates P such that for any distinguisher A outputting a bit it is the case that

$$\mathsf{Adv}^{\mathsf{pro}}_{H^P,S}(A) = |\Pr[A^{H^P,P} \Rightarrow 1] - \Pr[A^{\mathcal{F}_n,S^{\mathcal{F}_n}} \Rightarrow 1]|$$

is small where the probabilities are taken over the coins used the experiments. S can make queries to $\mathcal{F}_n$. The S's task is to simulate P such that relations among responses of (H^P, P) hold in responses of $(\mathcal{F}_n, S)$ as well.

[4] Two independent ideal cipher can be obtained from a single ideal cipher by victimizing one bit of the key space. So using a blockcipher with the $2n+1$-bit key space and the n-bit key space, the hash functions which uses a single blockcipher can be realized. But the size of the blockcipher is not a practical size.

Preimage Awareness [7, 8]. The notion of preimage awareness is useful for PRO security proofs of NMAC hash functions. We only explain the definition of preimage awareness. Please see Section 3 of [8] for the spirit of the notion. Let F^P be a hash function using an ideal primitive P. The preimage awareness of F^P is estimated by the following experiment.

```
$\underline{\mathsf{Exp}^{\mathsf{pra}}_{F^P,P,\mathcal{E},A}}$
$x \xleftarrow{\$} A^{\mathsf{P},\mathsf{Ex}}$;
$z \leftarrow F^P(x)$;
Ret $(x \neq \mathsf{V}[z] \wedge \mathsf{Q}[z] = 1)$;
```

```
$\underline{\textbf{oracle } \mathsf{P}(m)}$
$c \leftarrow P(m)$;
$\alpha \xleftarrow{\cup} (m, c)$;
Ret $c$;
```

```
$\underline{\textbf{oracle } \mathsf{Ex}(z)}$
$\mathsf{Q}[z] \leftarrow 1$;
$\mathsf{V}[z] \leftarrow \mathcal{E}(z, \alpha)$;
Ret $\mathsf{V}[z]$;
```

Here an adversary A is provided two oracles P and Ex. The oracle P provides access to the ideal primitive P and records a query histry α. The extraction oracle Ex provides an interface to an extractor $\mathcal{E}$, which is a deterministic algorithm that uses z and the query history α of P, and returns either $\perp$ or an element x' such that $F^P(x') = z$. If x' can be constructed from α, it returns x' and otherwise returns $\perp$. In this experiment, the (initially everywhere $\perp$) array Q and the (initially empty) array V are used. When z is queried to Ex, $\mathsf{Q}[z] \leftarrow 1$ and then the output of $\mathcal{E}(z, \alpha)$ is assigned to $\mathsf{V}[z]$. For the hash function F^P, the adversary A, and the extractor $\mathcal{E}$, we define the advantage relation

$$\mathsf{Adv}^{\mathsf{pra}}_{F^P,P,\mathcal{E}} = \Pr[\mathsf{Exp}^{\mathsf{pra}}_{F^P,P,\mathcal{E},A} \Rightarrow \mathsf{true}]$$

where the probabilities are over the coins used in running the experiments. When there exists an efficient extractor $\mathcal{E}$ such that for any adversary A the above advantage is small, we say that F^P is preimage aware (PrA).

The pra-advantage can be evaluated from the cr-advantage (collision resistance advantage) and the 1-wpra (1-weak PrA) advantage [8]. The 1-WPrA experiment is described as follows.

```
$\underline{\mathsf{Exp}^{\mathsf{1wpra}}_{F^P,P,\mathcal{E}^+,A}}$
$x \xleftarrow{\$} A^{\mathsf{P},\mathsf{Ex}^+}$;
$z \leftarrow F^P(x)$;
Ret $(x \notin L \wedge \mathsf{Q}[z] = 1)$;
```

```
$\underline{\textbf{oracle } \mathsf{P}(m)}$
$c \leftarrow P(m)$;
$\alpha \xleftarrow{\cup} (m, c)$;
Ret $c$;
```

```
$\underline{\textbf{oracle } \mathsf{Ex}^+(z)}$
$\mathsf{Q}[z] \leftarrow 1$;
$L \leftarrow \mathcal{E}^+(z, \alpha)$;
Ret $L$;
```

The difference between the 1-WPrA experiment and the PrA experiment is the extraction oracle. In the 1-WPrA experiment, a multi-point extractor oracle Ex^+ is used. Ex^+ provides an interface to a multi-point extractor $\mathcal{E}^+$, which is a deterministic algorithm that uses z and α, and returns either $\perp$ or a set of an element in the domain of F^P. The output (set) of $\mathcal{E}^+$ is stored in list L. Thus, if $L \neq \{\perp\}$, for any $x' \in L$ $F^P(x') = z$. In this experiment, an adversary A can make only a single query to Ex^+. For a hash function F^P, an adversary A, and a multi-point extractor $\mathcal{E}^+$, we define the advantage relation

$$\mathsf{Adv}^{\mathsf{1wpra}}_{F^P,P,\mathcal{E}} = \Pr[\mathsf{Exp}^{\mathsf{1wpra}}_{F^P,P,\mathcal{E}^+,A} \Rightarrow \mathsf{true}]$$

where the probabilities are over the coins used in running the experiments. When there exists an efficient multi-point extractor $\mathcal{E}^+$ such that the above advantage is small for any adversary A, we say that F^P is 1-WPrA.

The definition of the cr-advantage as follows. Let A be an adversary that outputs a pair of values x and x'. To hash function F^P using primitive P and adversary A we associate the advantage relation

$$\mathsf{Adv}^{\mathsf{cr}}_{F^P,P}(A) = \Pr[(x, x') \xleftarrow{\$} A^P : F^P(x) = F^P(x') \wedge x \neq x']$$

where the probability is over the coins used by A and primitive P.

Then the pra-advantage can be evaluated as follows.

Lemma 1 (Lemmas 3.3 and 3.4 of [8]). *Let $\mathcal{E}^+$ be an arbitrary multi-point extractor. There exists an extractor $\mathcal{E}$ such that for any pra-advarsary A^{pra} making q_e extraction queries and q_P primitive queries there exists 1-wpra adversary A^{1wpra} and cr-adversary A^{cr} such that*

$$\mathsf{Adv}^{\mathsf{pra}}_{F^P,P,\mathcal{E}}(A^{\mathsf{pra}}) \leq q_e \cdot \mathsf{Adv}^{\mathsf{1wpra}}_{F^P,P,\mathcal{E}^+}(A^{\mathsf{1wpra}}) + \mathsf{Adv}^{\mathsf{cr}}_{F^P,P}(A^{\mathsf{cr}}).$$

A^{1wpra} runs in time at most $\mathcal{O}(q_e \mathsf{Time}(\mathcal{E}^+))$ and makes the same number of P queries as A^{pra}. A^{cr} asks q_P queries and run in time $\mathcal{O}(q_e \cdot \mathsf{Time}(\mathcal{E}^+))$. $\mathcal{E}$ runs in the same time as $\mathcal{E}^+$. ♦

NMAC Hash Function. Let $g : \{0,1\}^n \rightarrow \{0,1\}^n$ be a function and $H^P : \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function using primitive P such that g is not used in H^P. Dodis et al. [8] proved that the PRO security of the NMAC hash function $g \circ H^P$ can be reduced into the PrA security of H^P.

Lemma 2 (Theorem 4.1 of [8]). *Let P be an ideal primitive, g be a random oracle and $\mathcal{E}$ be any extractor for H^P. Then there exists a simulator $S = (S_P, S_g)$ such that for any PRO adversary A making at most q_F, q_P, q_g queries to its three oracles $(\mathcal{O}_F, \mathcal{O}_P, \mathcal{O}_g)$ where $(\mathcal{O}_F, \mathcal{O}_P, \mathcal{O}_g) = (g \circ H^P, P, g)$ or $(\mathcal{O}_F, \mathcal{O}_P, \mathcal{O}_g) = (\mathcal{F}_n, S_P, S_g)$, there exists a PrA adversary B such that*

$$\mathsf{Adv}^{\mathsf{pro}}_{g \circ H^P, S}(A) \leq \mathsf{Adv}^{\mathsf{pra}}_{H^P, P, \mathcal{E}}(B).$$

S runs in time $\mathcal{O}(q_P + q_g \cdot \mathsf{Time}(\mathcal{E}))$. Let l be the length of the longest query made by A to $\mathcal{O}_H$. B runs in time $\mathcal{O}(\mathsf{Time}(A) + q_F t_H + q_P + q_g)$, makes $q_P + q_H q_F$ queries, q_g extraction queries, and outputs a preimage of length at most l where for any input M to H^P the output of $H^P(M)$ can be calculated within at most t_H times and q_H queries to P. ♦

Dodis et al. proved that the SMD construction preserves the PrA security as follows. Therefore, the PRO security of the NMAC hash function using the SMD hash function can be reduced into the PrA security of the compression function.

Lemma 3 (Theorem 4.2 of [8]). *Let h^P be a compression function using an ideal primitive P. Let $\mathcal{E}_h$ be an arbitrary extractor for h^P. There exists an extractor $\mathcal{E}_H$ for SMD^{h^P} such that for any adversary A_H making at most q_P primitive queries and q_e extraction queries and outputting a message at most l blocks there exists an adversary A_h such that*

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathrm{SMD}^{h^P}, P, \mathcal{E}_H}(A_H) \leq \mathsf{Adv}^{\mathsf{pra}}_{h^P, P, \mathcal{E}_h}(A_h)$$

$\mathcal{E}_H$ runs in time at most $l(\mathsf{Time}(\mathcal{E}_h) + \mathsf{Time}(\mathsf{unpad}))$. A_h runs in time at most $\mathcal{O}(\mathsf{Time}(A_H) + q_e l)$, makes at most $q_H + q_P$ ideal primitive queries, and makes at most $q_e l$ extraction queries where q_H is the maximum number of P queries to calculate SMD^{h^P}. ♦

3 Blockcipher-Based Double-Length Hash Functions for PROs

Let $\mathsf{BC}_{2n,n} = (E, D)$, $\mathsf{BC}^1_{2n,n} = (E1, D1)$, $\mathsf{BC}^2_{2n,n} = (E2, D2)$, and $\mathsf{BC}^3_{2n,n} = (E3, D3)$ be blockciphers. Let $g : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ be a function. In this section, we propose the following DLHFs using a single blockcipher and prove that our hash functions are PROs up to $\mathcal{O}(2^n)$ query complexity in the IC model.

$$F^{\mathsf{BC}_{2n,n}}(M) = f^{\mathsf{BC}_{2n,n}}(\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}_{2n,n}}}(M))$$

where $f^{\mathsf{BC}_{2n,n}}(x) = E(x, c_1)||E(x, c_2)$ such that c_1 and c_2 are n-bit different constant values and are different from values which are defined by the compression function (see subsection 3.3). The hash functions use Hirose's compression function, Tandem-DM, and Abreast-DM as the underlying DLCF, respectively. We prove the PRO security by the three steps. Each step uses the PrA design framework, PRO for a small function and indifferentiability from a hash function, respectively.

- **Step 1.** We prove that Hirose's compression function, Tandem-DM, and Abreast-DM are PrA up to $\mathcal{O}(2^n)$ query complexity in the IC model. Lemma 2 and Lemma 3 then ensure that the following NMAC hash function is PRO up to $\mathcal{O}(2^n)$ query complexity as long as the blockcipher is IC and g is FILRO.

$$F_1^{g, \mathsf{BC}^1_{2n,n}}(M) = g(\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}^1_{2n,n}}}(M))$$

- **Step 2.** We prove that $f^{\mathsf{BC}^3_{2n,n}}$ is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model where c_1 and c_2 are n-bit different values. Then, we prove that the PRO security of F_1 and the PRO security of f ensure that the following hash function is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model, namely, it can be used as RO up to $\mathcal{O}(2^n)$ query complexity in the IC model.

$$F_2^{\mathsf{BC}^2_{2n,n}, \mathsf{BC}^3_{2n,n}}(M) = f^{\mathsf{BC}^3_{2n,n}}(\mathrm{SMD}^{\mathrm{DLCF}^{\mathsf{BC}^2_{2n,n}}}(M))$$

- **Step 3.** This is the final step. We use the indifferentiability from a hash function: we prove that $F^{\mathsf{BC}_{2n,n}}$ is indifferentiable from $F_2^{\mathsf{BC}^2_{2n,n},\mathsf{BC}^3_{2n,n}}$ up to $\mathcal{O}(2^n)$ query complexity in the IC model. Then, we prove that the indifferentiable result and the PRO security of F_2 ensure that $F^{\mathsf{BC}_{2n,n}}$ is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model.[5]

3.1 Step 1

We prove that Hirose's compression function [13] is PrA up to $\mathcal{O}(2^n)$ query complexity as long as the blockcipher is an ideal cipher. Similarly, we can prove that Abreast-DM and Tandem-DM [16] are PrA. These proofs are given in Appendix B and Appendix C, respectively.

Definition 1 (Hirose's Compression Function). *Let* $\mathsf{BC}^1_{2n,n} = (E1, D1)$ *be a blockcipher. Let* $\mathsf{CF}^{\mathsf{Hirose}}[\mathsf{BC}^1_{2n,n}] : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ *be a compression function such that* $(G_i, H_i) = \mathsf{CF}^{\mathsf{Hirose}}[\mathsf{BC}^1_{2n,n}](G_{i-1}||H_{i-1}, M_i)$ *where* $G_i, H_i, G_{i-1}, H_{i-1} \in \{0,1\}^n$ *and* $M_i \in \{0,1\}^n$. (G_i, H_i) *is calculated as follows:*

$$G_i = G_{i-1} \oplus E1(H_{i-1}||M_i, G_{i-1}) \tag{1}$$

$$H_i = C \oplus G_{i-1} \oplus E1(H_{i-1}||M_i, G_{i-1} \oplus C,) \tag{2}$$

We call the procedure 1 "first block" and the procedure 2 "second block". ♦

Lemma 4 (Hirose's Compression Function is PrA). *Let* $\mathcal{C}^1_{2n,n} = (E1_I, D1_I)$ *be an ideal cipher. There exists an extractor* $\mathcal{E}$ *such that for any adversary* A *making at most* q_P *queries to* $\mathcal{C}_{2n,n}$ *and* q_e *extraction queries we have*

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{CF}^{\mathsf{Hirose}}[\mathcal{C}^1_{2n,n}],\mathcal{C}^1_{2n,n},\mathcal{E}}(A) \le \frac{2q_P^2}{(2^n - 2q_P)^2} + \frac{2q_P}{2^n - 2q_P} + \frac{2q_P q_e}{(2^n - q_P)^2}$$

where $\mathcal{E}$ *runs in time at most* $\mathcal{O}(q_e q_P)$. ♦

Proof. We prove that Hirose's compression function is 1-WPrA, and then Lemma 1 gives the final bound. We note that Theorem 3 of [11] upperbounds the cr-advantage of A by $2q_P^2/(2^n - 2q_P)^2 + 2q_P/(2^n - 2q_P)$, yielding the first two terms.

Intuitively, the 1-WPrA game for the compression function is that A declares a value z then an extractor outputs preimages, stored in L, of z which can be constructed from input-output values of A's queries to $\mathcal{C}^1_{2n,n}$. Then A outputs a new preimage of z which is not stored in L. Note that A can adaptively query to $\mathcal{C}^1_{2n,n}$. We define the multi-point extractor to utilize the preimage resistant bound, proven in [11], of Hirose's compression function in Fig. 2. If an input-output triple of the first block is defined, automatically the input of the second block is defined, and vice versa, from the definition of the compression function. For a query (z, α) to $\mathcal{E}^+$, when there is an input-output triple (k, x, y) such that $x \oplus y = z[0]$, the multi-point extractor $\mathcal{E}^+$ checks whether the output of the second block is equal to $z[1]$ or not and if this holds the multi-point extractor stores it in the return list L, and vice versa. Therefore, A must find a new preimage of z to win the 1-WPrA experiment. Thus one can straightforwardly adapt the preimage resistant advantage of the compression function (described in Theorem 5 of [11])[6] because the proof of Theorem 5 of [11] can be applied to the case that an adversary selects an image z of the compression function and then finds the preimage of z. The advantage is at most $2q_P/(2^n - q_P)^2$. □

Lemma 4 ensures the following theorem via Lemma 2 and Lemma 3 where F_1 is PRO up to $\mathcal{O}(2^n)$ query complexity.

[5] One may think that since indifferentiability from a hash function considers indifferentiability of two hash functions based on small primitives, the universal composability (UC) [3] may be extended to this setting. However, this is not true because indifferentiability from a hash function follows the indifferentiability framework [22] and the definition of the indifferentiability framework is slightly different from that of the UC framework: indifferentiability considers "reducibility" of one system to another and UC considers "composability" of several systems.

[6] Note that while the 1-WPrA bound is equal to the preimage bound, this is not trivial because one needs to construct the extractor that converts the preimage bound into the 1-WPrA bound.

```
algorithm $\mathcal{E}^+(z, \alpha)$
Let $L$ be an empty list;
Parse $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i) \leftarrow \alpha$; $//E1(k_j, x_j) = y_j$
For $j = 1$ to $i$ do
    If $z[0] = x_j \oplus y_j$ then
        $y \leftarrow E1_I(k_j, x_j \oplus C)$;
        If $z[1] = C \oplus x_j \oplus y$ then $L \xleftarrow{\cup} (x_j || k[0], k[1])$;
    If $z[1] = x_j \oplus y_j$ then
        $y \leftarrow E1_I(k_j, x_j \oplus C)$;
        If $z[0] = C \oplus x_j \oplus y$ then $L \xleftarrow{\cup} ((x_j \oplus C) || k[0], k[1])$;
If $L$ is not an empty list then return $L$ otherwise return $\perp$;
```

Fig. 2. Multi-Point Extractor

Theorem 1. *There exists a simulator $S_1 = (S1_g, S1_{\mathcal{C}})$ where $S1_{\mathcal{C}} = (S1_E, S1_D)$ such that for any distinguisher A_1 making at most (q_H, q_g, q_E, q_D) queries to four oracles which are $(F_1, g, E1, D1)$ or $(\mathcal{F}_{2n}, S1_g, S1_E, S1_D)$, we have*

$$\mathsf{Adv}^{\mathsf{pro}}_{F_1^{g,\mathcal{C}^1_{2n,n}}, S1}(A_1) \leq \frac{2Q_1^2}{(2^n - 2Q_1)^2} + \frac{2Q_1}{2^n - 2Q_1} + \frac{2lq_gQ_1}{(2^n - Q_1)^2}$$

where S_1 works in time $\mathcal{O}(q_E + q_D + lq_gQ_1) + lq_g \times \mathsf{Time}(\mathsf{unpad})$ and $S1_g$ makes q_g queries to $\mathcal{F}_{2n}$ where $Q_1 = 2l(q_H + 1) + q_E + q_D$. $S1_g$ simulates g, which makes one query to $\mathcal{F}_{2n}$ for one $S1_g$ query, and $S1_{\mathcal{C}}$, which makes no query, simulates the ideal cipher. ⧫

3.2 Step 2

Lemma 5 ($f^{\mathcal{C}^3_{2n,n}}$ is PRO). *Let $\mathcal{C}^3_{2n,n} = (E3_I, D3_I)$ be an ideal cipher. Let $g = \mathcal{F}_{2n,2n}$. There exists a simulator $\mathsf{S} = (\mathsf{S}_E, \mathsf{S}_D)$ such that for any distinguisher A_2 making at most q_f, q_E and q_D queries to oracles $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D)$ where $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (f^{\mathcal{C}^3_{2n,n}}, E3_I, D3_I)$ or $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (g, \mathsf{S}_E, \mathsf{S}_D)$, we have*

$$\mathsf{Adv}^{\mathsf{pro}}_{f^{\mathcal{C}^3_{2n,n}}, \mathsf{S}}(A_2) \leq \frac{q_f + q_E + q_D}{2^n}$$

where S works in time $\mathcal{O}(\mathsf{Time}(A_2) + q_E + q_D)$ and makes at most queries $q_E + q_D$. S simulates the ideal cipher. ⧫

Proof. We define $\mathsf{S} = (\mathsf{S}_E, \mathsf{S}_D)$ such that it simulates $\mathcal{C}^3_{2n,n} = (E3_I, D3_I)$ and the relation among responses of $(f^{\mathcal{C}^3_{2n,n}}, E3_I, D3_I)$ holds in responses of $(g, \mathsf{S}_E, \mathsf{S}_D)$ as well. Since the relation $f^{\mathcal{C}^3_{2n,n}}(k) = E3_I(k, c_1) || E3_I(k, c_2)$ holds, we define S so that $\mathsf{S}_E(k, c_1) || \mathsf{S}_E(k, c_2) = g(k)$. That is, $\mathsf{S}_E(k, c_1) = y[0]$, $\mathsf{S}_E(k, c_2) = y[1]$, $\mathsf{S}_D(k, y[0]) = c_1$, and $\mathsf{S}_D(k, y[1]) = c_2$ where $y = g(k)$.

```
simulator $\mathsf{S}_E(k, x)$
If $\mathsf{E}[k, x] \neq \perp$, ret $\mathsf{E}[k, x]$;
If $\mathsf{E}[k, c_1] = \perp$,
    $y^* \leftarrow g(k)$;
    $\mathsf{E}[k, c_1] \xleftarrow{\$} y^*[0]$;
    $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
    $\mathsf{E}[k, c_2] \xleftarrow{\$} y^*[1]$;
    $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
If $x \neq c_1$ and $x \neq c_2$,
    $\mathsf{E}[k, x] \xleftarrow{\$} \{0, 1\}^n \backslash \mathcal{T}_E[k]$;
    $\mathsf{D}[k, \mathsf{E}[k, x]] \leftarrow x$;
Ret $\mathsf{E}[k, x]$;
```

```
simulator $\mathsf{S}_D(k, y)$
If $\mathsf{D}[k, y] \neq \perp$, ret $\mathsf{D}[k, y]$;
If $\mathsf{E}[k, c_1] = \perp$,
    $y^* \leftarrow g(k)$;
    $\mathsf{E}[k, c_1] \xleftarrow{\$} y^*[0]$;
    $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
    $\mathsf{E}[k, c_2] \xleftarrow{\$} y^*[1]$;
    $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
If $\mathsf{D}[k, y] \neq \perp$,
    $\mathsf{D}[k, y] \xleftarrow{\$} \{0, 1\}^n \backslash \{\mathcal{T}_D[k]\}$;
    $\mathsf{E}[k, \mathsf{D}[k, y]] \leftarrow y$;
Ret $\mathsf{D}[k, y]$;
```

S has (initially everywhere $\perp$) arrays E, D and (initially empty) tables $\mathcal{T}_E, \mathcal{T}_D$. When $y = \mathsf{S}_E(k, x)$, y is stored in $\mathsf{E}[k, x]$ and x is stored in $\mathsf{D}[k, y]$. For any (k, x) such that $\mathsf{E}[k, x] \neq \perp$, $\mathsf{E}[k, x]$ is stored in $\mathcal{T}_E[k]$, and for any (k, y) such that $\mathsf{D}[k, y] \neq \perp$, $\mathsf{D}[k, y]$ is stored in $\mathcal{T}_D[k]$.

We give the proof via a game-playing argument on the game sequences Game 0, Game 1, Game 2. Game 0 is the $f^{\mathcal{C}^3_{2n,n}}$ scenario and Game 2 is the g scenario. In each game, A_2 can make queries to three oracles $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D)$. Let Gj be the event that in Game j the distinguisher A_2 outputs 1. Therefore, $\Pr[A^{f^{\mathcal{C}^3_{2n,n}}, E3_I, D3_I} \Rightarrow 1] = \Pr[G0]$ and $\Pr[A^{g, S_E, S_D} \Rightarrow 1] = \Pr[G2]$. Thus

$$\mathsf{Adv}^{\mathsf{pro}}_{f^{\mathcal{C}^3_{2n,n}}, S}(A_2) = |\Pr[G0] - \Pr[G2]| \leq |\Pr[G1] - \Pr[G0]| + |\Pr[G2] - \Pr[G1]|$$

Game 0: Game 0 is the $f^{\mathcal{C}^3_{2n,n}}$ scenario. So $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (f^{\mathcal{C}^3_{2n,n}}, E3_I, D3_I)$.

Game 1: We change the underlying ideal cipher from $\mathcal{C}^3_{2n,n}$ to S. So $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (f^{\mathsf{S}}, \mathsf{S}_E, \mathsf{S}_D)$ where $f^{\mathsf{S}}(k) = \mathsf{S}_E(k, c_1) || \mathsf{S}_E(k, c_2)$. Note that only S has oracle access to g.

We must show that the A_2's view has statistically close distribution in Game 0 and Game 1. Since the difference between the games is the underlying function, we show that the output of the functions is statistically close; this in turn shows that the A_2's view has statistically close distribution in Game 0 and Game 1.

In the following, we use the following lazily-sample method of the ideal cipher.

Encryption Oracle $E_I(k, x)$

```
01 If $\mathsf{E}[k, x] \neq \perp$, ret $\mathsf{E}[k, x]$;
02 If $\mathsf{E}[k, c_1] = \perp$,
03     $\mathsf{E}[k, c_1] \xleftarrow{\$} \{0,1\}^n$;
04     $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
05     $\mathsf{E}[k, c_2] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$;
06     $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
07 If $x \neq c_1$ and $x \neq c_2$,
08     $\mathsf{E}[k, x] \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{T}_E[k]$;
09     $\mathsf{D}[k, \mathsf{E}[k, x]] \leftarrow x$;
10 Ret $\mathsf{E}[k, x]$;
```

Decryption Oracle $D_I(k, y)$

```
11 If $\mathsf{D}[k, y] \neq \perp$, ret $\mathsf{D}[k, y]$;
12 If $\mathsf{E}[k, c_1] = \perp$,
13     $\mathsf{E}[k, c_1] \xleftarrow{\$} \{0,1\}^n$;
14     $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
15     $\mathsf{E}[k, c_2] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$;
16     $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
17 If $\mathsf{D}[k, y] \neq \perp$,
18     $\mathsf{D}[k, y] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathcal{T}_D[k]\}$;
19     $\mathsf{E}[k, \mathsf{D}[k, y]] \leftarrow y$;
20 Ret $\mathsf{D}[k, y]$;
```

E and D are (initially everywhere $\perp$) arrays and $\mathcal{T}_E$ and $\mathcal{T}_D$ (initially empty) tables. For any (k, x) such that $\mathsf{E}[k, x] \neq \perp$, $\mathsf{E}[k, x]$ is stored in $\mathcal{T}_E[k]$, and for any (k, y) such that $\mathsf{D}[k, y] \neq \perp$, $\mathsf{D}[k, y]$ is stored in $\mathcal{T}_D[k]$. On a query which the key element is k, first the output of $E_I(k, c_1)$ is determined (Steps 03-04 or Steps 13-14) and second the output of $E_I(k, c_2)$ is determined (Steps 05-06 or Steps 15-16). Then the outputs of $E_I(k, x)$ such that $x \neq c_1$ and $x \neq c_2$ are determined. Since no adversary (distinguisher) learns $E_I(k, c_1)$ and $E_I(k, c_2)$ until querying the corresponding value, the procedures of Steps 03-06 and 13-16 do not affect the lazily-sample ideal cipher simulation.

We consider the difference of the ideal cipher and S. On a query in which the key element is k, the output of $\mathsf{S}_E(c_2, k)$ is randomly chosen from $\{0,1\}^n$ because g is RO while the output of $E_I(c_2, k)$ is randomly chosen from $\{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$. Thus the statistical distance between the uniform distribution in $\{0,1\}^n$ and the uniform distribution in $\{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$ is $1/2^n$. Since the number of queries to S is at most $q_f + q_E + q_D$ times,

$$|\Pr[G1] - \Pr[G0]| \leq \frac{q_f + q_E + q_D}{2^n}.$$

Game 2: We change $\mathcal{O}_f$ from $f^{\mathsf{S}}(k)$ to g. So $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (g, \mathsf{S}_E, \mathsf{S}_D)$ and this is the g scenario.

We show that the A_2's view in Game G1 and Game G2 is equivalent. $\mathcal{O}_f$ is different in both games. To prove the equivalence, we use the proof method in [5, 15]. Specifically, we prove the following two points. If those hold, then we can conclude that $\Pr[G1] = \Pr[G2]$.

1. In Game 1, for any query k $\mathcal{O}_f(k)$ is defined by $g(k)$. If this holds, the output distribution of $\mathcal{O}_f$ of Game 1 and Game 2 is the same.
2. In Game 2, $\mathcal{O}_E$ and $\mathcal{O}_D$ are consistent with $\mathcal{O}_f$ as well as in Game 1. $\mathcal{O}_f$ uses $\mathcal{O}_E$ in Game 1, while does not in Game 2 (note that in both games $(\mathcal{O}_E, \mathcal{O}_D) = (\mathsf{S}_E, \mathsf{S}_D)$). Thus, if this holds, the difference does not affect the output distribution of $\mathcal{O}_E$ and $\mathcal{O}_D$. Namely, the output distribution of $\mathcal{O}_E$ and $\mathcal{O}_D$ is the same.

Proof of Point 1. In Game 1, for any query k $\mathcal{O}_f(k) = \mathsf{S}_E(k, c_1)||\mathsf{S}_E(k, c_2)$. Since $\mathsf{S}_E(k, c_1) = y^*[0]$ and $\mathsf{S}_E(k, c_2) = y^*[1]$ where $y^* = g(k)$, $\mathcal{O}_f(k) = \mathsf{S}_E(k, c_1)||\mathsf{S}_E(k, c_2) = g(k)$.

Proof of Point 2. In Game 1, $\mathcal{O}_f$ uses $\mathcal{O}_E$, namely, if the outputs of $\mathsf{S}_E(k, c_1)$ and $\mathsf{S}_E(k, c_2)$ are defined, $\mathcal{O}_f(k) = \mathsf{S}_E(k, c_1)||\mathsf{S}_E(k, c_2)$. So we must show that the same holds in Game 2. Since $\mathsf{S}_E(k, c_1) = y^*[0]$ and $\mathsf{S}_E(k, c_2) = y^*[1]$ where $y^* = g(k)$, if the outputs of $\mathsf{S}_E(k, c_1)$ and $\mathsf{S}_E(k, c_2)$ are defined, $g(k) = \mathsf{S}_E(k, c_1)||\mathsf{S}_E(k, c_2)$. Namely, in Game 2, $\mathcal{O}_E$ and $\mathcal{O}_D$ consistent with $\mathcal{O}_f$ as in Game 1.

We thus have that $\Pr[G1] = \Pr[G2]$.

From above discussions, we have that

$$\mathsf{Adv}^{\mathsf{pro}}_{f^{\mathcal{C}^3_{2n,n}}, S}(A_2) \leq \frac{q_f + q_E + q_D}{2^n}.$$

□

Using Theorem 1 and Lemma 5, we show that F_2 using Hirose's compression function is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model. Similarly, we can prove the hash functions using Tandem-DM and Abreast-DM.

Theorem 2 (F_2 is PRO). *There exists a simulator $S_2 = (S2, S3)$ where $S2 = (S2_E, S2_D)$ and $S3 = (S3_E, S3_D)$ such that for any distinguisher A_3 making at most $(q_H, q_{E2}, q_{D2}, q_{E3}, q_{D3})$ queries to five oracles which are $(F_2, E2, D2, E3, D3)$ or $(\mathcal{F}_{2n}, S2_E, S2_D, S3_E, S3_D)$, we have*

$$\mathsf{Adv}^{\mathsf{pro}}_{F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}, S_2}(A_3) \leq \frac{2Q_2^2}{(2^n - 2Q_2)^2} + \frac{2Q_2}{2^n - 2Q_2} + \frac{2lq_3Q_2}{(2^n - Q_2)^2} + \frac{q_H + q_3}{2^n}$$

where S_2 works in time $\mathcal{O}(q_2 + lq_3Q_2) + lq_3 \times \mathsf{Time}(\mathsf{unpad})$ and $S3$ makes q_3 queries to $\mathcal{F}_{2n}$ where $Q_2 = 2l(q_H + 1) + q_{E2} + q_{D2}$, $q_2 = q_{E2} + q_{D2}$ and $q_3 = q_{E3} + q_{D3}$. ♦

Proof. We use Theorem 1 and Lemma 5. We define the simulator $S_2 = (S2, S3)$ by $S2 = S1_{\mathcal{C}}$ and $S3 = \mathsf{S}^{S1_g}$ where $(S1_g, S1_{\mathcal{C}})$ are defined in Theorem 1 and S is defined in Lemma 5. Namely, $S2_E = S1_E$, $S2_D = S1_D$, $S3_E = \mathsf{S}_E^{S1_g}$ and $S3_D = \mathsf{S}_D^{S1_g}$ where on a query to $S3_E$, S_E accepts the query, calculates the output value by using $S1_g$ and returns it, and similarly $S3_D$ is defined. In the following, we evaluate the PRO bound of $F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}$. We assume that A_2 is a distinguisher such that the PRO bound of $f^{\mathcal{C}^3_{2n,n}}$ is maximum, and A_1

is a distinguisher such that the PRO bound of $F_1^{g,\mathcal{C}^2_{2n,n}}$ is maximum. For any distinguisher A_3,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{pro}}_{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},S_2}(A_3) =& |\Pr[A_3^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}} \Rightarrow 1] - \Pr[A_3^{\mathcal{F}_{2n},S2^{\mathcal{F}_{2n}},S3^{\mathcal{F}_{2n}}} \Rightarrow 1]| \\
=& |\Pr[A_3^{F_1^{f^{\mathcal{C}^3_{2n,n}},\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}} \Rightarrow 1] - \Pr[A_3^{\mathcal{F}_{2n},S2^{\mathcal{F}_{2n}},S3^{\mathcal{F}_{2n}}} \Rightarrow 1]| \\
\leq& |\Pr[A_3^{F_1^{f^{\mathcal{C}^3_{2n,n}},\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}} \Rightarrow 1] - \Pr[A_3^{F_1^{g,\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},\mathsf{S}^g} \Rightarrow 1]| + \\
& |\Pr[A_3^{F_1^{g,\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},\mathsf{S}^g} \Rightarrow 1]| - \Pr[A_3^{\mathcal{F}_{2n},S1_{\mathcal{C}}^{\mathcal{F}_{2n}},\mathsf{S}^{S1_g^{\mathcal{F}_{2n}}}} \Rightarrow 1]| \\
\leq& |\Pr[A_2^{f^{\mathcal{C}^3_{2n,n}},\mathcal{C}^3_{2n,n}} \Rightarrow 1] - \Pr[A_2^{g,\mathsf{S}^g} \Rightarrow 1]| + \\
& |\Pr[A_1^{F_1^{g,\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},g} \Rightarrow 1]| - \Pr[A_1^{\mathcal{F}_{2n},S1_{\mathcal{C}}^{\mathcal{F}_{2n}},S1_g^{\mathcal{F}_{2n}}} \Rightarrow 1]| \\
=& \mathsf{Adv}^{\mathsf{pro}}_{f^{\mathcal{C}^3_{2n,n}},\mathsf{S}}(A_2) + \mathsf{Adv}^{\mathsf{pro}}_{F_1^{g,\mathcal{C}^2_{2n,n}},S_1}(A_1).
\end{aligned}$$

The second equation holds because $F_1^{f^{\mathcal{C}^3_{2n,n}},\mathcal{C}^2_{2n,n}} = F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}$. From the third equation to the forth equation, we change $A_3^{F_1^{\mathcal{O}_a,\mathcal{C}^2_{2n,n}},\mathcal{C}^2_{2n,n},\mathcal{O}_b}$ to $A_1^{\mathcal{O}_a,\mathcal{O}_b}$ and $A_3^{\mathcal{O}_c,\mathcal{O}_d,\mathsf{S}^{\mathcal{O}_e}}$ to $A_2^{\mathcal{O}_c,\mathcal{O}_d,\mathcal{O}_e}$. The second last equation holds because A_1 and A_2 are distinguishers where the PRO bounds are maximum.

Thus, when using the bounds of Theorem 1 and Lemma 5, the PRO bound of F_2 can be obtained where A_2 can make at most (q_H, q_{E3}, q_{D3}) queries to its three oracles (see Lemma 5) and A_1 can make at most $(q_H, q_3, q_{E2}, q_{D2})$ queries to its four oracles (see Theorem 1). □

3.3 Step 3

In the following proof, we consider the hash function using Hirose's compression function. Using the same proof, we can prove the cases of Tandem-DM and Abreast-DM. So we omit these proofs. When using Hirose's compression function, we use the constant values c_1 and c_2 of the post-processing function f such that c_1 and c_2 are not equal to $C \oplus IV[0]$ and $IV[0]$ where IV is the initial value of $\text{SMD}^{\text{DLCF}^{\mathsf{BC}_{2n,n}}}$ and C is the constant value used in Hirose's compression function. If c_1 and c_2 which are equal to $C \oplus IV[0]$ or $IV[0]$ are used, we cannot prove the security of the hash function. In this case, we fail to construct a simulator. When using Tandem-DM, c_1 and c_2 are such that the values are not equal to $IV[0]$ and $IV[1]$. When using Abreast-DM, c_1 and c_2 are such that the values are not equal to $IV[0]$ and $IV[1]$.

First, we define the indifferentiability from a hash function as follows.

Definition 2. *Let $H_1^{P_1} : \{0,1\}^* \to \{0,1\}^{2n}$ and $H_2^{P_2} : \{0,1\}^* \to \{0,1\}^{2n}$ be hash functions using ideal primitives P_1 and P_2, respectively. $H_1^{P_1}$ is indifferentiable from $H_2^{P_2}$ if there exists a simulator $\mathcal{S}$ such that for any distinguisher A_4 outputting a bit it is the case that*

$$\mathsf{Adv}^{\mathsf{indif}}_{H_1^{P_1},H_2^{P_2},\mathcal{S}}(A_4) \leq |\Pr[A_4^{H_1^{P_1},P_1} \Rightarrow 1] - \Pr[A_4^{H_2^{P_2},\mathcal{S}^{P_2}} \Rightarrow 1]|$$

is small where the probabilities are taken over the coins used the experiments. ♦

The following lemma is that F is indifferentiable from F_2 up to $\mathcal{O}(2^n)$ query complexity in the IC model.

Lemma 6. *Let $\mathcal{C}_{2n,n} = (E_I, D_I)$ be an ideal cipher. Let $\mathcal{C}^2_{2n,n} = (E2_I, D2_I)$ and $\mathcal{C}^3_{2n,n} = (E3_I, D3_I)$ be different ideal ciphers. There exists a simulator $\mathcal{S} = (\mathcal{S}_E, \mathcal{S}_D)$ such that for any distinguisher A_4 making at most q_F, q_E and q_D queries to its oracles $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D)$ which are $(F^{\mathcal{C}_{2n,n}}, E_I, D_I)$ or $(F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}, \mathcal{S}_E, \mathcal{S}_D)$, we have*

$$\mathsf{Adv}^{\mathsf{indif}}_{F^{\mathcal{C}_{2n,n}},F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}}(A_4) \leq \frac{14 \times (2(lq_F + 1) + q_E + q_D)}{2^n - (2(lq_F + 1) + q_E + q_D)}$$

where $\mathcal{S}$ works in time $\mathcal{O}(\mathsf{Time}(A) + 3(q_E + q_D))$ and makes at most ideal cipher queries $q_E + q_D$. l is the maximum number of n-bit blocks of a query to $\mathcal{O}_F$. ♦

```
simulator $\mathcal{S}_E(k,x)$
E01 If $\mathsf{E}[k,x] \neq \perp$ then ret $\mathsf{E}[k,x]$;
E02 If $\mathsf{E}[k,c_1] = \perp$,
E03     $y \leftarrow E3_I(k,c_1)$;
E04     $\mathsf{E}[k,c_1] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_1$;
E05     $y \leftarrow E3_I(k,c_2)$;
E06     $\mathsf{E}[k,c_2] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_2$;
E07 If $x \neq c_1$ and $x \neq c_2$,
E08     $y \leftarrow E2_I(k,x)$;
E09     $\mathsf{E}[k,x] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow x$;
E10 Ret $\mathsf{E}[k,x]$;
```

```
simulator $\mathcal{S}_D(k,y)$
D01 If $\mathsf{D}[k,y] \neq \perp$ then ret $\mathsf{D}[k,y]$;
D02 If $\mathsf{E}[k,c_1] = \perp$,
D03     $y \leftarrow E3_I(k,c_1)$;
D04     $\mathsf{E}[k,c_1] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_1$;
D05     $y \leftarrow E3_I(k,c_2)$;
D06     $\mathsf{E}[k,c_2] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_2$;
D07 If $\mathsf{D}[k,y] = \perp$,
D08     $x \leftarrow D2_I(k,y)$;
D09     $\mathsf{E}[k,x] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow x$;
D10 Ret $x$;
```

Fig. 3. Simulator

Proof. Without loss of generality, we omit the padding function of our hash function which is more general case than including the padding function. In Fig. 3, we define a simulator $\mathcal{S} = (\mathcal{S}_E, \mathcal{S}_D)$ such that it simulates the ideal cipher $\mathcal{C}_{2n,n} = (E_I, D_I)$ and the relation among responses of $(F^{\mathcal{C}_{2n,n}}, E_I, D_I)$ holds in responses of $(F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}, \mathcal{S}_E, \mathcal{S}_D)$ as well, namely, $F^{\mathcal{S}}(M) = F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}(M)$. Since $E2_I$ is used in inner calculations and $E3_I$ is used in the post-processing calculations, if for a query (k,x) to $\mathcal{S}_E$ (k,x) is used in the post-processing calculations, it returns the output of $E3_I(k,x)$, and otherwise it returns the output of $E2_I(k,x)$. Since in post-processing calculation the second value x of a E query is c_1 or c_2, we define $\mathcal{S}$ such that $\mathcal{S}_E(k,x)$ is defined by $E3_I(k,x)$, if $x = c_1$ or $x = c_2$, and is defined by $E2_I(k,x)$ otherwise.[7] E and D are (initially everywhere $\perp$) arrays.

We give the proof via a game-playing argument on the game sequences Game 0, Game 1, and Game 2. Game 0 is the F scenario and Game 2 is the F_2 scenario. In each game, A_4 can make queries to three oracles $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D)$. Let Gj be the event that in Game j the distinguisher A_4 outputs 1. Therefore, $\Pr[A_4^{F^{\mathcal{C}_{2n,n}}, E_I, D_I} \Rightarrow 1] = \Pr[G0]$ and $\Pr[A_4^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}, \mathcal{S}_E, \mathcal{S}_D} \Rightarrow 1] = \Pr[G2]$. Thus

$$\mathsf{Adv}^{\mathsf{indif}}_{F^{\mathcal{C}_{2n,n}}, F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}, \mathcal{S}}(A_4) = |\Pr[G0] - \Pr[G1]| \leq |\Pr[G1] - \Pr[G0]| + |\Pr[G2] - \Pr[G1]|$$

Game 0: Game 0 is the F scenario. So $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D) = (F^{\mathcal{C}_{2n,n}}, E_I, D_I)$.

Game 1: We modify the underlying functions $(\mathcal{O}_E, \mathcal{O}_D)$ from (E_I, D_I) to $(\mathcal{S}_E, \mathcal{S}_D)$. So $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D) = (F^{\mathcal{S}}, \mathcal{S}_E, \mathcal{S}_D)$ where only $\mathcal{S}$ has oracle access to $(\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n})$.

We must show that the A_4's view has statistically close distribution in Game 0 and Game 1. Since the difference between the games is the underlying function, we show that the output of the functions is statistically close; this in turn shows that the A_4's view has statistically close distribution in Game 0 and Game 1. First we rewrite $\mathcal{S}$ in Fig. 4. $\mathcal{C}^3_{2n,n}$ is hard-coded in the steps e03-e05, e06-e08, d03-05 and d06-08 where $\mathsf{E2}$ and $\mathsf{D2}$ are (initially everywhere $\perp$) arrays to store the output of the ideal cipher and $\mathcal{T}_{E2}$ and $\mathcal{T}_{D2}$ are (initially everywhere empty) tables. Similarly, $\mathcal{C}^3_{2n,n}$ is hard-coded in Steps e10-e11 and d10-d11 where $\mathsf{E3}$ and $\mathsf{D3}$ are (initially everywhere $\perp$) arrays to store the output of the ideal cipher. For any k, if $\mathsf{E2}[k,x] \neq \perp$, $\mathsf{E2}[k,x] \in \mathcal{T}_{E2}[k]$, and if $\mathsf{D2}[k,y] \neq \perp$, $\mathsf{D2}[k,y] \in \mathcal{T}_{D2}[k]$.

In the following, we use the lazily-sample ideal cipher in Fig. 5. E and D are (initially everywhere $\perp$) arrays and $\mathcal{T}_E$ and $\mathcal{T}_D$ (initially empty) tables. For any (k,x) such that $\mathsf{E}[k,x] \neq \perp$, $\mathsf{E}[k,x]$ is stored in $\mathcal{T}_E[k]$, and for any (k,y) such that $\mathsf{D}[k,y] \neq \perp$, $\mathsf{D}[k,y]$ is stored in $\mathcal{T}_D[k]$. On a query which the key element is k, first the output of $E_I(k,c_1)$ is determined (steps 03-04 or steps 13-14) and second the output of $E_I(k,c_2)$ is determined (Steps 05-06 or Steps 15-16). Then the outputs of $E_I(k,x)$ such that $x \neq c_1$ and $x \neq c_2$ are determined. Since no adversary (distinguisher) learns $E_I(k,c_1)$ and $E_I(k,c_2)$ until querying the corresponding value, the procedures of the steps 03-06 and 13-16 do not affect the lazily-sample ideal cipher simulation.

We compare the simulator with the lazily-sample ideal cipher. In the simulator and the ideal cipher, $\mathsf{E}[k,c_1]$ and $\mathsf{E}[k,c_2]$ (and also $\mathsf{D}[k,\mathsf{E}[k,c_1]]$ and $\mathsf{D}[k,\mathsf{E}[k,c_2]]$) are chosen from the same distribution, while

[7] If c_1 and c_2 which are equal to $C \oplus IV[0]$ or $IV[0]$ are used, $\mathcal{S}$ cannot decide whether using $E2_I$ or $E3_I$.

```
simulator $\mathcal{S}_E(k, x)$
e01 If $\mathsf{E}[k, x] \neq \perp$ then ret $\mathsf{E}[k, x]$;
e02 If $\mathsf{E}[k, c_1] = \perp$,
e03     $y_1 \xleftarrow{\$} \{0,1\}^n$;
e04     $\mathsf{E3}[k, c_1] \leftarrow y_1$; $\mathsf{D3}[k, y_1] \leftarrow c_1$;
e05     $\mathsf{E}[k, c_1] \leftarrow \mathsf{E3}[k, c_1]$; $\mathsf{D}[k, y_1] \leftarrow c_1$;
e06     $y_2 \xleftarrow{\$} \{0,1\}^n \backslash \{y_1\}$;
e07     $\mathsf{E3}[k, c_2] \xleftarrow{\$} y_2$; $\mathsf{D3}[k, y_2] \leftarrow c_2$;
e08     $\mathsf{E}[k, c_1] \leftarrow \mathsf{E3}[k, c_1]$; $\mathsf{D}[k, y_2] \leftarrow c_1$;
e09 If $x \neq c_1$ and $x \neq c_2$,
e10     $y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{T}_{E2}[k]$;
e11     $\mathsf{E2}[k, x] \leftarrow y$; $\mathsf{D2}[k, y] \leftarrow x$;
e12     $\mathsf{E}[k, x] \leftarrow y$; $\mathsf{D}[k, y] \leftarrow x$;
e13 Ret $\mathsf{E}[k, x]$;
```

```
simulator $\mathcal{S}_D(k, y)$
d01 If $\mathsf{D}[k, y] \neq \perp$ then ret $\mathsf{D}[k, y]$;
d02 If $\mathsf{E}[k, c_1] = \perp$,
d03     $y_1 \xleftarrow{\$} \{0,1\}^n$;
d04     $\mathsf{E3}[k, c_1] \leftarrow y_1$; $\mathsf{D3}[k, y_1] \leftarrow c_1$;
d05     $\mathsf{E}[k, c_1] \leftarrow \mathsf{E3}[k, c_1]$; $\mathsf{D}[k, y_1] \leftarrow c_1$;
d06     $y_2 \xleftarrow{\$} \{0,1\}^n \backslash \{y_1\}$;
d07     $\mathsf{E3}[k, c_2] \xleftarrow{\$} y_2$; $\mathsf{D3}[k, y_2] \leftarrow c_2$;
d08     $\mathsf{E}[k, c_1] \leftarrow \mathsf{E3}[k, c_1]$; $\mathsf{D}[k, y_2] \leftarrow c_1$;
d09 If $\mathsf{D}[k, y] = \perp$,
d10     $x \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{T}_{D2}[k]$;
d11     $\mathsf{E2}[k, x] \leftarrow y$; $\mathsf{D2}[k, y] \leftarrow x$;
d12     $\mathsf{E}[k, x] \leftarrow y$; $\mathsf{D}[k, y] \leftarrow x$;
d13 Ret $x$;
```

Fig. 4. Revised Simulator

```
Encryption Oracle $E_I(k, x)$
01 If $\mathsf{E}[k, x] \neq \perp$, ret $\mathsf{E}[k, x]$;
02 If $\mathsf{E}[k, c_1] = \perp$,
03     $\mathsf{E}[k, c_1] \xleftarrow{\$} \{0,1\}^n$;
04     $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
05     $\mathsf{E}[k, c_2] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$;
06     $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
07 If $x \neq c_1$ and $x \neq c_2$,
08     $\mathsf{E}[k, x] \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{T}_E[k]$;
09     $\mathsf{D}[k, \mathsf{E}[k, x]] \leftarrow x$;
10 Ret $\mathsf{E}[k, x]$;
```

```
Decryption Oracle $D_I(k, y)$
11 If $\mathsf{D}[k, y] \neq \perp$, ret $\mathsf{D}[k, y]$;
12 If $\mathsf{E}[k, c_1] = \perp$,
13     $\mathsf{E}[k, c_1] \xleftarrow{\$} \{0,1\}^n$;
14     $\mathsf{D}[k, \mathsf{E}[c_1, x]] \leftarrow c_1$;
15     $\mathsf{E}[k, c_2] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathsf{E}[k, c_1]\}$;
16     $\mathsf{D}[k, \mathsf{E}[c_2, x]] \leftarrow c_2$;
17 If $\mathsf{D}[k, y] \neq \perp$,
18     $\mathsf{D}[k, y] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathcal{T}_D[k]\}$;
19     $\mathsf{E}[k, \mathsf{D}[k, y]] \leftarrow y$;
20 Ret $\mathsf{D}[k, y]$;
```

Fig. 5. Lazily-Sample Ideal Cipher

$\mathsf{E}[k, x]$ (and $\mathsf{D}[k, \mathsf{E}[k, x]]$) where $x \neq c_1$ and $x \neq c_2$ is chosen different distribution. If in the step e10 y is randomly chosen from $\mathcal{T}_{E2}[k] \cup \{\mathsf{E}[k, c_1], \mathsf{E}[k, c_2]\}$ and in the step d10 x is randomly chosen from $\mathcal{T}_{D2}[k] \cup \{c_1, c_2\}$, then the output distribution of the simulator and the ideal cipher is the same. That is, if any value y randomly chosen from $\{0,1\}^n \backslash \mathcal{T}_{E2}[k]$ does not collide $\mathsf{E}[k, c_1]$ and $\mathsf{E}[k, c_2]$ and any value x randomly chosen from $\{0,1\}^n \backslash \mathcal{T}_{D2}[k]$ does not collide c_1 and c_2, then the output distribution between them is the same. Since for any k, the number of values in $\mathcal{T}_{E2}[k]$ and $\mathcal{T}_{D2}[k]$ is at most $2lq_F + q_E + q_D$, the statistical distance of $\mathsf{E}[k, x]$ (and $\mathsf{D}[k, \mathsf{E}[k, x]]$) where $x \neq c_1$ and $x \neq c_2$ is at most $2/(2^n - (2lq_F + q_E + q_D))$. So the statistical distance of the simulator and the ideal cipher is at most $(2lq_F + q_E + q_D) \times 2/(2^n - (2lq_F + q_E + q_D))$. We thus have that

$$|\Pr[G1] - \Pr[G0]| \leq \frac{2 \times (2lq_F + q_E + q_D)}{2^n - (2lq_F + q_E + q_D)}.$$

Game 2: We modify $\mathcal{O}_F$ from $F^{\mathcal{S}}$ to $F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}$. So $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D) = (F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}, \mathcal{S}_E, \mathcal{S}_D)$ and this is the F_2 scenario.

We show that unless the following bad events occur, the A_4's view of Game 1 and Game 2 is the same.

- Event B1: On some query (k, x) to $\mathcal{S}_E$, the output y is such that $y \oplus x$ is equal to c_1 or c_2.
- Event B2: On some query (k, x) to $\mathcal{S}_E$, the output y is such that $y \oplus x \oplus C$ is equal to c_1 or c_2.
- Event B3: On some query (k, y) to $\mathcal{S}_D$, the output x is equal to c_1 or c_2 such that x is defined in the step D08.

To prove this, we use the proof method in [5, 15]. Specifically, we prove the following two points.

1. In Game 1, unless the bad events occur, for any query M the output of $\mathcal{O}_F(M)$ is equal to that of $F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}(M)$. If this holds, the output distribution of $\mathcal{O}_F$ in Game 1 and Game 2 is equivalent.

2. In Game 2, unless the bad events occur, $\mathcal{O}_E$ and $\mathcal{O}_D$ are consistent with $\mathcal{O}_F$ as in Game 1. $\mathcal{O}_F$ uses $\mathcal{O}_E$ in Game 1 while does not in Game 2 (note that in both games $(\mathcal{O}_E, \mathcal{O}_D) = (\mathcal{S}_E, \mathcal{S}_D)$). So if this holds, the difference does not affect the output distribution of $\mathcal{O}_E$ and $\mathcal{O}_D$, namely, the output distribution of $\mathcal{O}_E$ and $\mathcal{O}_D$ in Game 1 and Game 2 is the same.

In the following, for input-output triple (k, x, y) of $\mathcal{S}$ we denote $x \oplus y$ by w, namely, $w = x \oplus y$. Before proving the above two points, we define chain triples and give a useful lemma.

Definition 3. $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i), (k'_1, x'_1, y'_1), \ldots, (k'_i, x'_i, y'_i), (k, x, y), (k', x', y')$ *stored in the simulator's tables* E, D *are chain triples if for some* M *the output of* $F^{\mathcal{S}}(M)$ *can be obtained from the triples. That is,* $x_1 = IV[0], k_1[0] = IV[1]$*,* $k_j = k'_j$ $(j = 1, \ldots, i)$*,* $w_j = x_{j+1}$ $(j = 1, \ldots, i-1)$*,* $w_j \oplus C = x'_{j+1}$ $(j = 1, \ldots, i-1)$*,* $w'_j = k_{j+1}[0]$ $(j = 1, \ldots, i-1)$*,* $x = c_1$*,* $x' = c_2$*,* $k = k'$*,* $k[0] = w_i$*,* $k[1] = w'_i$*,* $M = k_1[1] || \cdots || k_i[1]$*, and* $y || y' = F^{\mathcal{S}}(M)$*.*

Lemma 7. *For any chain triple* $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i), (k'_1, x'_1, y'_1), \ldots, (k'_i, x'_i, y'_i), (k, x, y), (k', x', y')$*, unless the bad events occur,* $F^{\mathcal{S}}(M) = F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}(M)$ *where* $M = k_1[1] || \cdots || k_i[1]$*.*

Proof. To contrary, assume that there exist chain triples $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i)$, $(k'_1, x'_1, y'_1), \ldots, (k'_i, x'_i, y'_i)$, (k, x, y), (k', x', y') such that $F^{\mathcal{S}}(M) \neq F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}(M)$ where $M = k_1[1] || \cdots || k_i[1]$. Then, since the output of $\mathcal{S}$ is defined by $E2_I$ or $E3_I$, one of the following events occur.

- Event 1: In the inner calculation of $F^{\mathcal{S}}(M)$, some triple is defined by $E3_I$. That is, some of (k_1, x_1, y_1), $\ldots$, (k_i, x_i, y_i), (k'_1, x'_1, y'_1), $\ldots$, (k'_i, x'_i, y'_i), is defined by $E3_I$.
- Event 2: In the post-processing function calculation of $F^{\mathcal{S}}(M)$, some triple is defined by $E2_I$. That is, (k, x, y) or (k', x', y') is defined by $E2_I$.

Consider Event 1. First consider the case that (k_j, x_j, y_j) is defined by $E3_I$. Since $x_1 = IV[0]$, $j \neq 1$. When the output of $\mathcal{S}_E(k_j, x_j)$ is defined by $E3_I$, $x_j = c_1$ or $x_j = c_2$. Which means that $w_{j-1} = c_1$ or $w_{j-1} = c_2$. So the bad event 1 occurs. Second consider the case that (k'_j, x'_j, y'_j) is defined by $E3_I$. Similarly, since $x_1 = IV[0] \oplus C$, $j \neq 1$. When the output of $\mathcal{S}_E(k_j, x_j)$ is defined by $E3_I$, $x_j = c_1$ or $x_j = c_2$. Which means that $w'_{j-1} \oplus C = c_1$ or $w'_{j-1} \oplus C = c_2$. So the bad event 2 occurs.

Next consider Event 2. First consider the case that (k, x, y) is defined by $E2_I$. Then the triple is defined in $\mathcal{S}_D$ because $x = c_1$ (if the triple is defined in $\mathcal{S}_E$, it is defined by $E2_I$ due to the condition of the step E07). So the triple is defined in the step D08. The bad event 3 occurs. Finally, consider the case that (k', x', y') is defined by $E2_I$. Then the triple is defined in $\mathcal{S}_D$ because $x = c_2$. So the triple is defined in the step D08. The bad event 3 occurs. □

Proof of Point 1. From the above lemma, unless the bad event occurs, the output of $\mathcal{O}_F(M) = F^{\mathcal{S}}(M) = F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}(M)$.

Proof of Point 2. Since in Game 1 for any M the output of $\mathcal{O}_F(M)$ is calculated by $F^{\mathcal{S}}(M)$, we must show that in Game 2 the relation also holds, that is, unless the bad events occur, for any chain triples $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i), (k'_1, x'_1, y'_1), \ldots, (k'_i, x'_i, y'_i), (k, x, y), (k', x', y')$ the output of $F^{\mathcal{S}}(M)$ is equal to $\mathcal{O}_F(M)$ $(= F_2^{\mathcal{C}^2_{2n,n}, \mathcal{C}^3_{2n,n}}(M))$ where $M = k_1[1] || \cdots || k_i[1]$. From the above lemma, unless the bad event occurs, this holds.

The Bound of $|\Pr[G2] - \Pr[G1]|$. The above two points imply that unless the bad events occur, the A_4's view of Game 1 and Game 2 is the same, and so we have that

$$|\Pr[G2] - \Pr[G1]| \leq 2 \times \max\{\Pr[B1_1] + \Pr[B2_1] + \Pr[B3_1], \Pr[B1_2] + \Pr[B2_2] + \Pr[B3_2]\}$$

where Bi_j is the event Bi in Game j. Since the number of queries to $\mathcal{S}$ in Game 1 is more than that in Game 2,

$$|\Pr[G2] - \Pr[G1]| \leq 2 \times (\Pr[B1_1] + \Pr[B2_1] + \Pr[B3_1]).$$

First, we evaluate the probability $\Pr[B1_1]$. In Game 1, the number of queries to $\mathcal{S}$ is at most $2(lq_F+1)+q_E+q_D$. So the output is randomly chosen from at least $2^n-(2(lq_F+1)+q_E+q_D)$ values. We thus have that

$$\Pr[B1_1] \leq \frac{2 \times (2(lq_F+1)+q_E+q_D)}{2^n-(2(lq_F+1)+q_E+q_D)}.$$

Second, we evaluate the probability $\Pr[B2_1]$. From the same discussion as $\Pr[B1_1]$,

$$\Pr[B2_1] \leq \frac{2 \times (2(lq_F+1)+q_E+q_D)}{2^n-(2(lq_F+1)+q_E+q_D)}.$$

Finally, we evaluate the probability $\Pr[B3_1]$. A value in the step D08 is defined by $D2_I$. That is, in this case, the output of $D2_I$ is equal to c_1 or c_2. Since the number of queries to $\mathcal{C}^2_{2n,n}$ is at most $2(lq_F+1)+q_E+q_D$, So the output of $D2_I$ is randomly chosen from at least $2^n-(2(lq_F+1)+q_E+q_D)$ values. We thus have that

$$\Pr[B1_1] \leq (2(lq_F+1)+q_E+q_D) \times \frac{2}{2^n-(2(lq_F+1)+q_E+q_D)}.$$

We thus have that

$$|\Pr[G2]-\Pr[G1]| \leq 2 \times \frac{6 \times (2(lq_F+1)+q_E+q_D)}{2^n-(2(lq_F+1)+q_E+q_D)}$$

Consequently, we can obtain the following bound.

$$\mathsf{Adv}^{\mathsf{indif}}_{F^{\mathcal{C}_{2n,n}},F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}}(A_4) \leq \frac{14 \times (2(lq_F+1)+q_E+q_D)}{2^n-(2(lq_F+1)+q_E+q_D)}.$$

□

Using Theorem 2 and Lemma 6, we show that F is PRO up to $\mathcal{O}(2^n)$ query complexity in the IC model.

Theorem 3 (F is PRO). *There exists a simulator $S=(S_E,S_D)$ such that for any distinguisher A making at most (q_H,q_E,q_D) queries to three oracles which are (F,E_I,D_I) or $(\mathcal{F}_{2n},S_E,S_D)$, we have*

$$\mathsf{Adv}^{\mathsf{pro}}_{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},S_2}(A) \leq \frac{2Q^2}{(2^n-2Q)^2}+\frac{2Q}{2^n-2Q}+\frac{2l(2q)Q}{(2^n-Q)^2}+\frac{q_H+2q}{2^n}+\frac{14Q}{2^n-Q}.$$

where S_2 works in time $\mathcal{O}(q+2lqQ)+2lq \times \mathsf{Time}(\mathsf{unpad})$ and makes $2q$ queries to $\mathcal{F}_{2n}$ where $Q=2l(q_H+1)+q_E+q_D$ and $q=q_E+q_D$. ♦

Proof. We use Theorem 2 and Lemma 6. We define a simulator by $S=\mathcal{S}^{S_2}$ where $\mathcal{S}$ is defined in Lemma 6 and S_2 is defined in Theorem 2. Specifically, $S_E=\mathcal{S}_E^{S2,S3}$ and $S_D=\mathcal{S}_D^{S2,S3}$. In the following, we evaluate the PRO bound of $F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}$. We assume that A_4 is a distinguisher such that the indifferentiable bound of F from F_2 is maximum, and A_3 is a distinguisher such that the PRO bound of F_2 is maximum. For any distinguisher A,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{indif}}_{F^{\mathcal{C}_{2n,n}},S}(A) =& |\Pr[A^{F^{\mathcal{C}_{2n,n}},\mathcal{C}_{2n,n}} \Rightarrow 1]-\Pr[A^{\mathcal{F}_{2n},S^{\mathcal{F}_{2n}}} \Rightarrow 1]| \\
\leq& |\Pr[A^{F^{\mathcal{C}_{2n,n}},\mathcal{C}_{2n,n}} \Rightarrow 1]-\Pr[A^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}} \Rightarrow 1]| \\
&+|\Pr[A^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}} \Rightarrow 1]-\Pr[A^{\mathcal{F}_{2n},\mathcal{S}^{S_2^{\mathcal{F}_{2n}}}} \Rightarrow 1]| \\
\leq& |\Pr[A_4^{F^{\mathcal{C}_{2n,n}},\mathcal{C}_{2n,n}} \Rightarrow 1]-\Pr[A_4^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}}} \Rightarrow 1]| \\
&+|\Pr[A_3^{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}} \Rightarrow 1]-\Pr[A_3^{\mathcal{F}_{2n},S_2^{\mathcal{F}_{2n}}} \Rightarrow 1]| \\
=&\mathsf{Adv}^{\mathsf{indif}}_{F^{\mathcal{C}_{2n,n}},F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},\mathcal{S}}(A_4)+\mathsf{Adv}^{\mathsf{pro}}_{F_2^{\mathcal{C}^2_{2n,n},\mathcal{C}^3_{2n,n}},S_2}(A_3).
\end{aligned}$$

From the second equation to the third equation, we change A to A_4 and $A^{\mathcal{O}_a,\mathcal{S}^{\mathcal{O}_b,\mathcal{O}_c}}$ to $A_3^{\mathcal{O}_a,\mathcal{O}_b,\mathcal{O}_c}$. Note that $S_2 = (S2, S3)$ which simulate ideal ciphers $\mathcal{C}^2_{2n,n}$ and $\mathcal{C}^3_{2n,n}$. The third equation holds because A_4 and A_3 are distinguishers where the indifferentiability bound and the PRO bound are maximum.

Thus, when using the bounds of Theorem 2 and Lemma 6, the PRO bound of F can be obtained where A_4 can make at most (q_H, q_E, q_D) queries to its three oracles (see Lemma 6) and A_3 can make at most $(q_H, q_E, q_D, 2(q_E + q_D), 0)$ queries to its five oracles (see Theorem 2). □

References

1. Mihir Bellare and Thomas Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2006.
2. Bruno O. Brachtl, Don Coppersmith, Myrna M. Hyden, Stephen M. Matyas Jr, Carl H. W. Meyer, Jonathan Oseas, Shaiy Pilpel, and Michael Schilling. Data authentication using modification detection codes based on a public one way encryption function. US Patent No. 4,908,861, 1990 (filed August 28, 1987).
3. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS 2001*, volume 2001, pages 136–145, 2001.
4. Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. In *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2006.
5. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.
6. Ivan Damgård. A Design Principle for Hash Functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
7. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009.
8. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In *ePrint 2009/177*, 2009.
9. Ewan Fleischmann, Christian Forler, Michael Gorski, and Stefan Lucks. Collision Resistant Double-Length Hashing. In *ProvSec*, volume 6402 of *Lecture Notes in Computer Science*, pages 102–118. Springer, 2010.
10. Ewan Fleischmann, Michael Gorski, and Stefan Lucksl. On the Security of Tandem-DM. In *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 2009.
11. Ewan Fleischmann, Michael Gorski, and Stefan Lucksl. Security of Cyclic Double Block Length Hash Functions. In *IMA Int. Conf*, volume 5921 of *Lecture Notes in Computer Science*, pages 153–175. Springer, 2009.
12. Zheng Gong, Xuejia Lai, and Kefei Chen. A synthetic indifferentiability analysis of some block-cipher-based hash functions. In *Des. Codes Cryptography 48*, pages 293–305, 2008.
13. Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.
14. Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007.
15. Jonathan J. Hoch and Adi Shamir. On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak. In *ICALP*, Lecture Notes in Computer Science, pages 616–630. Springer, 2008.
16. Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.
17. Jooyoung Lee and Daesung Kwon. The Security of Abreast-DM in the Ideal Cipher Model. IEICE Transactions 94-A(1), pages 104–109. IEICE, 2011.
18. Jooyoung Lee and Martijn Stam. Mjh: A faster alternative to mdc-2. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236. Springer, 2011.
19. Jooyoung Lee, Martijn Stam, and John Steinberger. The collision security of Tandem-DM in the ideal cipher model. ePrint 2010/409, 2010.
20. Stefan Lucks. A collision-resistant rate-1 double-block-length hash function. In *Symmetric Cryptography*, Symmetric Cryptography, Dagstuhl Seminar Proceedings 07021, 2007.
21. S. Matyas, C. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithms. In *IBM Technical Disclosure Bulletin 27(10a)*, pages 5658–5659, 1985.
22. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

23. Ralph C. Merkle. One Way Hash Functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
24. Carl H. W. Meyer and Michael Schilling. Chargement securise d'un programma avec code de detection. 1987.
25. National Institute of Standards and Technoloty. FIPS PUB 180-3 Secure Hash Standard. In *FIPS PUB*, 2008.
26. Onur Özen and Martijn Stam. Another Glance at Double-Length Hashing. In *IMA Int. Conf*, volume 5921 of *Lecture Notes in Computer Science*, pages 176–201. Springer, 2009.
27. Bart Preneel, Antoon Bosselaers, Rene Govaerts, and Joos Vandewalle. Collision-free Hashfunctions Based on Blockcipher Algorithmsl. In *Proceedings of 1989 International Carnahan Conference on Security Technology*, pages 203–210, 1989.
28. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.
29. Ronald L. Rivest. The MD4 Message Digest Algorithm. In *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer, 1990.
30. Ronald L. Rivest. The MD5 Message Digest Algorithm. In *RFC 1321*, 1992.
31. John P. Steinberger. The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2007.

A Confirmation of the Security against an Inversion Attack

Several schemes using a blockcipher, e.g., the Davies-Meyer mode are not indifferentiable from random oracles. For example, the differentiable attack of the Davies-Meyer mode uses the property of the decryption oracle [5]. So readers may think that the following attack, which is the same as the attack for the Davies-Meyer mode, can be applied to our hash functions. However we note that the attack clearly does not work.

First consider the PRO security of the post-processing function $f^{\mathcal{C}_{2n.n}}(x) = E_I(c_1, x)||E_I(c_2, x)$. Let $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D)$ be $(f^{\mathcal{C}_{2n,n}}, E_I, D_I)$ or (g, S_E, S_D) where (S_E, S_D) is a simulator for the ideal cipher $\mathcal{C}_{2n,n}$. For a $2n$-bit value w we denote the first n-bit value by $w[0]$ and the last n-bit value by $w[1]$. Then an attack based on the decryption oracle is as follows. q is the loop number which is depend on the total number of queries made by A.

```
Adversary $A^{\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D}$
for $j = 1, \ldots, q$
    $i \xleftarrow{\$} \{0,1\}$;
    $x \xleftarrow{\$} \{0,1\}^{2n}$; $w \leftarrow \mathcal{O}_f(x)$;
    If $i = 0$, $w \xleftarrow{\$} \{0,1\}^{2n}$;
    $x_1 \leftarrow \mathcal{O}_D(k, w[0])$; $x_2 \leftarrow \mathcal{O}_D(k, w[1])$;
    If $i = 1$, and $x_1 \neq c_1$ or $x_2 \neq c_2$, ret 0;
Ret 1;
```

When $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (f^{\mathcal{C}_{2n,n}}, E_I, D_I)$, clearly A outputs 1. When $(\mathcal{O}_f, \mathcal{O}_E, \mathcal{O}_D) = (g, S_E, S_D)$, we can construct a simulator S such that A outputs 1 with probability almost 1, since on query (k, w') to S_D, S_D can know the output of $g(k)$, if $w' = z[0]$ or $w' = z[1]$ where $z = g(k)$, she outputs c_1 or c_2, respectively, otherwise outputs a random value by the ideal cipher simulation. Thus the PRO advantage for A is negligible and the post-processing function resists the differentiable attack. Please see the PRO security analysis of the function in Subsection 3.2.

Next consider the PRO security of our hash function. Let $(\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D)$ be $(F^{\mathcal{C}_{2n,n}}, E_I, D_I)$ or $(\mathcal{F}_{2n}, S_E, S_D)$ where S_E is the simulator of E_I and S_D is the simulator of D_I. q is the loop number which is depend on the total number of queries made by A.

```
Adversary $A^{\mathcal{O}_F, \mathcal{O}_E, \mathcal{O}_D}$
for $j = 1, \ldots, q$
    $i \xleftarrow{\$} \{0,1\}$;
    $M \xleftarrow{\$} \{0,1\}^{2n}$; $w \leftarrow \mathcal{O}_F(M)$;
    If $i = 0$, $w \xleftarrow{\$} \{0,1\}^{2n}$;
    $x_1 \leftarrow \mathcal{O}_D(k, w[0])$; $x_2 \leftarrow \mathcal{O}_D(k, w[1])$;
    If $i = 0$, $x_1 \neq c_1$ or $x_2 \neq c_2$, ret 0;
Ret 1;
```

The value k used in line 05 is the first input of the post-processing function of $\mathcal{O}_F(M)$. Note that we don't write the procedure to know k. So there are the two cases: (case 1) A makes queries to $\mathcal{O}_E$ or $\mathcal{O}_D$ to know k and (case 2) A does not make the queries. In the case 2 the above attack explicitly does not work. So consider the case 1. One may think that since i is a random value, no simulator can know the line wherein w is defined, and by using this fact, A can distinguish $(F^{\mathcal{C}_{2n,n}}, E_I, D_I)$ from $(\mathcal{F}_{2n}, S_E, S_D)$. However, since the compression functions used in the inner calculation is PrA, in the case 1 the simulator can know M from k, and thus, the simulator can know the value i. That is the simulator can know the line. So the attack does not work.

B Abreast-DM Is PrA

Abreast-DM [16] incorporates two Davies-Meyer (DM) single block length compression functions which are used side-by-side. The compression function is formally given in Definition 4.

Definition 4. *Let $\mathsf{BC}_{2n,n} = (E, D)$ be a blockcipher. Let $\mathsf{CF}^{\mathsf{ADM}}[\mathsf{BC}_{2n,n}] : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a compression function such that $(G_i, H_i) = \mathsf{CF}^{\mathsf{ADM}}[\mathsf{BC}_{2n,n}](G_{i-1}||H_{i-1}, M_i)$ where $G_i, H_i, M_i, G_{i-1}, H_{i-1} \in \{0,1\}^n$. (G_i, H_i) is calculated as follows:*

$$\begin{aligned} G_i &= G_{i-1} \oplus E(H_{i-1}||M_i, G_{i-1}) \\ H_i &= H_{i-1} \oplus E(M_i||G_{i-1}, \overline{H_{i-1}}) \end{aligned}$$

where $\overline{H}$ denotes the bit-by-bit complement of H. We call the first procedure "first block" and the second procedure "second block".

We show that the Abreast-DM compression function is PrA with $\mathcal{O}(2^n)$ security.

Theorem 4 (Abreast-DM is PrA). *Let $\mathcal{C}_{2n,n} = (E_I, D_I)$ be an ideal cipher. There exists an extractor $\mathcal{E}$ such that for any adversary A making at most q_P queries to $\mathcal{C}_{2n,n}$ and q_e extraction queries we have*

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{CF}^{\mathsf{ADM}}[\mathcal{C}_{2n,n}],\mathcal{C}_{2n,n},\mathcal{E}}(A) \le 18\left(\frac{q_P}{2^{n-1}}\right)^2 + \frac{2q_Pq_e}{(2^n - q_P)^2}$$

where $\mathcal{E}$ runs in time at most $\mathcal{O}(q_eq_P)$.

Proof. We will prove that any such compression function is 1-WPrA, and then Lemma 1 gives the final bound. We note that Theorem 1 of [11] upperbounds the cr-advantage by $18(q_P/2^{n-1})^2$, yielding the first term above. Note that the cr-advantage is also bounded by the result of [17]. Let us define the multi-point extractor $\mathcal{E}^+$ as follows.

```
algorithm $\mathcal{E}^+(z, \alpha)$
Let $L$ be an empty list;
Parse $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i) \leftarrow \alpha$;
For $j = 1$ to $i$ do
    If $z[0] = x_j \oplus y_j$ then
        $y \leftarrow E_I(k_j[1]||x_j, \overline{k_j[0]})$;
        If $z[1] = \overline{k_j[0]} \oplus y$ then $L \overset{\cup}{\longleftarrow} (x_j||k_j[0], k_j[1])$;
    If $z[1] = x_j \oplus y_j$ then
        $y \leftarrow E_I(\overline{x_j}||k_j[0], k_j[1])$;
        If $z[0] = k_j[1] \oplus y$ then $L \overset{\cup}{\longleftarrow} (k_j[1]||\overline{x}_j, k_j[0])$;
If $L$ is not an empty list then return $L$ and otherwise return $\perp$;
```

If an input-output triple of the first block is defined, automatically the input of the second block is defined, and vice versa, from the definition of the compression function. For a query (z, α) to $\mathcal{E}^+$, when there is an input-output triple (k, x, y) such that $x \oplus y = z[0]$, the multi-point extractor $\mathcal{E}^+$ checks whether the output of the second block is equal to $z[1]$ or not and if this holds the multi-point extractor stores it in the return list L, and vice versa. Therefore, A must find a preimage (k, x) of z to win the 1-WPrA experiment. Thus one can straightforwardly adapt the preimage resistant advantage of the compression function (Theorem 2 in [11]). The advantage is at most $2q_P/(2^n - q_P)^2$. □

C Tandem-DM Is PrA

Tandem-DM [16] incorporates two Davies-Meyer (DM) single block length compression functions which are used side-by-side. The compression function is formally given in Definition 5.

Definition 5. *Let* $\mathsf{BC}_{2n,n} = (E, D)$ *be a blockcipher. Let* $\mathsf{CF}^{\mathsf{TDM}}[\mathsf{BC}_{2n,n}] : \{0,1\}^{2n} \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$ *be a compression function such that* $(G_i, H_i) = \mathsf{CF}^{\mathsf{TDM}}[\mathsf{BC}_{2n,n}](G_{i-1}||H_{i-1}, M_i)$ *where* $G_i, H_i, M_i, G_{i-1}, H_{i-1} \in \{0,1\}^n$. (G_i, H_i) *is calculated as follows:*

$$W_i = E(H_{i-1}||M_i, G_{i-1}) \quad (3)$$

$$G_i = G_{i-1} \oplus W_i \quad (4)$$

$$H_i = H_{i-1} \oplus E(M_i||W_i, H_{i-1}) \quad (5)$$

We call the procedures of 3 and 4 "first block" and the procedures of 5 "second block".

We show that the Tandem-DM compression function is PrA with $\mathcal{O}(2^n)$ security.

Theorem 5 (Tandem-DM is PrA). *Let* $\mathcal{C}_{2n,n} = (E_I, D_I)$ *be an ideal cipher. There exists an extractor* $\mathcal{E}$ *such that for any adversary* A *making at most* q_P *queries to* $\mathcal{C}_{2n,n}$ *and* q_e *extraction queries we have*

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{CF}^{\mathsf{TDM}},\mathcal{C}_{d,n},\mathcal{E}}(A) \leq p + \frac{2q_P q_e}{(2^n - q_P)^2}$$

where $\mathcal{E}$ *runs in time at most* $\mathcal{O}(q_e q_P)$ *and* p *is the cr-advantage of Tandem-DM described in Theorem 1 of [19].*

Proof. We will prove that any such compression function is 1-WPrA, and then Lemma 1 to give the final bound. We note that Theorem 1 of [10] upperbounds the cr-advantage by p, yielding the terms excluding the last term. Let us define the multi-point extractor $\mathcal{E}^+$ as follows:

```
algorithm $\mathcal{E}^+(z, \alpha)$
$L$ be an empty list;
Parse $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i) \leftarrow \alpha$;
For $j = 1$ to $i$ do
    If $z[0] = x_j \oplus y_j$ then
        $y \leftarrow E_I(k_j[1]||y_j, k_j[0])$;
        If $z[1] = k_j[0] \oplus y$ then $L \overset{\cup}{\leftarrow} (x_j||k_j[0], k_j[1])$;
    If $z[1] = x_j \oplus y_j$ then
        $x \leftarrow D_I(x_j||k_j[0], k_j[1])$;
        If $z[0] = k_j[1] \oplus x$ then $L \overset{\cup}{\leftarrow} (x||x_j, k_j[0])$;
If $L$ is an empty list then return $L$ otherwise return $\perp$;
```

If an input-output triple of the first block is defined, automatically the input triple of the second block is defined, and vice versa, from the definition of the compression function. For a query (z, α) to $\mathcal{E}^+$, when there is an input-output triple (k, x, y) such that $x \oplus y = z[0]$, the multi-point extractor $\mathcal{E}^+$ checks whether the output of the second block is equal to $z[1]$ or not and if this holds the multi-point extractor stores it in the return list L, and vice versa. Therefore, A must find a preimage (k, x) of z to win the 1-WPrA experiment. Then one can straightforwardly adapt the preimage resistant advantage of Tandem-DM (Theorem 2 in [10]). This advantage is at most $2q_P/(2^n - q_P)^2$. □