J. Cryptol. (2012) 25: 271–309
DOI: 10.1007/s00145-010-9095-5

Journal of
CRYPTOLOGY

A Simple Variant of the Merkle–Damgård Scheme with a Permutation*

Shoichi Hirose
Graduate School of Engineering, University of Fukui, Fukui, Japan
hrs_shch@u-fukui.ac.jp

Je Hong Park
Electronics and Telecommunications Research Institute, Daejeon, Korea
jhpark@ensec.re.kr

Aaram Yun
School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, Ulsan, Korea
aaramyun@unist.ac.kr

Communicated by Philip Rogaway

Received 6 November 2009
Online publication 11 December 2010

Abstract. We propose a new composition scheme for hash functions. It is a variant of the Merkle–Damgård construction with a permutation applied right before the processing of the last message block. We analyze the security of this scheme using the indifferentiability formalism, which was first adopted by Coron et al. to the analysis of hash functions. We also study the security of simple MAC constructions out of this scheme. Finally, we discuss the random oracle indifferentiability of this scheme with a double-block-length compression function or the Davies–Meyer compression function composed of a block cipher.

Key words. Hash function, Merkle–Damgård construction, Random oracle, Ideal cipher, Indifferentiability, Pseudorandom function, MAC.

1. Introduction

Background Merkle–Damgård [16,26] is an iterative hash function construction. Given a fixed-input-length (FIL) compression function, it combines the output of the compression function in a serial fashion to produce an arbitrary-input-length (AIL) or a

* A preliminary version of this work appeared in the proceedings of the ASIACRYPT 2007 conference [18].

variable-input-length (VIL)[1] hash function. While it is a clean design with proven collision resistance, it suffers from the extension property; one can compute $H(M_1 \| M_2)$ from M_2, $H(M_1)$, and the length of M_1.

Suppose that we try to use a Merkle–Damgård (MD) hash function for message authentication. There are many proposals for hash-based MACs, but currently the most popular hash-based MAC is definitely HMAC [3,4]. It has a simple structure, and also it has rigorous security proofs. However, given a hash function H, one of the best ways to make a MAC out of H is the prefix construction [30]:

$$M_K(x) \stackrel{\text{def}}{=} H(K \| x).$$

The above construction is more efficient than HMAC especially for short messages: HMAC requires one more hash computation with an input of fixed length which depends on the key length. We also know that it gives a secure MAC if H is a random oracle rather than a concrete hash algorithm. Unfortunately, due to the extension property, the prefix construction is not secure when the underlying hash function is an MD hash function; given a message x and its MAC $M_K(x) = H(K \| x)$, the attacker can easily forge another message x', which has x as its prefix, and compute the MAC $M_K(x')$.

The goal of HMAC was to design an efficient MAC with security proofs, out of already widely deployed MD hash functions. Therefore, the designers of HMAC did not modify the underlying hash function and instead designed HMAC so that it is secure even when an MD hash function with the extension property is used.

We may consider another way, namely, to start freshly with a hash function design without such structural flaws like the extension property. Then perhaps we may use much simpler hash-based MACs such as the prefix construction $H(K \| M)$. Indeed, after Wang's attacks on many popular hash functions, there are renewed interests in the design of hash functions. So this would be a good opportunity to consider an alternative to the MD scheme.

In CRYPTO 2005, Coron et al. introduced a new methodology for assessing generic, structural properties of hash function constructions [15]. They applied the notion of indifferentiability, which was first introduced by Maurer et al. [23], to the analysis of hash functions. Coron et al. analyzed the structural property of hash function constructions by first swapping the underlying compression function with an FIL random oracle, then comparing the pair of the resulting hash function and the FIL random oracle with a pair of a VIL random oracle and a simulator of the FIL random oracle. If no efficient distinguisher can tell them apart, then the construction is considered secure, i.e., it has no structural flaws. The notion of indifferentiability is an appropriate framework to express these ideas rigorously. In fact, Coron et al. showed that the MD scheme is *not* indifferentiable from a random oracle and suggested a few modifications for the MD scheme so that all of these are indifferentiable from a random oracle.

Hence, we now have a rigorous methodology for assessing the structural flaws of a hash function, such as the extension property of the MD scheme, which was the main obstacle for adopting the simple constructions like the prefix construction instead of

[1] Or up to some large number ($2^{64} - 1$ in case of SHA-1, for example) depending on the padding and other specific details.

HMAC. Now all we need is an actual design for hash function composition scheme which is efficient and structurally sound (in the sense of random oracle indifferentiability), and which admits a direct and efficient usage as a MAC. Then in the future hash function design, we may adopt such a construction as an alternative to the MD scheme.

Our Contribution We propose a simple and efficient hash composition scheme. We call it Merkle–Damgård with Permutation (MDP). It is almost identical to the plain Merkle–Damgård scheme, but just before the last message block is processed, a permutation π is applied: for a message $M = M_1 M_2 \cdots M_k$,

$$H(M) = F\big(\pi\big(F\big(\cdots F\big(F(IV, M_1), M_2\big)\cdots, M_{k-1}\big)\big), M_k\big)$$

if $k \geq 2$, and $H(M) = F(\pi(IV), M_1)$ if $k = 1$. IV is an initial value. Note that π should be a permutation with at most few fixed points. We prove that it satisfies many desirable security properties:

- It is collision-resistant if the underlying compression function is.
- It is indifferentiable from a random oracle when a FIL random oracle is used as the compression function.
- It is a pseudorandom function (PRF) when keyed via the IV (IV is replaced by a secret key) if the compression function is a PRF under a very mild related-key attack when keyed via the chaining variable. In addition, if the compression function is also a PRF when keyed via the input message block, then MDP yields a PRF when the key is prepended to the message: $M \mapsto H(K\|M)$ for a secret key K.
- It is unforgeable if the underlying compression function is an unforgeable FIL MAC in a dedicated key setting. In this setting, each compression function F in the computation of H has a secret key to H as a part of its input.

Despite the miniscule modification MDP makes to the original MD scheme, we see that it has many benefits. MDP loses essentially none of the efficiency of the MD scheme. As categorized above, MDP preserves collision resistance, random oracle, and unforgeability. Furthermore it "almost" preserves PRF property, with a weak related-key assumption. So not only it gives a strong hash function, but, as a PRF, it also gives a secure and efficient MAC mechanism such as the prefix construction.

We also study the random-oracle indifferentiability of MDP when the underlying compression function has some structure; we consider MDP with two specific types of compression functions. One is a double-block-length (DBL) compression function of the form $F(s\|x) = f(s\|x)\|f(p(s)\|x)$, where f is a compression function, and p is a permutation. The other is the Davies–Meyer compression function. We show that MDP emulates a VIL random oracle if

- f is a random oracle and π and p are chosen appropriately in the DBL compression function F, or
- F is the Davies–Meyer compression function in the ideal cipher model.

Related Work A hash function composition scheme very similar to MDP was suggested before; in a public comment to a FIPS 180-2 draft, Kelsey [19] proposed a simple enhancement to SHA-2 hash functions, which was originally suggested by Ferguson.

Their scheme is a special case of MDP when the permutation $\pi(x) = x \oplus C$, where C is a fixed and nonzero offset. Their motivation was to eliminate the extension property of MD hash functions with least modification. But, as far as the authors know, the security of this proposal was never rigorously proven before.

While proposing indifferentiability from a random oracle as an important security goal for a hash function, Coron et al. also proposed four modified MD schemes which satisfy indifferentiability from a random oracle: with prefix-free encoding, chopping off some output bits (chop-MD), based on NMAC or HMAC [15]. Bellare and Ristenpart also proposed a variant of the MD scheme called EMD (Enveloped MD) [7]. MDP achieves essentially the same goals (multi-property preservation) as EMD, but there are a few differences:

- The structure of MDP is simpler than that of EMD; this is reflected in the fact that MDP is slightly more efficient than EMD, especially for short messages.
- When used as a MAC by key-via-IV strategy, MDP needs slightly stronger assumption than EMD; assuming that the compression function is secure as a PRF under a very weak related-key attack, we prove that the keyed MDP is secure as a PRF. Therefore, at least for the PRF property, MDP is *not* a "multi-property-preserving" transform like EMD.
- On the other hand, MDP needs only one key in the above situation, while EMD needs two separate keys. One may consider a one-key version of EMD by employing some key derivation function similar to the case of HMAC. However, then one would need an additional assumption on the compression function, namely PRF security under some related-key attack, which is essentially the same type of assumption needed for MDP.
- Given an MDP hash function H, one can use H as a black-box to obtain a PRF by the prefix construction $H(K \| M)$. This seems to be difficult in the case of EMD.

Chang et al. [12] further discussed the indifferentiability from a random oracle for the MD scheme with prefix-free encoding. They considered compression functions consisting of a block cipher [28] and DBL compression functions of the same form we considered. Nandi [27] introduced this formalization of a class of DBL compression functions and discussed the collision-resistance of hash functions composed of them.

In studying MAC properties of MDP, we follow two directions. First, we show that MDP gives a very efficient MAC by showing its pseudorandomness under the assumption that the compression function is secure as a PRF against a mild form of related-key attacks. For this, we use a restricted version of the notion of PRF security against related-key attacks formalized and studied by Bellare and Kohno [6]. Essentially, the proof can be considered as a related-key version of the proof for prefix-free PRF security of the cascade construction given in [5].

We are also interested in seeing whether security of MDP as MAC can be proved under weaker assumptions, similarly to the security of HMAC under a weaker-than-PRF assumption on the compression function [3]. After An and Bellare [2] initiated such investigations, Maurer and Sjödin [24] provided several transforms and a general security proof technique. As stated in [7], these works consider the setting where compression functions and hash functions are families indexed by a dedicated key and only focus on MAC preservation when the underlying compression function is a MAC itself, namely, it is an unforgeable FIL MAC.

Bellare and Ristenpart [8] considered several hash function constructions in the dedicated-key setting and provided a multi-property-preservation oriented treatment of them. Andreeva et al. [1] proposed a composition scheme called ROX which preserves seven security properties formalized by Rogaway and Shrimpton [29].

We also mention some of the related work after the publication of the preliminary version of this paper [18]. Hirose and Kuwakado [17] further discussed the security of a block-cipher-based hash function using the Matyas–Meyer–Oseas (MMO) compression function and the MDP composition. Chang and Nandi [13] improved the security bound on indifferentiability for chop-MD. Wide-pipe construction [22] with chop-MD is quite popular among candidates of the SHA-3 competition held by NIST. Bhattacharyya, Mandal, and Nandi [10] presented a unified framework for indifferentiability analysis of hash functions by providing a model of composition schemes called GDE (Generalized Domain Extension). Actually, GDE includes MDP, and an indifferentiability result on MDP similar to Theorem 1 in Sect. 4.2 can also be obtained with their technique. Some multi-property-preserving double-piped composition schemes were recently proposed by Yasuda [31] and by Lee and Steinberger [21]. Yasuda showed that a variant of HMAC without the key to the outer hash function is a PRF if the underlying compression function is a PRF (when keyed via IV) with an additional reasonable property [32].

Organization of the Paper In Sect. 2, we provide basic definitions of PRFs, RKA-secure PRFs, indifferentiability, and unforgeability. We also fix notational conventions in this section. In Sect. 3, we formally define the MDP construction. In Sect. 4, we analyze the security of MDP. Section 4 consists of three parts; first, we prove that MDP is indifferentiable from a random oracle, then prove that MDP gives a secure PRF under necessary assumptions, and finally we prove that MDP yields a secure MAC under a weaker-than-PRF assumption. In Sect. 5, we focus on the indifferentiability of MDP based on two specific types of compression functions: one is a DBL compression function, and the other is the Davies–Meyer compression function composed of a block cipher.

2. Preliminaries

2.1. *Notation*

We denote by $x_1 \| x_2$ the concatenation of bitstrings x_1 and x_2. We will often abbreviate $x_1 \| x_2 \| \cdots \| x_k$ simply as $x_1 x_2 \cdots x_k$.

We denote by $s \xleftarrow{\$} S$ the operation of selecting a random element from S (the uniform probability distribution over S is assumed). Let $s_1, s_2, \ldots, s_k \xleftarrow{\$} S$ mean that $s_1, s_2, \ldots, s_k$ are selected uniformly and independently from S.

For two integers n_1 and n_2 such that $n_1 \leq n_2$, let $[n_1, n_2]$ be the set of integers from n_1 to n_2.

We sometimes use the O-notation. This is not about asymptotics, but we use this notation to hide unimportant small constants which are dependent on specific machine formalisms, and whose values can be determined from the proofs.

2.2. *Definitions*

Pseudorandom Functions Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function family from $\mathcal{D}$ to $\mathcal{R}$ indexed by keys $K \in \mathcal{K}$. Usually we use $F_K(x)$ as a shorthand for $F(K, x)$. Let $\mathsf{Maps}(\mathcal{D}, \mathcal{R})$ denote the set of all functions $f : \mathcal{D} \to \mathcal{R}$. Given an adversary A^g with access to an oracle $g(\cdot)$, we define its PRF-advantage over F as

$$\mathrm{Adv}^{\mathrm{prf}}_F(A) = \Pr\left[A^{F_K} \Rightarrow 1 \,\middle|\, K \xleftarrow{\$} \mathcal{K}\right] - \Pr\left[A^{\rho} \Rightarrow 1 \,\middle|\, \rho \xleftarrow{\$} \mathsf{Maps}(\mathcal{D}, \mathcal{R})\right].$$

Informally, we say that F is a PRF when no efficient adversary A can have any significant PRF-advantage over F.

RKA-secure PRFs Related-key attacks were considered in cryptanalysis of block ciphers, and many modern block ciphers are designed against such attacks. Bellare and Kohno [6] first gave a formal definition to related-key attacks and provided a theoretical treatment. They extended the formal definition of PRFs to PRFs secure against related-key attacks (RKA-secure PRFs).

According to the definition given by Bellare and Kohno, they consider a set Φ of related-key-deriving (RKD) functions $\phi : \mathcal{K} \to \mathcal{K}$. As in the case of the plain PRFs, an adversary cannot access the given secret key K directly, but she can query the PRF with respect to other keys $\phi(K)$ by selecting an RKD function ϕ from Φ. The set Φ is a parameter of the definition, and it formalizes the varying capabilities of related-key adversaries on different situations.

In this paper, we need only very weak adversaries in terms of related-key attacks: the RKD function set Φ consists of only two functions: $\Phi = \{id, \pi\}$, where $id : \mathcal{K} \to \mathcal{K}$ is the identity function, and $\pi : \mathcal{K} \to \mathcal{K}$ is a permutation. We will refer this type of related-key attacks as the π-related-key attacks and formalize in the following way. Given an adversary $A^{g,g'}$ with access to a pair of oracles $g(\cdot)$ and $g'(\cdot)$, we define its PRF-advantage over F with respect to π-related-key attacks as

$$\mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(A) = \Pr\left[A^{F_K, F_{\pi(K)}} \Rightarrow 1 \mid K \xleftarrow{\$} \mathcal{K}\right] - \Pr\left[A^{\rho,\rho'} \Rightarrow 1 \mid \rho, \rho' \xleftarrow{\$} \mathsf{Maps}(\mathcal{D}, \mathcal{R})\right].$$

Note that this formalism is equivalent to that of Bellare and Kohno when $\Phi = \{id, \pi\}$ is used.

Again informally, we say that F is a π-RKA-secure PRF when no efficient adversary A can have any significant advantage over F. Since the π-related-key attack is the only kind of related-key attacks that we consider in this paper, sometimes we will abuse the terminology and call F simply an RKA-secure PRF.

Indifferentiability We use the indifferentiability framework [15,23] to assess the security of MDP. Consider a cryptosystem C with oracle access to an ideal primitive $\mathcal{F}$. Also consider an ideal primitive $\mathcal{H}$ and a simulator S which has oracle access to $\mathcal{H}$. $C^{\mathcal{F}}$ is supposed to be a "construction" involving $\mathcal{F}$. For example, $\mathcal{F}$ could be a FIL random oracle, and $C^{\mathcal{F}}$ then could be the MD hash function using $\mathcal{F}$ as the compression function. The goal of the simulator $S^{\mathcal{H}}$ is to mimic $\mathcal{F}$ in order to convince an adversary that $\mathcal{H}$

is C. Let A be an adversary with access to two oracles. We define the indifferentiability advantage of A against C with respect to S as

$$\mathrm{Adv}^{\mathrm{indiff}}_{C,S}(A) = \Pr\bigl[A^{C^{\mathcal{F}},\mathcal{F}} \Rightarrow 1\bigr] - \Pr\bigl[A^{\mathcal{H},S^{\mathcal{H}}} \Rightarrow 1\bigr].$$

Informally, we say that $C^{\mathcal{F}}$ is indifferentiable from $\mathcal{H}$ if there exists an efficient simulator S such that no efficient adversary A can have any significant indifferentiability advantage against C with respect to S.

Unforgeability A MAC is a family of functions $F : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$. The security of a MAC is measured via its resistance to existential forgery under an adaptive chosen-message attack. A forger A queries the oracle F_K for adaptively chosen messages and learns the corresponding tag values. It then returns a forgery (M, τ). The forger A is considered successful if $F_K(M) = \tau$ but M was not queried to F_K. The MAC-advantage of a forger A over F is

$$\mathrm{Adv}^{\mathrm{mac}}_{F}(A) = \Pr\bigl[A^{F_K} \text{ is successful} \bigm| K \xleftarrow{\$} \mathcal{K}\bigr].$$

Informally, a MAC is considered secure against existential forgery under an adaptive chosen-message attack if there is no efficient forger with any significant MAC-advantage over F.

3. The MDP Construction

Let b and c be positive integers, and let $\mathcal{B} = \{0,1\}^b$ and $\mathcal{C} = \{0,1\}^c$. Let $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$ be a compression function. Here, b is the size of the message blocks, and c is the size of the chaining variables. As is usual with popular hash functions, we assume that $c \le b$.

Let $\mathcal{B}^i$ be the set of all messages of form $M_1 M_2 \cdots M_i$, where $M_j \in \mathcal{B}$ for all $j = 1, \ldots, i$. Clearly, $\mathcal{B}^0 = \{\epsilon\}$, where ϵ means the null bitstring, the bitstring of length 0. Let us define $\mathcal{B}^* = \bigcup_{i=0}^{\infty} \mathcal{B}^i$, $\mathcal{B}^+ = \bigcup_{i=1}^{\infty} \mathcal{B}^i$, and $\mathcal{B}^{\le k} = \bigcup_{i=1}^{k} \mathcal{B}^i$.

Given $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$, we define $F^* : \mathcal{C} \times \mathcal{B}^* \to \mathcal{C}$ as follows: for $s \in \mathcal{C}$ and $M = M_1 M_2 \cdots M_k$ ($M_i \in \mathcal{B}$ for all i),

$$F^*(s, M) \stackrel{\mathrm{def}}{=} \begin{cases} s & \text{if } k = 0, \text{ i.e., } M = \epsilon, \\ F(F^*(s, M_1 M_2 \cdots M_{k-1}), M_k) & \text{otherwise.} \end{cases}$$

This is the plain Merkle–Damgård iteration of F. Now we define $F^{\circ}_{\pi} : \mathcal{C} \times \mathcal{B}^+ \to \mathcal{C}$ as follows:

$$F^{\circ}_{\pi}(s, M_1 M_2 \cdots M_k) \stackrel{\mathrm{def}}{=} F\bigl(\pi\bigl(F^*(s, M_1 \cdots M_{k-1})\bigr), M_k\bigr),$$

where π is a fixed permutation given as a parameter of the definition. We require both π and its inverse π^{-1} to be efficiently computable. Often we omit π from the notation F°_{π} and simply write F°.

In order to let MDP process messages of arbitrary lengths (or up to $2^v - 1$ for some integer v satisfying $0 < v \le b$), we have to use a padding function $\mathsf{pad} : \bigcup_{i=0}^{2^v-1} \{0,1\}^i \to$

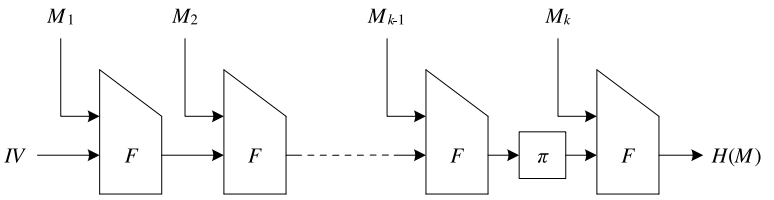

Fig. 1. The structure of MDP. $\mathsf{pad}(M) = M_1 M_2 \cdots M_k$.

$\mathcal{B}^+$ with the Merkle–Damgård (MD) strengthening: the last block of $\mathsf{pad}(M)$ encodes the ν-bit representation of the length $|M|$ of M. For example, the SHA-1's padding rule could be used.

Finally, given a compression function $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$, a padding function pad, a permutation π, and a fixed IV $IV \in \mathcal{C}$, we formally define the MDP (Merkle–Damgård with Permutation) hash function as

$$\mathsf{MDP}(M) \stackrel{\text{def}}{=} F_\pi^\circ\big(IV, \mathsf{pad}(M)\big).$$

When we want to emphasize the dependency of $\mathsf{MDP}(M)$ to F and π, we sometimes use the notation $\mathsf{MDP}[F, \pi](M)$.

Figure 1 illustrates the structure of MDP. One can consider the MDP construction as a minor variant of the MD scheme with the MD strengthening. Therefore the efficiency of MDP is exactly the same as the Strengthened MD (SMD).

More precisely, let us write the number of compression function invocations needed to compute the hash value of an ℓ-bit string as $N(\ell)$. Suppose that we use the padding function similar to the padding function of SHA-1: given a message M of length ℓ, append the bit "1" to the end of the message, followed by k zero bits, where k is the smallest nonnegative solution to the equation $\ell + 1 + k \equiv b - \nu \pmod{b}$. Then append the ν-bit representation of the number ℓ. In case of SHA-1, we have $b = 512$ and $\nu = 64$. Then for MDP (and SMD), the following holds:

$$N(\ell) = \begin{cases} \lceil \ell/b \rceil & \text{if } \ell \bmod b < b - \nu, \\ \lceil \ell/b \rceil + 1 & \text{otherwise.} \end{cases}$$

For comparison, this is slightly better than the efficiency of EMD; for EMD, the following holds for $\ell \geq b$:

$$N(\ell) = \begin{cases} \lceil \ell/b \rceil & \text{if } \ell \bmod b < b - c - \nu, \\ \lceil \ell/b \rceil + 1 & \text{otherwise.} \end{cases}$$

Concretely, if we take the parameters of SHA-1, that is, $b = 512$, $c = 160$, and $\nu = 64$, then for messages of length ℓ such that $288 \leq \ell \bmod 512 \leq 447$, EMD needs one more invocation than MDP.

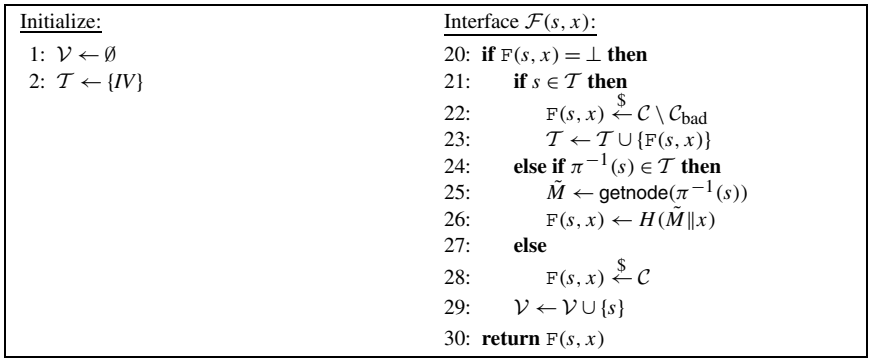

Fig. 2. Pseudocode for the simulator S_F. $\mathcal{C}_{\mathrm{bad}} = \mathcal{V} \cup \mathcal{T} \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T}) \cup \pi(\mathcal{T}) \cup P_\pi$. P_π is the set of fixed points of π. $\pi(\mathcal{T}) = \{\pi(s) \mid s \in \mathcal{T}\}$. $\pi^{-1}(\mathcal{V} \cup \mathcal{T})$ is defined similarly.

4. Security of MDP

In this section, we study the security of MDP and prove that MDP indeed meets all the security goals that we wanted. Actually, a padding function with the MD strengthening is necessary only for collision resistance. Thus, without loss of generality, we assume that the inputs to MDP is simply in $\mathcal{B}^+$ except for collision resistance.

4.1. *Collision Resistance*

Given a collision-resistant compression function F, MDP construction from F is also collision-resistant. The proof is trivial; since the structure of MDP is very similar to the MD scheme, we may follow the proof of collision resistance of the MD almost verbatim.

4.2. *Indifferentiability from Random Oracle*

We show that MDP is indifferentiable from a VIL random oracle H, when a FIL random oracle F is used as the compression function. We need a simulator S_F such that no efficient adversary can distinguish (or rather, differentiate) the pair $(\mathsf{MDP}[F,\pi], F)$ from the pair (H, S_F). We will use the simulator illustrated in Fig. 2.

S_F maintains a structure $\mathtt{F}$ where it stores selected values for previous queries. Initially $\mathtt{F}(s,x) = \perp$ for all s and x, where $\perp$ means undefined. S_F also maintains two sets $\mathcal{V}$ and $\mathcal{T}$. As more queries are inquired, new elements are added to the sets. Note that elements never leave the sets.

If we consider the labeled directed graph G whose edges are "$s \xrightarrow{x} \mathtt{F}(s,x)$" for all $\mathtt{F}(s,x) \neq \perp$, then we can see that $\mathcal{V}$ denotes the set of all vertices of G with out-degree at least one. On the other hand, $\mathcal{T}$ is then the set of all vertices that can be reached by following a path from the vertex IV. The procedure $\mathsf{getnode}(v)$ in S_F returns the labels of the edges on the path from IV to v.

In order to prove the indifferentiability of MDP, we need a few lemmas about the simulator S_F. Since $\mathtt{F}$ can also be regarded as a partial function, we can define a partial function $\mathtt{F}^*$ with $\mathtt{F}$ similarly to F^* with F; $\mathtt{F}^*(s, M\|x) = \mathtt{F}(\mathtt{F}^*(s,M), x)$ if both $s' = \mathtt{F}^*(s,M)$ and $\mathtt{F}(s',x)$ are defined.

Lemma 1. *At any time during the execution of the simulator S_F, if $s \in \mathcal{T}$, then $\mathtt{F}^*(IV, M) = s$ for some M. Conversely, if $\mathtt{F}^*(IV, M) \neq \perp$, then $\mathtt{F}^*(IV, M) \in \mathcal{T}$.*

Lemma 2. *Suppose that both $\mathtt{F}^*(IV, M)$ and $\mathtt{F}^*(IV, M')$ are defined. Then, $\mathtt{F}^*(IV, M) = \mathtt{F}^*(IV, M')$ if and only if $M = M'$.*

Lemma 3. *Suppose that both $\mathtt{F}^*(IV, M)$ and $\mathtt{F}^*(IV, M')$ are defined. Then, $\mathtt{F}^*(IV, M) \neq \pi(\mathtt{F}^*(IV, M'))$ and $\mathtt{F}^*(IV, M) \neq \pi^{-1}(\mathtt{F}^*(IV, M'))$.*

Lemmas 1 and 2 essentially say that the subgraph $\mathcal{T}$ is in fact a rooted tree with IV as the root. By keeping the tree, though it is implicit in Fig. 2, $\mathsf{getnode}(\pi^{-1}(s))$ trails the edges backwards from $\pi^{-1}(s)$ to IV and returns $\tilde{M}$ in $O(|\mathcal{T}|)$ time. Note that, because these three lemmas are about the subgraph $\mathcal{T}$ and it is altered only by the lines 21 to 23, the lines 24 to 28 do not affect the validity of the lemmas. Also, due to Lemmas 1 and 2, the lines 25 and 26 work correctly.

The proofs of the three lemmas are straightforward, and we will give only brief proof sketches. First, Lemma 1 is clear because $\mathcal{T}$ is extended in lines 21 to 23 of Fig. 2.

Next, suppose that $\mathtt{F}^*(IV, M) \neq \perp$. Let x be the tail of the message M, that is, the last block of M. Then, we may see that, among the queries which define $\mathtt{F}^*(IV, M)$, the query involving x was the last one made; suppose not and let $(\check{s}, \check{x})$ be the last query which defines $\mathtt{F}^*(IV, M)$. This means that $M = \check{M}\|\check{x}\|\hat{M}$, $\check{s} = \mathtt{F}^*(IV, \check{M})$ and $\mathtt{F}^*(IV, M) = \mathtt{F}^*(\mathtt{F}(\check{s}, \check{x}), \hat{M})$. Since $(\check{s}, \check{x})$ is the last query, $\hat{s} = \mathtt{F}(\check{s}, \check{x})$ should be already in $\mathcal{V}$ (note that since $\check{x}$ is not the tail, $\hat{M}$ is not null). But $\mathtt{F}(\check{s}, \check{x})$ should be chosen from $\mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$, while $\mathcal{V} \subset \mathcal{C}_{\mathrm{bad}}$, which is a contradiction.

Now we can give a proof sketch for Lemmas 2 and 3. Suppose that $s = \mathtt{F}^*(IV, M) = \mathtt{F}^*(IV, M') \neq \perp$. Let x (resp. x') be the tail of M (resp. M'). Without loss of generality, we may assume that the query involving x' was made later than the one involving x. Let this query be (s', x') for some s'. Then, $\mathtt{F}(s', x') = s$, but $\mathtt{F}(s', x')$ should be chosen out of $\mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$, while $s \in \mathcal{T} \subset \mathcal{C}_{\mathrm{bad}}$, since there is already a query involving x with s as the response. This contradicts the assumption and proves Lemma 2. For Lemma 3, we may use the same argument except using $\pi(\mathcal{T}) \subset \mathcal{C}_{\mathrm{bad}}$.

The basic intuition involved in the pseudocode of S_F is this: the permutation π disrupts the extension property of the MD scheme if it has only a small number of fixed points and IV is not a fixed point. Now, the best strategy of an adversary seems to be computing $F^*(IV, M)$ for various messages M (by querying the FIL oracle), until one of the following happens:

- The adversary finds two distinct messages M, M' such that $F^*(IV, M) = F^*(IV, M')$: in this case, we have $H(M\|P) = H(M'\|P)$ for any message block P if H is MDP. On the other hand, the probability of this equality is very low if H is a true random oracle.
- The adversary finds two distinct messages M, M' such that $F^*(IV, M) = \pi(F^*(IV, M'))$: in this case, we have $H(M\|P\|Q) = F(\pi(H(M'\|P)), Q)$ for any message block P and Q if H is MDP. However, the probability of this equality is very low if H is a true random oracle, because the simulator which selects the value $F(\pi(H(M'\|P)), Q)$ has information about Q, but it does not have access to the adversarial choice of P.

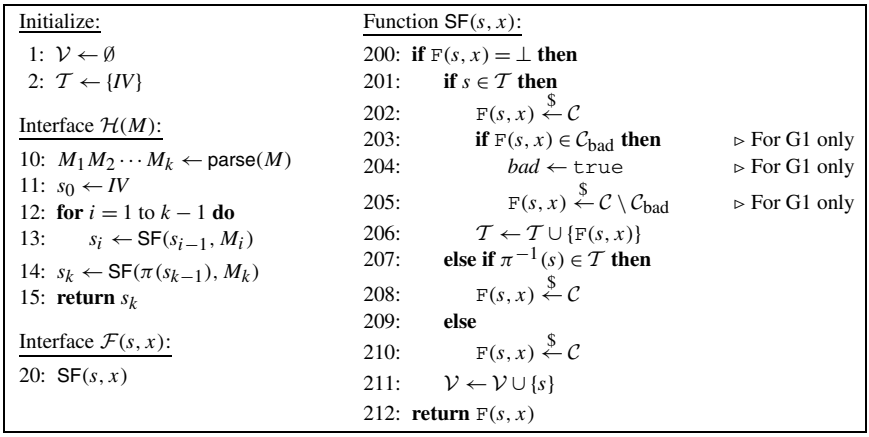

Fig. 3. The games G0 and G1. $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$ means that $M = M_1 \| M_2 \| \cdots \| M_k$ and $|M_i| = b$ for all $i = 1, \ldots, k$. The lines 203, 204, and 205 are active only in G1. $\mathcal{C}_{\mathrm{bad}} = \mathcal{V} \cup \mathcal{T} \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T}) \cup \pi(\mathcal{T}) \cup P_\pi$.

Other minor strategy is to find a message M such that $F^*(IV, M)$ is a fixed point of π or a part of a previous query to F.

The simulator S_F is designed so that the three Lemmas 1, 2, and 3 hold, which delays the above failing situations as late as possible. This is achieved by careful expansion of the tree $\mathcal{T}$ at the lines 22 and 23. Note that by the birthday attack, eventually the attacker can find the message pair M, M' such that $F^*(IV, M)$ equals $F^*(IV, M')$ or $\pi(F^*(IV, M'))$. Therefore, MDP can be indifferentiable from a random oracle only up to the birthday bound.

Now, the indifferentiability of MDP is expressed in the next theorem.

Theorem 1. *Let A be an adversary distinguishing the pairs $(\mathsf{MDP}[F, \pi], F)$ and (H, S_F), where the simulator S_F is defined in Fig. 2. Let π be a permutation on $\mathcal{C}$ and P_π be the set of its fixed points such that $IV \notin P_\pi$. Suppose that A makes at most q_{F} queries to the FIL oracle and q_{V} queries to the VIL oracle and that each VIL query has at most ℓ message blocks. Let $q = \ell q_{\mathrm{V}} + q_{\mathrm{F}}$. Then,*

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_F}(A) \leq \frac{3q^2 + (2|P_\pi| + 3)q}{2^{c+1}} + \frac{2\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 3q - |P_\pi|}.$$

The total number of queries of the simulator S_F is at most q_{F}, and the total running time of S_F is $O({q_{\mathrm{F}}}^2)$.

Proof. The proof uses the code-based game-playing technique [9]. Each game Gi exposes two interfaces $\mathcal{H}(M)$ and $\mathcal{F}(s,x)$ to the adversary A. $A^{\mathrm{G}i}$ means that A has oracle access to the interfaces in the game Gi. Without loss of generality, we can assume that A makes no repeated queries to $\mathcal{H}$ and $\mathcal{F}$.

The proof starts with the game G0 given in Fig. 3. It is clear that G0 is a faithful implementation of MDP with a FIL random oracle. Although it updates $\mathcal{V}$ and $\mathcal{T}$, these

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$:
10: $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SF}(s_{i-1}, M_i)$
14: $s_k \leftarrow \mathsf{SF}(\pi(s_{k-1}), M_k)$
15: **return** s_k

Interface $\mathcal{F}(s,x)$:
20: $\mathsf{SF}(s,x)$

Function $\mathsf{SH}(M)$:
100: **if** $\mathtt{H}(M) = \perp$ **then**
101: $\quad \mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathtt{H}(M)$

Function $\mathsf{SF}(s,x)$:
200: **if** $\mathtt{F}(s,x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(s,x)\}$
204: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
205: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
206: $\quad\quad \mathtt{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M} \| x)$
207: $\quad$ **else**
208: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C}$
209: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
210: **return** $\mathtt{F}(s,x)$

Fig. 4. The game G2.

sets do not affect the uniform and random choices of $\mathtt{F}(s,x)$ made at lines 202, 208, and 210. In effect, $\mathcal{F}$ implements a FIL random oracle by lazy sampling. Therefore,

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] = \Pr\left[A^{\mathsf{MDP}[F,\pi],F} \Rightarrow 1\right].$$

(G0 $\rightarrow$ G1). Since G0 and G1 are identical until *bad* gets `true` in G1, from Lemma 1 in [9],

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] \le \Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right].$$

$\Pr[A^{\mathrm{G1}} \text{ sets } bad]$ can be estimated as follows: the line 203 of G1 is satisfied with probability

$$\frac{|\mathcal{V} \cup \mathcal{T} \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T}) \cup \pi(\mathcal{T}) \cup P_\pi|}{|\mathcal{C}|} \le \frac{2\,|\mathcal{V} \cup \mathcal{T}| + |\mathcal{T}| + |P_\pi|}{2^c}.$$

Let $\mathcal{V}_i$ and $\mathcal{T}_i$ be the sets $\mathcal{V}$ and $\mathcal{T}$, respectively, at the line 203 in the ith invocation of SF for $i \ge 1$. Then, it is easy to see that $|\mathcal{V}_i \cup \mathcal{T}_i| \le i$ and $|\mathcal{T}_i| \le i$. SF is invoked at most $q = \ell q_{\mathrm{V}} + q_{\mathrm{F}}$ times in total. Thus,

$$\Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right] \le \sum_{i=1}^{q} \frac{2\,|\mathcal{V}_i \cup \mathcal{T}_i| + |\mathcal{T}_i| + |P_\pi|}{2^c} \le \sum_{i=1}^{q} \frac{3i + |P_\pi|}{2^c}$$
$$= \frac{3q^2 + (2\,|P_\pi| + 3)\,q}{2^{c+1}}.$$

(G1 $\rightarrow$ G2). The game G2 is presented in Fig. 4. The line 202 of G2 and the lines 202 to 205 of G1 correspond to the line 22 of the simulator S_F in Fig. 2. Therefore, Lemmas 1, 2, and 3 hold for G1 and G2. Then, the line 205 of G2 works properly;

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{T}_A \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$ for G3:
10: $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SF}(s_{i-1}, M_i)$
14: $s_k \leftarrow \mathsf{SF}(\pi(s_{k-1}), M_k)$
15: **return** s_k

Interface $\mathcal{H}(M)$ for G4:
10: $\mathsf{SH}(M)$

Interface $\mathcal{F}(s,x)$:
20: $\mathsf{SF}(s,x)$

Function $\mathsf{SH}(M)$:
100: **if** $\mathtt{H}(M) = \perp$ **then**
101: $\quad \mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathtt{H}(M)$

Function $\mathsf{SF}(s,x)$:
200: **if** $\mathtt{F}(s,x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(s,x)\}$
204: $\quad\quad$ **if** (s,x) is from $\mathcal{F}$ **then**
205: $\quad\quad\quad$ **if** $s \in \mathcal{T}_A$ **then**
206: $\quad\quad\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(s,x)\}$
207: $\quad\quad\quad$ **else**
208: $\quad\quad\quad\quad \mathit{bad} \leftarrow \mathtt{true}$
209: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
210: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
211: $\quad\quad \mathtt{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$
212: $\quad\quad$ **if** (s,x) is from $\mathcal{F}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$ **then**
213: $\quad\quad\quad \mathit{bad} \leftarrow \mathtt{true}$
214: $\quad$ **else**
215: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C}$
216: $\quad\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
217: **else if** (s,x) is from $\mathcal{F}$ **then**
218: $\quad$ **if** $s \in \mathcal{T}_A$ **then**
219: $\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(s,x)\}$
220: $\quad$ **else if** $\pi^{-1}(s) \notin \mathcal{T}_A$ **then**
221: $\quad\quad \mathit{bad} \leftarrow \mathtt{true}$
222: **return** $\mathtt{F}(s,x)$

Fig. 5. The games G3 and G4.

there exists one-to-one correspondence between $\pi^{-1}(s) \in \mathcal{T}$ and $\tilde{M}$ via $\mathtt{F}^*(IV, \tilde{M}) = \pi^{-1}(s)$.

The difference between G1 and G2 is the selection of $\mathtt{F}(s,x)$ in case $\pi^{-1}(s) \in \mathcal{T}$. In G1, $\mathtt{F}(s,x)$ is randomly chosen at the line 208. In G2, $\mathtt{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$, where $\mathtt{F}^*(IV, \tilde{M}) = \pi^{-1}(s)$, and the actual random selection is deferred by the subroutine SH. SH maintains a structure H, which is initially $\perp$ for every M. Due to Lemmas 1 and 2, there is a one-to-one correspondence between (s,x) and $\tilde{M}\|x$. Therefore, in essence, G2 simply selects $\mathtt{F}(s,x)$ randomly. Thus,

$$\Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right].$$

(G2 $\rightarrow$ G3). The game G3 is given in Fig. 5. In G3, a new set $\mathcal{T}_A$ is introduced, which is initially $\{IV\}$. Some operations on $\mathcal{T}_A$ are also added to SF: from 204 to 208, from 212 to 213, and from 217 to 221. However, these differences between G2 and G3 do not affect the outputs by $\mathcal{H}$ and $\mathcal{F}$. Actually, the new lines introduced in G3 only change $\mathcal{T}_A$ and the variable *bad*, and the outputs by $\mathcal{H}$ and $\mathcal{F}$ are chosen independently of them. Thus,

$$\Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right].$$

(G3 $\rightarrow$ G4). The game G4 is also given in Fig. 5. The difference between G3 and G4 is the implementation of $\mathcal{H}(M)$. In G3, $\mathcal{H}(M)$ is implemented as the MDP iteration, while $\mathcal{H}(M)$ is just a random oracle in G4.

We will first confirm that the return values of $\mathcal{H}$ are always determined by SH in G3. From this viewpoint, there is no difference between G3 and G4.

Suppose that A asks M to $\mathcal{H}$ in G3. Then, $\mathcal{H}$ invokes SF with (s_{i-1}, M_i) for $1 \le i \le k-1$ and $(\pi(s_{k-1}), M_k)$. Notice that each of them may be asked before. In any case, however, SF receives $(s_0, M_1), \ldots, (s_{k-2}, M_{k-1}), (\pi(s_{k-1}), M_k)$ in this order.

Suppose that SF receives (s_{i-1}, M_i) at the α_ith invocation for $1 \le i \le k-1$ and $(\pi(s_{k-1}), M_k)$ at the α_kth invocation for the first time in G3. We will show that $\alpha_1 < \alpha_2 < \cdots < \alpha_k$. Since $s_0 = IV \in \mathcal{T}$, s_1 is selected at the line 202 and added to $\mathcal{T}$ at the line 203. Thus, $\alpha_1 < \alpha_2$ since $s_1 \notin \mathcal{V}_{\alpha_1}$. Similarly, $\alpha_2 < \alpha_3 < \cdots < \alpha_{k-1}$ and $s_i \in \mathcal{T}$ for $2 \le i \le k-1$. Moreover, $\alpha_{k-1} < \alpha_k$ since $s_{k-1} \notin \pi^{-1}(\mathcal{V}_{\alpha_{k-1}})$. Thus, at the α_kth invocation with $(\pi(s_{k-1}), M_k)$, SF invokes SH with M. Notice that $\mathtt{H}(M)$ is undefined before this invocation.

There is still a difference between G3 and G4. $\mathcal{H}$ calls SF in G3, while $\mathcal{H}$ does not in G4. In G4, since SF is invoked only by $\mathcal{F}$ and A makes no repeated queries, the lines from 217 to 221 are never executed. Moreover, at the time of each invocation of SF, $\mathcal{T} = \mathcal{T}_A$. Thus, *bad* never gets `true` in G4. In G3, on the other hand, *bad* may get `true`. A may accidentally find an intermediate part of the paths made by $\mathcal{H}$ without starting from IV. Thus,

$$\Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] \le \Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right].$$

Now, we will evaluate $\Pr[A^{\mathrm{G3}} \text{ sets } bad]$. If *bad* gets `true`

- at 208, then $s \in \mathcal{T}$ and $s \notin \mathcal{T}_A$,
- at 213, then $\pi^{-1}(s) \in \mathcal{T}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$,
- at 221, then $\mathtt{F}(s, x)$ has already been set by $\mathcal{H}$, $s \notin \mathcal{T}_A$, and $\pi^{-1}(s) \notin \mathcal{T}_A$.

Even in the last case, $s \in \mathcal{T}$ or $\pi^{-1}(s) \in \mathcal{T}$. Thus, in any case, if *bad* gets `true`, then it implies that A successfully asks the value of s' or $\pi(s')$ to $\mathcal{F}$ for some $s' \in \mathcal{T}$ without having access to s' using $\mathcal{F}$ from IV. Since the number of elements in $\mathcal{T}$ fixed by $\mathcal{H}$ is at most ℓq_{V} and they are chosen from $\mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$,

$$\Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right] \le \frac{2\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 3q - |P_\pi|}.$$

(G4 $\rightarrow$ G5). The game G5 is presented in Fig. 6. It is a faithful implementation of a true random oracle and the simulator S_F. Thus,

$$\Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right] = \Pr\left[A^{H, S_F} \Rightarrow 1\right].$$

G5 is obtained from G4 by removing $\mathcal{T}_A$ in the initialization, and the operations related to $\mathcal{T}_A$ in SF: lines 204 to 208, 212 to 213, and 217 to 221. These modifications do not affect the outputs by $\mathcal{H}$ and $\mathcal{F}$. Thus,

$$\Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right].$$

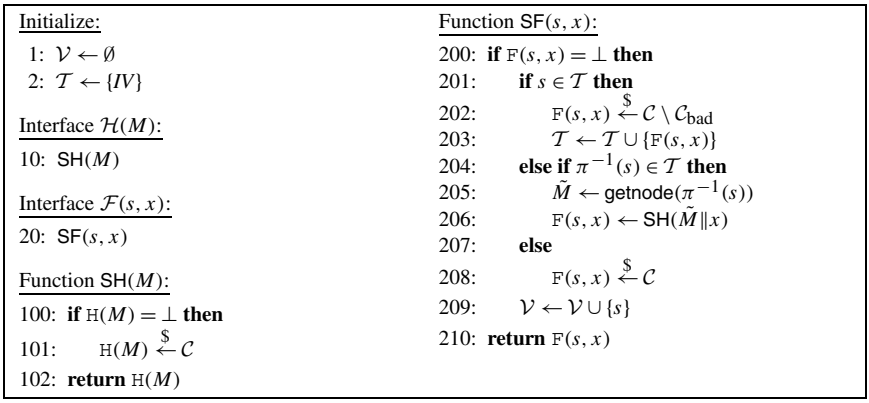

Fig. 6. The game G5.

Combining all of the above, we get

$$\begin{aligned}\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_F}(A) &\le \Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right] + \Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right] \\ &\le \frac{3q^2 + (2|P_\pi| + 3)q}{2^{c+1}} + \frac{2\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 3q - |P_\pi|}.\end{aligned}$$

□

4.3. *MDP Yields a Secure PRF*

In this section, we show that when the compression function F is a PRF secure against a π-related-key attack, then MDP yields a secure PRF. This construction could be used as an alternative to HMAC or NMAC.

In order to use MDP as a PRF, we need to provide a keying strategy to MDP. We may consider at least two straightforward such approaches.

- Keyed-MDP: We may use a secret key $K \xleftarrow{\$} \mathcal{C}$ instead of the fixed IV and define a MAC scheme out of MDP by $\mathsf{KMDP}_K(M) = F^\circ(K, \mathsf{pad}(M))$.
- Prefix-MDP: Given a message M and a key $K \xleftarrow{\$} \mathcal{B}$, we define $\mathsf{PMDP}_K(M) = \mathsf{MDP}(K \| M)$, i.e., the secret prefix construction. Note that $\mathsf{PMDP}_K(M) = \mathsf{KMDP}_{F(IV,K)}(M)$. Although less efficient than Keyed-MDP, this has a benefit that it may use the underlying hash function as a black-box.

We may consider Keyed-MDP as analogous to NMAC, and Prefix-MDP as analogous to HMAC.

Remark 1. If $\mathsf{KMDP}_K(M)$ were a secure PRF whenever F is a secure PRF, then we may say that MDP preserves the PRF property in the sense of Bellare and Ristenpart [7]. Unfortunately this is not the case; if, for example, F satisfies $F_K(x) = F_{\pi(K)}(x)$ for any K and x, then the MDP construction reduces to the plain Merkle–Damgård scheme, which is vulnerable to the extension attack.

4.3.1. *Related-Key Multioracles*

In order to prove the security of the two MAC schemes, first we need to introduce the notion of multioracle distinguishers. This was first given in [5] in order to prove that, if the MD scheme is keyed via IV, then the resulting iterated construction is a PRF with respect to prefix-free adversaries. What we actually need is not this notion itself, but an extension of it, which we call the related-key multioracle distinguisher.

Given a π-RKA-secure PRF F, consider the problem of distinguishing a $2m$-tuple of instances of F from a $2m$-tuple of independent random functions. For the $2m$-tuple of F, we choose m of the keys $K_1, \ldots, K_m$ randomly and independently and use $\pi(K_1), \ldots, \pi(K_m)$ as the other m keys. That is, we would like to distinguish the distribution of the following $2m$-tuple of functions:

$$(F_{K_1}, F_{\pi(K_1)}, \ldots, F_{K_m}, F_{\pi(K_m)})$$

from that of a $2m$-tuple of independent random functions.

We define the advantage of a distinguisher A with access to $2m$ oracles g_1, g'_1, g_2, g'_2, $\ldots$, g_m, g'_m as follows:

$$\begin{aligned}\mathrm{Adv}^{m\text{-prf-rka}}_{\pi,F}(A) = \Pr\bigl[A^{F_{K_1},F_{\pi(K_1)},\ldots,F_{K_m},F_{\pi(K_m)}} \Rightarrow 1 \bigm| K_1, \ldots, K_m \xleftarrow{\$} \mathcal{C}\bigr] \\ - \Pr\bigl[A^{\rho_1,\rho'_1,\ldots,\rho_m,\rho'_m} \Rightarrow 1 \bigm| \rho_1, \rho'_1, \ldots, \rho_m, \rho'_m \xleftarrow{\$} \mathsf{Maps}(\mathcal{B},\mathcal{C})\bigr].\end{aligned}$$

Lemma 4 (Related-Key Multioracle Lemma). *Let A be a distinguisher with access to $2m$ oracles $g_1, g'_1, \ldots, g_m, g'_m$ as above. Suppose that A has time-complexity at most t and makes at most q queries. Then, we can construct an adversary $B^{g,g'}$ attacking the π-RKA-security of F such that*

$$\mathrm{Adv}^{m\text{-prf-rka}}_{\pi,F}(A) = m \cdot \mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(B).$$

B makes at most q queries, and the running time of B is bounded by $t + O(q \cdot \mathrm{Time}(F) + q(b \log q + c) + mc)$, where $\mathrm{Time}(F)$ is the time required to compute F.

Proof. Given such a distinguisher A, for $i = 0, \ldots, m$, let us define

$$P_i = \Pr\bigl[A^{F_{K_1},F_{\pi(K_1)},\ldots,F_{K_i},F_{\pi(K_i)},\rho_{i+1},\rho'_{i+1},\ldots,\rho_m,\rho'_m} \Rightarrow 1\bigr],$$

where $K_1, \ldots, K_i$ are randomly and independently chosen from $\mathcal{C}$, and ρ_{i+1}, ρ'_{i+1}, $\ldots$, ρ_m, ρ'_m are randomly and independently chosen from $\mathsf{Maps}(\mathcal{B},\mathcal{C})$. Note that

$$\mathrm{Adv}^{m\text{-prf-rka}}_{\pi,F}(A) = P_m - P_0.$$

Using the distinguisher A, let us define an adversary $B[i]$ for $i = 1, \ldots, m$, attacking the RKA-secure PRF F as follows: $B[i]$ is given two oracles g and g'. $B[i]$ then prepares the following sequence of $2m$ functions to feed A:

$$F_{K_1}, F_{\pi(K_1)}, \ldots, F_{K_{i-1}}, F_{\pi(K_{i-1})}, g, g', \rho_{i+1}, \rho'_{i+1}, \ldots, \rho_m, \rho'_m,$$

where $K_1, \ldots, K_{i-1}$ are randomly and independently chosen from $\mathcal{C}$, and $\rho_{i+1}, \rho'_{i+1}, \ldots, \rho_m, \rho'_m$ are randomly and independently chosen from $\mathsf{Maps}(\mathcal{B}, \mathcal{C})$. Note that $B[i]$ "implements" these independent random functions ρ_j, ρ'_j by lazy sampling: initially, $B[i]$ sets all values of ρ_j, ρ'_j as undefined, and when A queries one of the random functions, for example, $\rho_j(x)$, then $B[i]$ first checks whether $\rho_j(x) \neq \perp$ and if so, simply returns $\rho_j(x)$, and if $\rho_j(x) = \perp$, then defines $\rho_j(x)$ randomly and returns $\rho_j(x)$.

$B[i]$ runs A in its simulated environment and, whenever A makes a query, answers the query using the above $2m$ functions. Then it is clear that

$$\Pr\left[B[i]^{F_K, F_{\pi(K)}} \Rightarrow 1 \;\middle|\; K \xleftarrow{\$} \mathcal{C}\right] = P_i$$

and

$$\Pr\left[B[i]^{\rho,\rho'} \Rightarrow 1 \;\middle|\; \rho, \rho' \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}, \mathcal{C})\right] = P_{i-1}.$$

Finally, let us define another adversary B attacking F, by combining all $B[i]$ as follows: when B runs, it first chooses $i \xleftarrow{\$} [1, m]$ and then behaves identically to $B[i]$. Then,

$$\Pr\left[B^{F_K, F_{\pi(K)}} \Rightarrow 1 \;\middle|\; K \xleftarrow{\$} \mathcal{C}\right] = \frac{1}{m} \sum_{i=1}^{m} P_i$$

and

$$\Pr\left[B^{\rho,\rho'} \Rightarrow 1 \;\middle|\; \rho, \rho' \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}, \mathcal{C})\right] = \frac{1}{m} \sum_{i=0}^{m-1} P_i.$$

Therefore,

$$\mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(B) = \frac{P_m - P_0}{m}.$$

We can see that B has to query q times at the worst case. The running time is that of A plus the time to pick the extra keys, which is $O(mc)$, and to record and respond to queries by A, which is $O(q \cdot \mathrm{Time}(F) + q(b \log q + c))$. □

4.3.2. *Security of Keyed-MDP*

Now that we have Lemma 4, we prove the following lemma which connects the PRF-security of the Keyed-MDP with the related-key multioracles:

Lemma 5 (Reduction to the Related-Key Multioracle). *Let A be a PRF-adversary against* KMDP. *Suppose that A has time-complexity at most t and makes at most q queries and that each query has at most ℓ message blocks. Then, we can construct a related-key multioracle distinguisher B with access to $2q$ oracles such that*

$$\mathrm{Adv}^{\text{prf}}_{\mathsf{KMDP}}(A) = \ell \cdot \mathrm{Adv}^{q\text{-prf-rka}}_{\pi,F}(B).$$

B makes at most q queries, and the running time of B is bounded by $t + O(\ell q(\mathrm{Time}(F) + b \log q) + qc)$.

Proof. For $i \in [0, \ell]$ and two functions $\alpha : \mathcal{B}^{\leq i} \to \mathcal{C}$ and $\beta : \mathcal{B}^i \to \mathcal{C}$, we define $I_i^{\alpha,\beta} : \mathcal{B}^{\leq \ell} \to \mathcal{C}$ as follows:

$$I_i^{\alpha,\beta}(M_1 M_2 \cdots M_k) = \begin{cases} \alpha(M_1 \cdots M_k) & \text{if } k \leq i, \\ F^{\circ}(\beta(M_1 \cdots M_i), M_{i+1} \cdots M_k) & \text{if } k > i. \end{cases}$$

Define P_i as

$$P_i = \Pr\left[A^{I_i^{\alpha,\beta}} \Rightarrow 1 \,\middle|\, \alpha \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^{\leq i}, \mathcal{C}),\ \beta \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^i, \mathcal{C})\right].$$

Then

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathsf{KMDP}}(A) = P_0 - P_\ell.$$

Notice that α and β are simply random elements in $\mathcal{C}$ if $i = 0$.

Using A as a subroutine, we construct a related-key multioracle distinguisher $B^{g_1, g_1', \ldots, g_q, g_q'}$ taking $2q$ oracles. For $i \in [1, \ell]$, we first define a distinguisher $B[i]^{g_1, g_1', \ldots, g_q, g_q'}$.

$B[i]$ first picks a random function $\rho \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^{\leq i-1}, \mathcal{C})$. (Again, this means that $B[i]$ implements ρ via lazy sampling.) Then $B[i]$ runs A in its simulated environment. $B[i]$ has to answer the q queries of A appropriately. In order to do that, $B[i]$ maintains a counter *idx*, which is initially set to 0. When $B[i]$ is given the jth query $M^j = M_1^j M_2^j \cdots M_k^j$ of A, $B[i]$ answers it as follows:

- If $k < i$, then return $\rho(M_1^j \cdots M_k^j)$.
- If $k = i$, then return $g'_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j)$.
- If $k > i$, then return $F^{\circ}(g_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j), M_{i+1}^j \cdots M_k^j)$.

In the above, $\mathsf{idx}(M_1^j \cdots M_{i-1}^j)$ is a unique integer in $[1, q]$ which depends on the query $M_1^j \cdots M_{i-1}^j$. It can be defined using the counter *idx*; if there was a previous query M^p $(p < j)$ such that $M_1^p \cdots M_{i-1}^p = M_1^j \cdots M_{i-1}^j$, then define $\mathsf{idx}(M_1^j \cdots M_{i-1}^j) = \mathsf{idx}(M_1^p \cdots M_{i-1}^p)$, and otherwise increase *idx* by 1 and define $\mathsf{idx}(M_1^j \cdots M_{i-1}^j) = idx$.

Now, suppose that $B[i]$ is given oracles $F_{K_1}, F_{\pi(K_1)}, \ldots, F_{K_q}, F_{\pi(K_q)}$ with $K_1, \ldots, K_q \xleftarrow{\$} \mathcal{C}$. Then, when A makes the jth query $M^j = M_1^j M_2^j \cdots M_k^j$, the response A will get from $B[i]$ will be:

- If $k < i$, then $\rho(M_1^j \cdots M_k^j)$.
- If $k = i$, then $F(\pi(K_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}), M_i^j)$.
- If $k > i$, then $F^{\circ}(F(K_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}, M_i^j), M_{i+1}^j \cdots M_k^j)$, which is equal to $F^{\circ}(K_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}, M_i^j M_{i+1}^j \cdots M_k^j)$.

Since $K_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}$ is a random function of $M_1^j \cdots M_{i-1}^j$, the view that A sees is identical to $I_{i-1}^{\alpha,\beta}$ with $\alpha \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^{\leq i-1}, \mathcal{C})$, $\beta \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^{i-1}, \mathcal{C})$. Therefore,

$$\Pr\left[B[i]^{F_{K_1}, F_{\pi(K_1)}, \ldots, F_{K_q}, F_{\pi(K_q)}} \Rightarrow 1\right] = P_{i-1}.$$

Next, suppose that $B[i]$ is given $\rho_1, \rho_1', \ldots, \rho_q, \rho_q' \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}, \mathcal{C})$. Then, the response A will get from $B[i]$ will be:

- If $k < i$, then $\rho(M_1^j \cdots M_k^j)$.
- If $k = i$, then $\rho'_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j)$.
- If $k > i$, then $F^{\circ}(\rho_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j), M_{i+1}^j \cdots M_k^j)$.

Since $\rho_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j)$ and $\rho'_{\mathsf{idx}(M_1^j \cdots M_{i-1}^j)}(M_i^j)$ are independent random functions of $M_1^j \cdots M_{i-1}^j M_i^j$, the view that A sees is identical to $I_i^{\alpha,\beta}$ with $\alpha \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^{\leq i}, \mathcal{C})$, $\beta \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}^i, \mathcal{C})$. Therefore,

$$\Pr\left[B[i]^{\rho_1, \rho_1', \ldots, \rho_q, \rho_q'} \Rightarrow 1\right] = P_i.$$

Finally, let us define the distinguisher B, by combining all $B[i]$ as follows: when B runs, it first chooses $i \xleftarrow{\$} [1, \ell]$, then behaves identically to $B[i]$. Then

$$\begin{aligned}
\mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(B) &= \Pr\left[B^{F_{K_1}, F_{\pi(K_1)}, \ldots, F_{K_q}, F_{\pi(K_q)}} \Rightarrow 1 \,\middle|\, K_1, \ldots, K_q \xleftarrow{\$} \mathcal{C}\right] \\
&\quad - \Pr\left[B^{\rho_1, \rho_1', \ldots, \rho_q, \rho_q'} \Rightarrow 1 \,\middle|\, \rho_1, \rho_1', \ldots, \rho_q, \rho_q' \xleftarrow{\$} \mathsf{Maps}(\mathcal{B}, \mathcal{C})\right] \\
&= \frac{1}{\ell} \sum_{i=0}^{\ell-1} P_i - \frac{1}{\ell} \sum_{i=1}^{\ell} P_i = \frac{P_0 - P_\ell}{\ell}.
\end{aligned}$$

□

Combining Lemmas 4 and 5, we obtain the following theorem:

Theorem 2 (PRF-Security of Keyed-MDP). *Let A be a PRF-adversary against* KMDP. *Suppose that A has time-complexity at most t, makes at most q queries, and each query has at most ℓ message blocks. Then, we can construct an adversary B against F such that*

$$\mathrm{Adv}^{\text{prf}}_{\mathsf{KMDP}}(A) = \ell q \cdot \mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(B).$$

B makes at most q queries, and the running time of B is bounded by $t + O(\ell q(\mathrm{Time}(F) + b \log q) + qc)$.

4.3.3. *Security of Prefix-MDP*

We prove the security of the Prefix-MDP scheme by lifting the security proof for the Keyed-MDP. Recall that

$$\mathsf{PMDP}_K(M) = \mathsf{MDP}(K \| M) = \mathsf{KMDP}_{F(IV,K)}(M).$$

Hence, here we have to regard $F(s, x)$ as a function family indexed by the data input x. We express this formally by defining a dual function family $\bar{F} : \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ of F:

$$\bar{F}(K, x) \stackrel{\text{def}}{=} F(x, K).$$

In order to prove the security of the Prefix-MDP, in addition to the previous assumption that F is a π-RKA-secure PRF, we also need to assume that F is a PRF when keyed by its data input, i.e., $\bar{F}$ is a PRF. Then we have:

Lemma 6. *Let A be a PRF-adversary against* PMDP *that has time-complexity at most t. Then, we can construct a PRF-adversary $B_{\bar{F}}$ against the dual $\bar{F}$ such that*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PMDP}}(A) = \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{KMDP}}(A) + \mathrm{Adv}^{\mathrm{prf}}_{\bar{F}}(B_{\bar{F}}).$$

$B_{\bar{F}}$ has time-complexity at most t and makes only 1 *query.*

Proof. Using A, $B_{\bar{F}}$ can be constructed as follows. $B_{\bar{F}}$ is given an oracle $g : \mathcal{C} \to \mathcal{C}$. At the beginning, $B_{\bar{F}}$ picks a random key $K' \in \mathcal{C}$ by $K' \leftarrow g(IV)$. Then, $B_{\bar{F}}$ runs A in its simulated environment. If A makes a query M, then $B_{\bar{F}}$ answers it by $\mathsf{KMDP}_{K'}(M)$. Clearly,

$$\Pr\left[B_{\bar{F}}{}^{\bar{F}_K} \Rightarrow 1 \,\middle|\, K \stackrel{\$}{\leftarrow} \mathcal{B}\right] = \Pr\left[A^{\mathsf{PMDP}_K} \Rightarrow 1 \,\middle|\, K \stackrel{\$}{\leftarrow} \mathcal{B}\right],$$

and

$$\Pr\left[B_{\bar{F}}{}^{\rho} \Rightarrow 1 \,\middle|\, \rho \stackrel{\$}{\leftarrow} \mathsf{Maps}(\mathcal{C}, \mathcal{C})\right] = \Pr\left[A^{\mathsf{KMDP}_{K'}} \Rightarrow 1 \,\middle|\, K' \stackrel{\$}{\leftarrow} \mathcal{C}\right].$$

Therefore,

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{prf}}_{\bar{F}}(B_{\bar{F}}) &= \Pr\left[B_{\bar{F}}{}^{\bar{F}_K} \Rightarrow 1 \,\middle|\, K \stackrel{\$}{\leftarrow} \mathcal{B}\right] - \Pr\left[B_{\bar{F}}{}^{\rho} \Rightarrow 1 \,\middle|\, \rho \stackrel{\$}{\leftarrow} \mathsf{Maps}(\mathcal{C}, \mathcal{C})\right] \\
&= \Pr\left[A^{\mathsf{PMDP}_K} \Rightarrow 1 \,\middle|\, K \stackrel{\$}{\leftarrow} \mathcal{B}\right] - \Pr\left[A^{\mathsf{KMDP}_{K'}} \Rightarrow 1 \,\middle|\, K' \stackrel{\$}{\leftarrow} \mathcal{C}\right] \\
&= \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PMDP}}(A) - \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{KMDP}}(A).
\end{aligned}$$

□

Theorem 3 (PRF-Security of Prefix-MDP). *Let A be a PRF-adversary against* PMDP. *Suppose that A has time-complexity at most t, makes at most q queries, and each query has at most ℓ message blocks. Then, we can construct an adversary B_F against F and an adversary $B_{\bar{F}}$ against the dual $\bar{F}$ such that*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PMDP}}(A) = \ell q \cdot \mathrm{Adv}^{\text{prf-rka}}_{\pi,F}(B_F) + \mathrm{Adv}^{\mathrm{prf}}_{\bar{F}}(B_{\bar{F}}).$$

B_F has time-complexity at most $t + O(\ell q(\mathrm{Time}(F) + b \log q) + qc)$ and makes at most q queries. $B_{\bar{F}}$ has time-complexity at most t and makes only 1 query.

Remark 2. Even if F is a secure PRF, it could be vulnerable to a π-related-key attack. For example, Contini and Yin [14] exhibited a related-key distinguishing attack on the keyed MD5 compression function using pseudo-collisions of MD5 [11]. This attack shows that the keyed MD5 compression function is not a good π-RKA-secure PRF when π is bitwise addition of a nonzero constant.

Remark 3. Kim et al. [20], and also Contini and Yin [14], showed how to construct various attacks on HMAC and NMAC using weakness of keyed compression functions like MD4. The same attacks will work against PMDP under the same keyed compression functions.

4.4. *Unforgeability Preservation*

We may use MDP as a MAC under a different keying strategy from the above sections. Now, we consider MDP in the dedicated-key setting, where a compression function is a MAC $F : \mathcal{K} \times \mathcal{C} \times \mathcal{B} \to \mathcal{C}$ with a dedicated key input.

Theorem 4. *Let π be a permutation on $\mathcal{C}$, and P_π be the set of its fixed points such that $IV \notin P_\pi$. Let A be a forger of $\mathsf{MDP}[F, \pi]$. Suppose that A has time-complexity at most t and the total number of message blocks of its queries and forgery is at most μ. Then, we can construct a forger B of the FIL MAC F such that*

$$\mathrm{Adv}^{\mathrm{mac}}_{\mathsf{MDP}[F,\pi]}(A) \le \left(\frac{3}{2}\mu^2 + \left(|P_\pi| + \frac{3}{2}\right)\mu + 1\right)\mathrm{Adv}^{\mathrm{mac}}_{F}(B).$$

B has time-complexity at most $t + O(\mu(b + c))$ and makes at most μ queries.

Proof. A forger B against the FIL MAC F has a MAC oracle F_K. Invoking the oracle, it can easily simulate the MAC oracle $\mathsf{MDP}[F_K, \pi]$ required by a forger A. Under this simulation environment, we make use of the proof technique of [24]. Let (z_i, y_i) denote the ith MAC query made by B and the corresponding answer from F_K.

In the simulation process of $\mathsf{MDP}[F_K, \pi]$, B takes the following strategy: stop the simulation at some MAC query z_i just before asking it to F_K, and return a forgery (z_i, τ') for F_K. So the forger B is characterized by the time i when it stops and the way producing a forgery. Our security reduction requires to find a set of strategies $\mathcal{S}$ such that whenever A is successful, there is at least one strategy $s \in \mathcal{S}$ with which B is successful. If B simply picks its strategy uniformly at random from $\mathcal{S}$, then

$$\mathrm{Adv}^{\mathrm{mac}}_{F}(B) = \frac{\mathrm{Adv}^{\mathrm{mac}}_{\mathsf{MDP}[F,\pi]}(A)}{|\mathcal{S}|}.$$

The naive strategy s_{na} of B is to return $(\mathbf{z}, \tau)$ as a forgery, where (M, τ) is a forgery of A, and $\mathbf{z}$ is the input to the last F_K in the computation of $\mathsf{MDP}[F_K, \pi](M)$. B is successful with s_{na} whenever A is successful and $\mathbf{z}$ is new. So, we need a set of strategies

such that whenever $\mathbf{z}$ is not new, there exists at least one strategy s in it with which B is successful. We consider the following sets of deterministic strategies:

- $\mathcal{S}_y = \{s_{i,y} \mid i \in [1,\mu]\}$. $s_{i,y}$ is a strategy of stopping at the query z_i and returning (z_i, y). $|\mathcal{S}_y| \leq \mu$.
- $\mathcal{S}_{\mathrm{col}} = \{s_{\mathrm{col},i,j} \mid i,j \in [1,\mu], i > j\}$. $s_{\mathrm{col},i,j}$ is a strategy of stopping at the query z_i and returning (z_i, y_j). $|\mathcal{S}_{\mathrm{col}}| \leq \mu(\mu-1)/2$.
- $\mathcal{S}_{\pi\text{-col}} = \{s_{\pi\text{-col},i,j} \mid i,j \in [1,\mu], i > j\}$. $s_{\pi\text{-col},i,j}$ is a strategy of stopping at the query z_i and returning $(z_i, \pi(y_j))$. $|\mathcal{S}_{\pi\text{-col}}| \leq \mu(\mu-1)/2$.
- $\mathcal{S}_{\pi^{-1}\text{-col}} = \{s_{\pi^{-1}\text{-col},i,j} \mid i,j \in [1,\mu], i > j\}$. $s_{\pi^{-1}\text{-col},i,j}$ is a strategy of stopping at the query z_i and returning $(z_i, \pi^{-1}(y_j))$. $|\mathcal{S}_{\pi^{-1}\text{-col}}| \leq \mu(\mu-1)/2$.

Let $\mathcal{S}' = \mathcal{S}_{\mathrm{col}} \cup \mathcal{S}_{\pi\text{-col}} \cup \mathcal{S}_{\pi^{-1}\text{-col}} \cup \mathcal{S}_{IV} \cup \mathcal{S}_{\pi(IV)} \cup \mathcal{S}_{\pi^{-1}(IV)} \cup \bigcup_{y \in P_\pi} \mathcal{S}_y$. Then,

$$\left|\mathcal{S}' \cup \{s_{\mathrm{na}}\}\right| \leq \frac{3}{2}\mu^2 + \left(|P_\pi| + \frac{3}{2}\right)\mu + 1.$$

We will confirm that there exists at least one strategy $s \in \mathcal{S}'$ with which B is successful whenever $\mathbf{z}$ is not new.

Let $\tilde{z}_1, \ldots, \tilde{z}_t$ denote the sequence of queries to F_K resulting from the forgery message M of A. Note that M is new. Since $\tilde{z}_t = \mathbf{z}$ is not new, $\tilde{z}_t$ must be an earlier query to F_K, resulting from some query M' to $\mathsf{MDP}[F_K, \pi]$. Let $\tilde{z}'_1, \ldots, \tilde{z}'_{t'}$ denote the sequence of queries to F_K in the computation of $\mathsf{MDP}[F_K, \pi](M')$. Then, $\tilde{z}_t = \tilde{z}'_i$ for some $i \in [1, t']$. We consider two cases, $t = 1$ and $t > 1$.

Suppose that $t = 1$. Then, $t' > 1$ since M is new. It is impossible to have $\tilde{z}_1 = \tilde{z}'_1$ since $\tilde{z}_1 = \pi(IV)\|v$ and $\tilde{z}'_1 = IV\|v'$ for some $v, v' \in \mathcal{B}$, and $\pi(IV) \neq IV$. So $\tilde{z}_1 = \tilde{z}'_i$ for some $i \in [2, t']$. If $\tilde{z}_1 = \tilde{z}'_{t'}$, then $\pi(IV) = \pi(F_K(\tilde{z}'_{t'-1}))$, and so $F_K(\tilde{z}'_{t'-1}) = IV$. Otherwise, we have $\pi(IV) = F_K(\tilde{z}'_{i-1})$.

Suppose that $t > 1$. If $t' = 1$, then $\tilde{z}_t = \tilde{z}'_1$ and $\pi(F_K(\tilde{z}_{t-1})) = \pi(IV)$. Thus, $F_K(\tilde{z}_{t-1}) = IV$. Suppose that $t' > 1$. If $\tilde{z}_t = \tilde{z}'_1$, then we have $\pi(F_K(\tilde{z}_{t-1})) = IV$. Thus, $F_K(\tilde{z}_{t-1}) = \pi^{-1}(IV)$. If $\tilde{z}_t = \tilde{z}'_i$ for some $i \in [2, t'-1]$, then we have $\pi(F_K(\tilde{z}_{t-1})) = F_K(\tilde{z}'_{i-1})$. If $F_K(\tilde{z}_{t-1}) \in P_\pi$, then both $\tilde{z}_{t-1}$ and $\tilde{z}'_{i-1}$ are preimages of a fixed point of π. Otherwise, $\tilde{z}_{t-1} \neq \tilde{z}'_{i-1}$, and it implies a kind of collision of F_K with respect to π. Let z and z' be the earliest occurrences of $\tilde{z}_{t-1}$ and $\tilde{z}'_{i-1}$ in the queries made by B to F_K, respectively. Let y and y' be the corresponding replies. If z' precedes z, then stopping at z and returning $(z, \pi^{-1}(y'))$ is a successful strategy. Otherwise, stopping at z' and returning $(z', \pi(y))$ is a successful strategy. If $\tilde{z}_t = \tilde{z}'_{t'}$, then we have $M \neq M'$ satisfying $\mathsf{MDP}[F_K, \pi](M) = \mathsf{MDP}[F_K, \pi](M')$. Without loss of generality, we assume that $t' \geq t$. Then, either there exists an index $j \in [1, t-1]$ such that $\tilde{z}_{t-j} \neq \tilde{z}'_{t'-j}$ and $\tilde{z}_{t-j+1} = \tilde{z}'_{t'-j+1}$, or $\tilde{z}_1 = \tilde{z}'_{t'-t+1}$. The former implies that a nontrivial collision of F_K occurs since $F_K(\tilde{z}_{t-j}) = F_K(\tilde{z}'_{t'-j})$, and the latter implies that $F_K(\tilde{z}_{t'-t}) = IV$ with $t' - t \geq 1$. □

Note that the security loss of roughly μ^2 in Theorem 4 is unavoidable for iterative constructions of this nature [24].

Remark 4. If $IV \in P_\pi$, then we can show that MDP does not preserve unforgeability. The following example is very similar in structure to the result of An and Bellare [2] for the CBC MAC construction.

Assume that we have a secure FIL MAC $G : \mathcal{K} \times \mathcal{C}^2 \times \mathcal{B} \to \mathcal{C}$. Using G, we define another FIL MAC $F : \mathcal{K} \times \mathcal{C}^2 \times \mathcal{B} \to \mathcal{C}^2$ as follows:

$$F_K\big(x\|y\|M\big) \stackrel{\text{def}}{=} \pi^{-1}\big(G_K(x\|y\|M)\|x\big),$$

where $x, y \in \mathcal{C}$ and $M \in \mathcal{B}$.

It is easy to see that F is at least as secure as G. But the composition of F with MDP is completely insecure; an adversary can succeed in forgery with probability 1 after only a single query.

The attack can be described as follows. An attacker chooses an arbitrary $2b$-bit string $M_1\|M_2$, where $M_1, M_2 \in \mathcal{B}$, and queries the value of the iteration for $M_1\|M_2$. By the definition of MDP, after the first application of the compression function, the state value is equal to $\pi^{-1}(\tau_1\|v) = \pi^{-1}(G_K(IV\|M_1)\|v)$, where v is the first half of $IV \in \mathcal{C}^2$, and after the second application, to $\pi^{-1}(G_K(\tau_1\|v\|M_2)\|\tau_1)$. From the response of the query for $M_1\|M_2$, we can obtain τ_1 easily. Then the attacker can forge the message-tag pair $(M_1, \pi^{-1}(\tau_1\|v))$ because $\mathsf{MDP}[F_K](M_1) = F_K(\pi(IV)\|M_1) = F_K(IV\|M_1) = \pi^{-1}(\tau_1\|v)$, so this attack always succeeds.

If we put π as an identity function, our attack also provides an example to show that the (strengthened) MD construction does not preserve unforgeability which is different from that of Bellare and Ristenpart [8].

5. Further Results on Indifferentiability

5.1. *MDP with a Double-Block-Length Compression Function*

A compression function F is called double-block-length (DBL) if it is composed of a compression function f and the output length of F is twice as large as that of f. We consider a DBL compression function of the form defined in the following definition.

Definition 1. Let c be an even integer, and $f : \mathcal{C} \times \mathcal{B} \to \{0,1\}^{c/2}$. $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$ is a DBL compression function such that $F(s,x) = f(s,x)\|f(p(s),x)$, where $s \in \mathcal{C}$, $x \in \mathcal{B}$, and p is an involution on $\mathcal{C}$ with no fixed points.

The following theorem states that $\mathsf{MDP}[F,\pi]$ is indifferentiable from a VIL random oracle if f is a FIL random oracle and π is chosen appropriately.

Theorem 5. *Let F be a DBL compression function defined in Definition 1. Let π be a permutation on $\mathcal{C}$ and $P_{\pi,p} = \{u \mid (u \in \mathcal{C}) \wedge ((\pi(u) = u) \vee (\pi(u) = p(u)))\}$. Let A be an adversary distinguishing the pairs $(\mathsf{MDP}[F,\pi], F)$ and (H, S_F), where the simulator*

Initialize:

1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$

Interface $\mathcal{F}(s, x)$:

20: **if** $\mathrm{F}(s, x) = \perp$ **then**
21: **if** $s \in \mathcal{T}$ **then**
22: $\mathrm{F}(s, x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
23: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(s, x)\}$
24: $\mathrm{F}(p(s), x) \leftarrow \mathsf{swap}(\mathrm{F}(s, x))$
25: **else if** $p(s) \in \mathcal{T}$ **then**
26: $\mathrm{F}(p(s), x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
27: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(p(s), x)\}$
28: $\mathrm{F}(s, x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s), x))$
29: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
30: $\tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
31: $\mathrm{F}(s, x) \leftarrow H(\tilde{M} \| x)$
32: $\mathrm{F}(p(s), x) \leftarrow \mathsf{swap}(\mathrm{F}(s, x))$
33: **else if** $\pi^{-1}(p(s)) \in \mathcal{T}$ **then**
34: $\tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(p(s)))$
35: $\mathrm{F}(p(s), x) \leftarrow H(\tilde{M} \| x)$
36: $\mathrm{F}(s, x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s), x))$
37: **else**
38: $\mathrm{F}(s, x) \xleftarrow{\$} \mathcal{C}$
39: $\mathrm{F}(p(s), x) \leftarrow \mathsf{swap}(\mathrm{F}(s, x))$
40: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
41: **return** $\mathrm{F}(s, x)$

Fig. 7. Pseudocode for the simulator S_F. $\mathsf{swap}(t_1 \| t_2) = t_2 \| t_1$ for every $t_1, t_2 \in \{0,1\}^{c/2}$. $\mathcal{C}_{\mathrm{bad}} = \mathcal{V} \cup \mathcal{T} \cup p(\mathcal{T}) \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T} \cup p(\mathcal{T})) \cup \pi(\mathcal{T}) \cup p(\pi(\mathcal{T})) \cup P_{\pi,p}$.

S_F is defined in Fig. 7. Suppose that $IV \notin P_{\pi,p}$. Then,

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_F}(A) \le \frac{5q^2 + (|P_{\pi,p}|+1)q}{2^c} + \frac{4\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 10q - |P_{\pi,p}| - 4},$$

where $q = \ell q_{\mathrm{V}} + q_{\mathrm{F}}$. q_{V} is the number of queries to the VIL oracle, and q_{F} is the number of queries to the FIL oracle. ℓ is the maximum number of message blocks for each VIL query. S_F makes at most q_{F} queries and runs in time $O({q_{\mathrm{F}}}^2)$.

Proof. First of all, it should be justified that a simulator is prepared for F instead of f. Actually, since f is a FIL random oracle, F is completely indifferentiable from a function chosen uniformly at random from $\{R \mid R \in \mathsf{Maps}(\mathcal{C} \times \mathcal{B}, \mathcal{C}) \wedge R(s, x) = \mathsf{swap}(R(p(s), x))\}$. Let $\hat{p}$ be a permutation on $\mathcal{C} \times \mathcal{B}$ such that $\hat{p}(s, x) = (p(s), x)$. Since p has no fixed points and $p \circ p$ is an identity permutation, so does $\hat{p}$. Since $\hat{p} \circ \hat{p}$ is an identity permutation, $f(s, x)$ and $f(\hat{p}(s, x))$ are only used for $F(s, x)$ and $F(\hat{p}(s, x))$ for every $(s, x) \in \mathcal{C} \times \mathcal{B}$. Thus, $F(s, x)$ and $F(s', x')$ are random and independent of each other if $(s', x') \neq \hat{p}(s, x)$, since f is a random oracle. Moreover, since $\hat{p}$ has no fixed points and $F(s, x) = f(s, x) \| f(\hat{p}(s, x))$, the first half and the second half of $F(s, x)$ are also random and independent of each other.

This proof also uses the game-playing technique, and it is similar to that of Theorem 1. Each game exposes two interfaces $\mathcal{H}(M)$ and $\mathcal{F}(s, x)$ to the adversary A.

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$:
10: $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SF}(s_{i-1}, M_i)$
14: $s_k \leftarrow \mathsf{SF}(\pi(s_{k-1}), M_k)$
15: **return** s_k

Interface $\mathcal{F}(s,x)$:
20: $\mathsf{SF}(s,x)$

Function $\mathsf{SF}(s,x)$:
200: **if** $\mathtt{F}(s,x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C}$
203: $\quad\quad$ **if** $\mathtt{F}(s,x) \in \mathcal{C}_{\mathrm{bad}}$ **then** ▷ For G1 only
204: $\quad\quad\quad bad \leftarrow \mathtt{true}$ ▷ For G1 only
205: $\quad\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$ ▷ For G1 only
206: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(s,x)\}$
207: $\quad\quad \mathtt{F}(p(s),x) \leftarrow \mathsf{swap}(\mathtt{F}(s,x))$
208: $\quad$ **else if** $p(s) \in \mathcal{T}$ **then**
209: $\quad\quad \mathtt{F}(p(s),x) \xleftarrow{\$} \mathcal{C}$
210: $\quad\quad$ **if** $\mathtt{F}(p(s),x) \in \mathcal{C}_{\mathrm{bad}}$ **then** ▷ For G1 only
211: $\quad\quad\quad bad \leftarrow \mathtt{true}$ ▷ For G1 only
212: $\quad\quad\quad \mathtt{F}(p(s),x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$ ▷ For G1 only
213: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(p(s),x)\}$
214: $\quad\quad \mathtt{F}(s,x) \leftarrow \mathsf{swap}(\mathtt{F}(p(s),x))$
215: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
216: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C}$
217: $\quad\quad \mathtt{F}(p(s),x) \leftarrow \mathsf{swap}(\mathtt{F}(s,x))$
218: $\quad$ **else if** $\pi^{-1}(p(s)) \in \mathcal{T}$ **then**
219: $\quad\quad \mathtt{F}(p(s),x) \xleftarrow{\$} \mathcal{C}$
220: $\quad\quad \mathtt{F}(s,x) \leftarrow \mathsf{swap}(\mathtt{F}(p(s),x))$
221: $\quad$ **else**
222: $\quad\quad \mathtt{F}(s,x) \xleftarrow{\$} \mathcal{C}$
223: $\quad\quad \mathtt{F}(p(s),x) \leftarrow \mathsf{swap}(\mathtt{F}(s,x))$
224: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
225: **return** $\mathtt{F}(s,x)$

Fig. 8. The games G0 and G1. The lines 203 to 205 and 210 to 212 are active only in G1.

Without loss of generality, we can assume that A makes no repeated queries to $\mathcal{H}$ and $\mathcal{F}$.

The first game is given in Fig. 8. Since G0 is a faithful implementation of MDP with a FIL random oracle, we have

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] = \Pr\left[A\left(\mathsf{MDP}[F,\pi], F\right) \Rightarrow 1\right].$$

(G0 $\rightarrow$ G1). Since G0 and G1 are identical until *bad* gets `true` in G1, we have

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] \leq \Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right].$$

$\Pr[A^{\mathrm{G1}}$ sets *bad*$]$ can be estimated as follows: each of the lines 203 and 210 of G1 is satisfied with probability

$$\frac{|\mathcal{C}_{\mathrm{bad}}|}{|\mathcal{C}|} \leq \frac{2\,|\mathcal{V}| + 6\,|\mathcal{T}| + |P_{\pi,p}|}{2^c}$$

since $\mathcal{C}_{\mathrm{bad}} = \mathcal{V} \cup \mathcal{T} \cup p(\mathcal{T}) \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T} \cup p(\mathcal{T})) \cup \pi(\mathcal{T}) \cup p(\pi(\mathcal{T})) \cup P_{\pi,p}$. Let $\mathcal{V}_i$ and $\mathcal{T}_i$ be the sets $\mathcal{V}$ and $\mathcal{T}$, respectively, at 203 or 210 in the ith invocation of SF. Then,

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$:
10: $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SF}(s_{i-1}, M_i)$
14: $s_k \leftarrow \mathsf{SF}(\pi(s_{k-1}), M_k)$
15: **return** s_k

Interface $\mathcal{F}(s,x)$:
20: $\mathsf{SF}(s,x)$

Function $\mathsf{SH}(M)$:
100: **if** $\mathrm{H}(M) = \perp$ **then**
101: $\quad \mathrm{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathrm{H}(M)$

Function $\mathsf{SF}(s,x)$:
200: **if** $\mathrm{F}(s,x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathrm{F}(s,x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(s,x)\}$
204: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
205: $\quad$ **else if** $p(s) \in \mathcal{T}$ **then**
206: $\quad\quad \mathrm{F}(p(s),x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
207: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(p(s),x)\}$
208: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s),x))$
209: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
210: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
211: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$
212: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
213: $\quad$ **else if** $\pi^{-1}(p(s)) \in \mathcal{T}$ **then**
214: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(p(s)))$
215: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$
216: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s),x))$
217: $\quad$ **else**
218: $\quad\quad \mathrm{F}(s,x) \xleftarrow{\$} \mathcal{C}$
219: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
220: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
221: **return** $\mathrm{F}(s,x)$

Fig. 9. The game G2.

it is easy to see that $|\mathcal{V}_i| \le 2(i-1)$ and $|\mathcal{T}_i| \le i$. SF is invoked at most $q = \ell q_{\mathrm{V}} + q_{\mathrm{F}}$ times in total. Thus,

$$\Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right] \le \sum_{i=1}^{q} \frac{2|\mathcal{V}_i| + 6|\mathcal{T}_i| + |P_{\pi,p}|}{2^c} \le \sum_{i=1}^{q} \frac{10i + |P_{\pi,p}| - 4}{2^c}$$
$$= \frac{5q^2 + (|P_{\pi,p}| + 1)q}{2^c}.$$

(G1 $\rightarrow$ G2). The game G2 is presented in Fig. 9. The difference between G1 and G2 is the selection of $\mathrm{F}(s,x)$ or $\mathrm{F}(p(s),x)$ in case $\pi^{-1}(s) \in \mathcal{T}$ or $\pi^{-1}(p(s)) \in \mathcal{T}$. In G1, $\mathrm{F}(s,x)$ and $\mathrm{F}(p(s),x)$ are randomly chosen at the lines 216 and 219, respectively. In G2, $\mathrm{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$, where $\mathrm{F}^*(IV, \tilde{M}) = \pi^{-1}(s)$, and the actual random selection is deferred by the subroutine SH. There is a one-to-one correspondence between (s,x) and $\tilde{M}\|x$ via $\mathrm{F}^*(IV, \tilde{M}) = \pi^{-1}(s) \in \mathcal{T}$. There is also a one-to-one correspondence between $(p(s),x)$ and $\tilde{M}\|x$ via $\mathrm{F}^*(IV, \tilde{M}) = \pi^{-1}(p(s)) \in \mathcal{T}$. Therefore, in essence G2 simply selects $\mathrm{F}(s,x)$ and $\mathrm{F}(p(s),x)$ randomly. Thus,

$$\Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right].$$

(G2 $\rightarrow$ G3). The game G3 is given in Fig. 10. In G3, a new set $\mathcal{T}_A$ is introduced, which is initially $\{IV\}$. Some operations on $\mathcal{T}_A$ are also added to SF: 205 to 209, 214 to 218, 223 to 224, 229 to 230, and 235 to 241. However, these differences between G2

Initialize:

1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{T}_A \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$ for G3:

10: $M_1 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k - 1$ **do**
13: $\quad s_i \leftarrow \mathsf{SF}(s_{i-1}, M_i)$
14: $s_k \leftarrow \mathsf{SF}(\pi(s_{k-1}), M_k)$
15: **return** s_k

Interface $\mathcal{H}(M)$ for G4:

10: $\mathsf{SH}(M)$

Interface $\mathcal{F}(s, x)$:

20: $\mathsf{SF}(s, x)$

Function $\mathsf{SH}(M)$:

100: **if** $\mathtt{H}(M) = \perp$ **then**
101: $\quad \mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathtt{H}(M)$

Function $\mathsf{SF}(s, x)$:

200: **if** $\mathtt{F}(s, x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathtt{F}(s, x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(s, x)\}$
204: $\quad\quad \mathtt{F}(p(s), x) \leftarrow \mathsf{swap}(\mathtt{F}(s, x))$
205: $\quad\quad$ **if** (s, x) is from $\mathcal{F}$ **then**
206: $\quad\quad\quad$ **if** $s \in \mathcal{T}_A$ **then**
207: $\quad\quad\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(s, x)\}$
208: $\quad\quad\quad$ **else**
209: $\quad\quad\quad\quad bad \leftarrow \mathtt{true}$
210: $\quad$ **else if** $p(s) \in \mathcal{T}$ **then**
211: $\quad\quad \mathtt{F}(p(s), x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
212: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{F}(p(s), x)\}$
213: $\quad\quad \mathtt{F}(s, x) \leftarrow \mathsf{swap}(\mathtt{F}(p(s), x))$
214: $\quad\quad$ **if** (s, x) is from $\mathcal{F}$ **then**
215: $\quad\quad\quad$ **if** $p(s) \in \mathcal{T}_A$ **then**
216: $\quad\quad\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(p(s), x)\}$
217: $\quad\quad\quad$ **else**
218: $\quad\quad\quad\quad bad \leftarrow \mathtt{true}$
219: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
220: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
221: $\quad\quad \mathtt{F}(s, x) \leftarrow \mathsf{SH}(\tilde{M} \| x)$
222: $\quad\quad \mathtt{F}(p(s), x) \leftarrow \mathsf{swap}(\mathtt{F}(s, x))$
223: $\quad\quad$ **if** (s, x) is from $\mathcal{F}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$ **then**
224: $\quad\quad\quad bad \leftarrow \mathtt{true}$
225: $\quad$ **else if** $\pi^{-1}(p(s)) \in \mathcal{T}$ **then**
226: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(p(s)))$
227: $\quad\quad \mathtt{F}(p(s), x) \leftarrow \mathsf{SH}(\tilde{M} \| x)$
228: $\quad\quad \mathtt{F}(s, x) \leftarrow \mathsf{swap}(\mathtt{F}(p(s), x))$
229: $\quad\quad$ **if** (s, x) is from $\mathcal{F}$ and $\pi^{-1}(p(s)) \notin \mathcal{T}_A$ **then**
230: $\quad\quad\quad bad \leftarrow \mathtt{true}$
231: $\quad$ **else**
232: $\quad\quad \mathtt{F}(s, x) \xleftarrow{\$} \mathcal{C}$
233: $\quad\quad \mathtt{F}(p(s), x) \leftarrow \mathsf{swap}(\mathtt{F}(s, x))$
234: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
235: **else if** (s, x) is from $\mathcal{F}$ **then**
236: $\quad$ **if** $s \in \mathcal{T}_A$ **then**
237: $\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(s, x)\}$
238: $\quad$ **else if** $p(s) \in \mathcal{T}_A$ **then**
239: $\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{F}(p(s), x)\}$
240: $\quad$ **else if** $\pi^{-1}(s) \notin \mathcal{T}_A$ and $\pi^{-1}(p(s)) \notin \mathcal{T}_A$ **then**
241: $\quad\quad bad \leftarrow \mathtt{true}$
242: **return** $\mathtt{F}(s, x)$

Fig. 10. The games G3 and G4.

and G3 do not affect the outputs by $\mathcal{H}$ and $\mathcal{F}$. Thus,

$$\Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right].$$

(G3 $\rightarrow$ G4). The difference between G3 and G4 is the implementation of $\mathcal{H}(M)$. In G3, $\mathcal{H}(M)$ is implemented as the MDP iteration, while in G4, $\mathcal{H}(M)$ is just a random oracle.

We will first confirm that the return values of $\mathcal{H}$ are always determined by SH in G3. From this viewpoint, there is no difference between G3 and G4.

Suppose that A asks M to $\mathcal{H}$. Then, $\mathcal{H}$ invokes SF with (s_{i-1}, M_i) for $1 \le i \le k-1$ and $(\pi(s_{k-1}), M_k)$. Notice that each of them may be asked before. In any case, however, SF receives (t_0, M_1), (t_1, M_2), $\ldots$, (t_{k-2}, M_{k-1}), (t_{k-1}, M_k) in this order, where t_i equals s_i or $p(s_i)$ for $0 \le i \le k-2$, and t_{k-1} equals $\pi(s_{k-1})$ or $p(\pi(s_{k-1}))$.

Suppose that SF receives (t_{i-1}, M_i) at the α_ith invocation for the first time in G3 for $1 \le i \le k$. We will show that $\alpha_1 < \alpha_2 < \cdots < \alpha_k$. Since t_0 equals IV or $p(IV)$, $t_0 \in \mathcal{T}$, or $p(t_0) \in \mathcal{T}$. Thus, $s_1 = \mathtt{F}(IV, M_1)$ is selected at 202 or 211 and added to $\mathcal{T}$. Thus, $\alpha_1 < \alpha_2$ since $s_1 \notin \mathcal{V}_{\alpha_1}$. Similarly, $\alpha_2 < \alpha_3 < \cdots < \alpha_{k-1}$ and $s_i \in \mathcal{T}$ for $2 \le i \le k-1$ and $\alpha_{k-1} < \alpha_k$ since $s_{k-1} \notin \pi^{-1}(\mathcal{V}_{\alpha_{k-1}})$. Thus, at the α_kth invocation with (t_{k-1}, M_k), SF invokes SH with M. Notice that $\mathtt{H}(M)$ is undefined before this invocation.

There still be difference between G3 and G4. $\mathcal{H}$ calls SF in G3, while $\mathcal{H}$ does not in G4. In G4, after a query M to $\mathcal{H}$, A obtains the path from IV to $\mathcal{H}(M)$ in $\mathcal{T}$ only by successive queries (t_0, M_1), $\ldots$, (t_{k-2}, M_{k-1}), (t_{k-1}, M_k). Thus, *bad* never gets `true` in G4.

In G3, on the other hand, *bad* may get `true`. A may accidentally find an intermediate part of the paths made by $\mathcal{H}$ without starting from IV. Thus,

$$\Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] \le \Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right].$$

Now, we will evaluate $\Pr[A^{\mathrm{G3}}$ sets *bad*$]$. If *bad* gets `true`

- at 209, then $s \in \mathcal{T}$ and $s \notin \mathcal{T}_A$,
- at 218, then $p(s) \in \mathcal{T}$ and $p(s) \notin \mathcal{T}_A$,
- at 224, then $\pi^{-1}(s) \in \mathcal{T}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$,
- at 230, then $\pi^{-1}(p(s)) \in \mathcal{T}$ and $\pi^{-1}(p(s)) \notin \mathcal{T}_A$,
- at 241, then $\mathtt{F}(s, x)$ has already been set by $\mathcal{H}$, $s \notin \mathcal{T}_A$, $p(s) \notin \mathcal{T}_A$, $\pi^{-1}(s) \notin \mathcal{T}_A$, and $\pi^{-1}(p(s)) \notin \mathcal{T}_A$.

Even in the last case, $s \in \mathcal{T}$, $p(s) \in \mathcal{T}$, $\pi^{-1}(s) \in \mathcal{T}$, or $\pi^{-1}(p(s)) \in \mathcal{T}$. Thus, in any case, if *bad* gets `true`, then it implies that A successfully asks the value of s', $p(s')$, $\pi(s')$, or $p(\pi(s'))$ to $\mathcal{F}$ for some $s' \in \mathcal{T}$ without having access to s' using $\mathcal{F}$ from IV. Since the number of elements in $\mathcal{T}$ fixed by $\mathcal{H}$ is at most ℓq_{V} and they are chosen from $\mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$,

$$\Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right] \le \frac{4\,\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 10\,q - |P_{\pi,p}| - 4}.$$

(G4 $\rightarrow$ G5). The game G5 is presented in Fig. 11. It is a faithful implementation of a true random oracle and the simulator S_F. Thus,

$$\Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right] = \Pr\left[A^{H, S_F} \Rightarrow 1\right].$$

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$:
10: $\mathsf{SH}(M)$

Interface $\mathcal{F}(s,x)$:
20: $\mathsf{SF}(s,x)$

Function $\mathsf{SH}(M)$:
100: **if** $\mathrm{H}(M) = \perp$ **then**
101: $\quad \mathrm{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathrm{H}(M)$

Function $\mathsf{SF}(s,x)$:
200: **if** $\mathrm{F}(s,x) = \perp$ **then**
201: $\quad$ **if** $s \in \mathcal{T}$ **then**
202: $\quad\quad \mathrm{F}(s,x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(s,x)\}$
204: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
205: $\quad$ **else if** $p(s) \in \mathcal{T}$ **then**
206: $\quad\quad \mathrm{F}(p(s),x) \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}_{\mathrm{bad}}$
207: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathrm{F}(p(s),x)\}$
208: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s),x))$
209: $\quad$ **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
210: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
211: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$
212: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
213: $\quad$ **else if** $\pi^{-1}(p(s)) \in \mathcal{T}$ **then**
214: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(p(s)))$
215: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{SH}(\tilde{M}\|x)$
216: $\quad\quad \mathrm{F}(s,x) \leftarrow \mathsf{swap}(\mathrm{F}(p(s),x))$
217: $\quad$ **else**
218: $\quad\quad \mathrm{F}(s,x) \xleftarrow{\$} \mathcal{C}$
219: $\quad\quad \mathrm{F}(p(s),x) \leftarrow \mathsf{swap}(\mathrm{F}(s,x))$
220: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
221: **return** $\mathrm{F}(s,x)$

Fig. 11. The game G5.

G5 is obtained from G4 by removing $\mathcal{T}_A$ in the initialization and all the operations related to $\mathcal{T}_A$ in SF. These modifications do not affect the outputs by $\mathcal{H}$ and $\mathcal{F}$. Thus,

$$\Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right].$$

Combining all of the above, we get

$$\begin{aligned}\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_F}(A) &\le \Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right] + \Pr\left[A^{\mathrm{G3}} \text{ sets } bad\right] \\ &\le \frac{5q^2 + (|P_{\pi,p}|+1)q}{2^c} + \frac{4\ell q_{\mathrm{V}} q_{\mathrm{F}}}{2^c - 10q - |P_{\pi,p}| - 4}.\end{aligned}$$

□

5.2. *MDP with the Davies–Meyer Compression Function*

In this section, we consider the case that F is the Davies–Meyer compression function [25] composed of a block cipher. We show that $\mathsf{MDP}[F,\pi]$ is indifferentiable from a VIL random oracle under the assumption that the underlying block cipher is ideal. Before presenting the theorem, we introduce the ideal cipher model.

Ideal Cipher Model A block cipher with the block length c and the key length b is called a (c,b) block cipher. Let $E : \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ be a (c,b) block cipher. Then, $E(K,\cdot)$ is a permutation for every $K \in \mathcal{B}$, and $D(K,\cdot) = E(K,\cdot)^{-1}$. $E(K,\cdot)$ and $D(K,\cdot)$ are simply denoted by $E_K(\cdot)$ and $D_K(\cdot)$, respectively.

A (c, b) block cipher E is called an ideal cipher if E_K is a truly random permutation for every $K \in \mathcal{B}$. The encryption oracle E first receives a pair of a key and a plaintext as a query, and returns a randomly selected ciphertext. On the other hand, the decryption oracle D first receives a pair of a key and a ciphertext as a query, and returns a randomly selected plaintext. The oracles E and D share a table of triplets of keys, plaintexts, and ciphertexts, which are produced by the queries and the corresponding replies. Referring to the table, they select a reply to a new query under the restriction that E_K is a permutation for every K.

The following theorem states the indifferentiability of MDP with the Davies–Meyer compression function in the ideal cipher model.

Theorem 6. *Let $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$ be the Davies–Meyer compression function with an ideal (c, b) block cipher E, that is, $F(s, x) = E_x(s) \oplus s$. Let π be a permutation on $\mathcal{C}$, and P_π be the set of its fixed points such that $IV \notin P_\pi$. Let A be an adversary that asks at most q_{V} queries to the VIL oracle, q_{e} queries to the FIL encryption oracle, and q_{d} queries to the FIL decryption oracle. Let ℓ be the maximum number of message blocks for each VIL query. Suppose that $\ell q_{\mathrm{V}} \geq 1$, $q_{\mathrm{e}} \geq 1$, and $q_{\mathrm{d}} \geq 1$. Then,*

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_E,S_D}(A) \leq \frac{7q^2 + (20q_{\mathrm{d}} + 4|P_\pi| + 5)q}{2^{c+1}} + \frac{2\ell q_{\mathrm{V}}(q_{\mathrm{e}} + q_{\mathrm{d}})}{2^c - 4q - 3q_{\mathrm{d}} - |P_\pi| + 1},$$

where $q = \ell q_{\mathrm{V}} + q_{\mathrm{e}}$. The simulators S_E and S_D are given in Fig. 12. S_E is a simulator for the encryption oracle, and S_D for the decryption oracle. S_E makes at most q_{e} queries and runs in time $O(q_{\mathrm{e}}(q_{\mathrm{e}} + q_{\mathrm{d}}))$. S_D makes at most $q_{\mathrm{e}}q_{\mathrm{d}}$ queries and runs in time $O(q_{\mathrm{d}}(q_{\mathrm{e}} + q_{\mathrm{d}}))$.

Proof. The proof starts with the game G0 given in Fig. 13. Each game exposes three interfaces $\mathcal{H}(M)$, $\mathcal{E}(x, s)$, and $\mathcal{D}(x, u)$ to the adversary A. Without loss of generality, we can assume that A makes no repeated queries to them.

G0 is a faithful implementation of MDP with the DM compression function composed of an ideal block cipher: SE and SD implements an ideal block cipher by lazy sampling. Actually, SE always chooses the response $\mathtt{E}_x(s)$ uniformly at random from $\mathcal{Q}(x)$ if it is undefined. SD also chooses the response $\mathtt{D}_x(u)$ uniformly at random from $\mathcal{P}(x)$ if it is undefined. Therefore,

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] = \Pr\left[A^{\mathsf{MDP}[F,\pi],E,D} \Rightarrow 1\right].$$

(G0 $\to$ G1). The game G1 is presented in Fig. 14. G0 and G1 are different at the lines 202 and 302, and the lines 206 to 208 in G1 correspond to 206 in G0. Let

$$\begin{aligned} P_1 &= \Pr\left[\mathtt{E}_x(s) \text{ is selected from } \mathcal{C}_{\mathrm{bad}} \text{ at 202 in G0}\right], \\ P_2 &= \Pr\left[\mathtt{D}_x(u) \text{ is selected from } \mathcal{T} \cup \pi(\mathcal{T}) \text{ at 302 in G0}\right], \\ P_3 &= \Pr\left[\mathtt{E}_x(s) \text{ is selected from } \mathcal{C} \setminus \mathcal{Q}(x) \text{ at 206 in G1}\right]. \end{aligned}$$

Then,

$$\Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] \leq P_1 + P_2 + P_3.$$

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{P}(x) \leftarrow \mathcal{C}$
4: $\mathcal{Q}(x) \leftarrow \mathcal{C}$

Interface $\mathcal{E}(x, s)$:
200: **if** $s \in \mathcal{T}$ **then**
201: $\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$
202: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
203: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
204: $\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
205: $\quad \mathtt{E}_x(s) \leftarrow H(\tilde{M} \| x) \oplus s$
206: $\quad$ **if** $\mathtt{E}_x(s) \notin \mathcal{Q}(x)$ **then**
207: $\quad\quad$ **return** `fail`
208: **else**
209: $\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
210: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
211: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
212: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
213: **return** $\mathtt{E}_x(s)$

Interface $\mathcal{D}(x, u)$:
300: $S \leftarrow \emptyset$
301: **for** every $s \in \mathcal{T}$ **do**
302: $\quad \tilde{M} \leftarrow \mathsf{getnode}(s)$
303: $\quad$ **if** $u = H(\tilde{M} \| x) \oplus \pi(s)$ **then**
304: $\quad\quad S \leftarrow S \cup \{s\}$
305: **if** $|S| \geq 2$ **then**
306: $\quad$ **return** `fail`
307: **if** $S = \{s^*\}$ **then**
308: $\quad$ **if** $\pi(s^*) \notin \mathcal{P}(x)$ **then**
309: $\quad\quad$ **return** `fail`
310: $\quad$ **else**
311: $\quad\quad \mathtt{D}_x(u) \leftarrow \pi(s^*)$
312: **else**
313: $\quad \mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
314: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
315: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
316: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
317: **return** $\mathtt{D}_x(u)$

Fig. 12. Pseudocode for the simulators S_E and S_D. $\mathcal{P}(x)$ and $\mathcal{C}(x)$ are the sets of plaintexts and ciphertexts, respectively, available as a reply to a query with the key x. E and D store values set by previous queries and replies. Initially, $\mathtt{E}_x(s) = \perp$ and $\mathtt{D}_x(u) = \perp$ for every x, s and u. Let (x_i, s_i, u_i) be the values set by the ith query and the corresponding reply. Then, $\mathtt{E}_{x_i}(s_i) = u_i$ and $\mathtt{D}_{x_i}(u_i) = s_i$. $\mathcal{C}_{\mathrm{bad}} = \{u \mid u \in \mathcal{C} \wedge u \oplus s \in \mathcal{V} \cup \mathcal{T} \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T}) \cup \pi(\mathcal{T}) \cup P_\pi\}$.

For P_1, since $\mathcal{C}_{\mathrm{bad}} = \{u \mid u \in \mathcal{C} \wedge u \oplus s \in \mathcal{V} \cup \mathcal{T} \cup \pi^{-1}(\mathcal{V} \cup \mathcal{T}) \cup \pi(\mathcal{T}) \cup P_\pi\}$,

$$\frac{|\mathcal{C}_{\mathrm{bad}}|}{|\mathcal{Q}(x)|} \leq \frac{2\,|\mathcal{V} \cup \mathcal{T}| + |\mathcal{T}| + |P_\pi|}{|\mathcal{Q}(x)|}.$$

Let $\mathcal{V}_i$, $\mathcal{T}_i$, and $\mathcal{Q}_i(x_i)$ be the sets $\mathcal{V}$, $\mathcal{T}$, and $\mathcal{Q}(x)$, respectively, in the ith invocation of SE. Then, it is easy to see that $|\mathcal{V}_i \cup \mathcal{T}_i| \leq i + q_{\mathrm{d}}$, $|\mathcal{T}_i| \leq i$, and $|\mathcal{Q}_i(x_i)| \geq 2^c - (i-1) - q_{\mathrm{d}}$ at the line 202. SE is invoked at most $q = \ell q_{\mathrm{V}} + q_{\mathrm{e}}$ times in total. Thus,

$$\begin{aligned} P_1 &\leq \sum_{i=1}^{q} \frac{2\,|\mathcal{V}_i \cup \mathcal{T}_i| + |\mathcal{T}_i| + |P_\pi|}{|\mathcal{Q}_i(x_i)|} \leq \sum_{i=1}^{q} \frac{3\,i + 2\,q_{\mathrm{d}} + |P_\pi|}{2^c - (i + q_{\mathrm{d}} - 1)} \\ &\leq \frac{3\,q^2 + (4\,q_{\mathrm{d}} + 2\,|P_\pi| + 3)\,q}{2\,(2^c - (q + q_{\mathrm{d}} - 1))}. \end{aligned}$$

For P_2, let $\mathcal{T}_j$ and $\mathcal{P}_j(x_j)$ be the sets $\mathcal{T}$ and $\mathcal{P}(x)$, respectively, in the jth invocation of SD. Then, $|\mathcal{T}_j| \leq q$, and $|\mathcal{P}_j(x_j)| \geq 2^c - (j-1) - q$ at the line 302. SD is invoked at most q_{d} times in total. Thus,

$$P_2 \leq \sum_{j=1}^{q_{\mathrm{d}}} \frac{2\,|\mathcal{T}_j|}{|\mathcal{P}_j(x_j)|} \leq \frac{2\,q\,q_{\mathrm{d}}}{2^c - (q + q_{\mathrm{d}} - 1)}.$$

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{P}(x) \leftarrow \mathcal{C}$
4: $\mathcal{Q}(x) \leftarrow \mathcal{C}$

Interface $\mathcal{H}(M)$:
10: $M_1 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SE}(M_i, s_{i-1}) \oplus s_{i-1}$
14: $s_k \leftarrow \mathsf{SE}(M_k, \pi(s_{k-1})) \oplus \pi(s_{k-1})$
15: **return** s_k

Interface $\mathcal{E}(x, s)$:
20: $\mathsf{SE}(x, s)$

Interface $\mathcal{D}(x, u)$:
30: $\mathsf{SD}(x, u)$

Function $\mathsf{SE}(x, s)$:
200: **if** $s \in \mathcal{T}$ **then**
201: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
202: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
203: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
204: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
205: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
206: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
207: **else**
208: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
209: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
210: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
211: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
212: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
213: **return** $\mathtt{E}_x(s)$

Function $\mathsf{SD}(x, u)$:
300: **if** $\mathtt{D}_x(u) \neq \perp$ **then**
301: **else**
302: $\quad \mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x)$
303: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
304: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
305: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
306: **return** $\mathtt{D}_x(u)$

Fig. 13. The game G0.

For P_3, since $|\mathcal{Q}_i(x_i)| \geq 2^c - (i-1) - q_{\mathrm{d}}$ at the line 206,

$$P_3 \leq \sum_{i=1}^{q} \frac{|\mathcal{C} \setminus \mathcal{Q}_i(x_i)|}{|\mathcal{C}|} \leq \frac{q^2 + (2q_{\mathrm{d}} - 1)q}{2^{c+1}}.$$

(G1 $\rightarrow$ G2). The game G2 is presented in Fig. 15. The difference between G1 and G2 is the selection of $\mathtt{E}_x(s)$ in case $\pi^{-1}(s) \in \mathcal{T}$ in SE. In G1, it is randomly chosen at the line 206. In G2, SH is newly introduced, and $\mathtt{E}_x(s) \leftarrow \mathsf{SH}(\tilde{M}\|x) \oplus s$, where $\tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$. There is a one-to-one correspondence between (s, x) and $\tilde{M}\|x$. Therefore, if $\mathtt{E}_x(s) = \perp$, then $\mathtt{H}(\tilde{M}\|x) = \perp$. Namely, G2 simply selects $\mathtt{E}_x(s)$ randomly in SH. Thus,

$$\Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right].$$

(G2 $\rightarrow$ G3). The game G3 is presented in Fig. 15. There exists difference only in SD. Since A makes no repeated queries, if $\mathtt{D}_x(u) \neq \perp$, then it has been set by a query to $\mathcal{H}$. Namely, $\mathtt{D}_x(u) \in \mathcal{T} \cup \pi(\mathcal{T})$. Thus,

$$\Pr\left[A^{\mathrm{G2}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right].$$

(G3 $\rightarrow$ G4). The game G4 is also presented in Fig. 15. There is difference in SD between G3 and G4. The line 302 of G3 is replaced by 302 to 310 of G4. In the case

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{P}(x) \leftarrow \mathcal{C}$
4: $\mathcal{Q}(x) \leftarrow \mathcal{C}$

Interface $\mathcal{H}(M)$:
10: $M_1 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SE}(M_i, s_{i-1}) \oplus s_{i-1}$
14: $s_k \leftarrow \mathsf{SE}(M_k, \pi(s_{k-1})) \oplus \pi(s_{k-1})$
15: **return** s_k

Interface $\mathcal{E}(x, s)$:
20: $\mathsf{SE}(x, s)$

Interface $\mathcal{D}(x, u)$:
30: $\mathsf{SD}(x, u)$

Function $\mathsf{SE}(x, s)$:
200: **if** $s \in \mathcal{T}$ **then**
201: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
202: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
204: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
205: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
206: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{C}$
207: $\quad\quad$ **if** $\mathtt{E}_x(s) \notin \mathcal{Q}(x)$ **then**
208: $\quad\quad\quad$ **return** `fail`
209: **else**
210: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
211: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
212: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
213: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
214: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
215: **return** $\mathtt{E}_x(s)$

Function $\mathsf{SD}(x, u)$:
300: **if** $\mathtt{D}_x(u) \neq \perp$ **then**
301: **else**
302: $\quad \mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
303: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
304: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
305: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
306: **return** $\mathtt{D}_x(u)$

Fig. 14. The game G1.

of $\mathtt{D}_x(u) \in \pi(\mathcal{T})$, SD of G4 simply tries to identify the unique node $s \in \mathcal{T}$ such that $\mathtt{D}_x(u) = \pi(s)$. Thus, G3 and G4 are equivalent unless $u = \mathsf{SH}(\tilde{M} \| x) \oplus \pi(s)$ for some $s \in \mathcal{T}$ such that $\tilde{M} \| x$ is not given to $\mathcal{H}$ by A. Let us call this event *Hit*. Notice that the condition at the line 307 gets `true` only if *Hit* gets `true`. Thus, since $|\mathcal{T}| \leq q$,

$$\Pr\left[A^{\mathrm{G3}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] \leq \Pr\left[\textit{Hit} \text{ gets } \mathtt{true}\right] \leq \frac{q\, q_{\mathrm{d}}}{2^c}.$$

(G4 $\rightarrow$ G5). The game G5 is presented in Fig. 16. In G5, a new set $\mathcal{T}_A$ is introduced, which is initially $\{IV\}$. Some operations on $\mathcal{T}_A$ are added to SE: from 204 to 208, from 210 to 211. The line 301 is also added to SD. However, these differences do not affect the outputs by $\mathcal{H}$, $\mathcal{E}$, and $\mathcal{D}$. Thus,

$$\Pr\left[A^{\mathrm{G4}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right].$$

(G5 $\rightarrow$ G6). The game G6 is also given in Fig. 16. The difference between G5 and G6 is the implementation of $\mathcal{H}(M)$. In G5, $\mathcal{H}(M)$ is implemented as the MDP iteration, while in G6, $\mathcal{H}(M)$ is just a random oracle.

We will first confirm that the return values of $\mathcal{H}$ are always determined by SH in G5. From this viewpoint, there is no difference between G5 and G6.

Initialize:

$\mathcal{V} \leftarrow \emptyset$
$\mathcal{T} \leftarrow \{IV\}$
$\mathcal{P}(x) \leftarrow \mathcal{C}$
$\mathcal{Q}(x) \leftarrow \mathcal{C}$

Interface $\mathcal{H}(M)$:

$M_1 \cdots M_k \leftarrow \mathsf{parse}(M)$
$s_0 \leftarrow IV$
for $i = 1$ to $k-1$ **do**
  $s_i \leftarrow \mathsf{SE}(M_i, s_{i-1}) \oplus s_{i-1}$
$s_k \leftarrow \mathsf{SE}(M_k, \pi(s_{k-1})) \oplus \pi(s_{k-1})$
return s_k

Interface $\mathcal{E}(x, s)$:

$\mathsf{SE}(x, s)$

Interface $\mathcal{D}(x, u)$:

$\mathsf{SD}(x, u)$

Function $\mathsf{SH}(M)$:

if $\mathtt{H}(M) = \perp$ **then**
  $\mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
return $\mathtt{H}(M)$

Function $\mathsf{SE}(x, s)$:

if $s \in \mathcal{T}$ **then**
  if $\mathtt{E}_x(s) = \perp$ **then**
    $\mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
else if $\pi^{-1}(s) \in \mathcal{T}$ **then**
  if $\mathtt{E}_x(s) = \perp$ **then**
    $\tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
    $\mathtt{E}_x(s) \leftarrow \mathsf{SH}(\tilde{M} \| x) \oplus s$
    if $\mathtt{E}_x(s) \notin \mathcal{Q}(x)$ **then**
      return `fail`
else
  if $\mathtt{E}_x(s) = \perp$ **then**
    $\mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
$\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
$\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
return $\mathtt{E}_x(s)$

Function $\mathsf{SD}(x, u)$ for G2:

if $\mathtt{D}_x(u) \neq \perp$ **then**
else
  $\mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
$\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
$\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
return $\mathtt{D}_x(u)$

Function $\mathsf{SD}(x, u)$ for G3:

if $\mathtt{D}_x(u) \in \mathcal{T}$ **then**
else
  if $\mathtt{D}_x(u) \in \pi(\mathcal{T})$ **then**
  else
    $\mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
$\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
$\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
return $\mathtt{D}_x(u)$

Function $\mathsf{SD}(x, u)$ for G4:

if $\mathtt{D}_x(u) \in \mathcal{T}$ **then**
else
  $S \leftarrow \emptyset$
  for all $s \in \mathcal{T}$ **do**
    $\tilde{M} \leftarrow \mathsf{getnode}(s)$
    if $u = \mathsf{SH}(\tilde{M} \| x) \oplus \pi(s)$ **then**
      $S \leftarrow S \cup \{s\}$
  if $|S| \geq 2$ **then**
    return `fail`
  if $S = \{s^*\}$ **then**
    $\mathtt{D}_x(u) \leftarrow \pi(s^*)$
  else
    $\mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
$\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
$\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
return $\mathtt{D}_x(u)$

Fig. 15. The games G2, G3 and G4.

Suppose that A asks M to $\mathcal{H}$. Then, $\mathcal{H}$ invokes SE with (M_i, s_{i-1}) for $1 \leq i \leq k-1$ and $(M_k, \pi(s_{k-1}))$. Notice that each of them may be asked before. In any case, however, SE receives $(M_1, s_0), (M_2, s_1), \ldots, (M_{k-1}, s_{k-2}), (M_k, \pi(s_{k-1}))$ in this order.

Suppose that SE receives (M_i, s_{i-1}) at the α_ith invocation for $1 \leq i \leq k-1$ and $(M_k, \pi(s_{k-1}))$ at the α_kth invocation for the first time in G5. We will show

Initialize:

1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{P}(x) \leftarrow \mathcal{C}$
4: $\mathcal{Q}(x) \leftarrow \mathcal{C}$
5: $\mathcal{T}_A \leftarrow \{IV\}$

Interface $\mathcal{H}(M)$ for G5:

10: $M_1 \cdots M_k \leftarrow \mathsf{parse}(M)$
11: $s_0 \leftarrow IV$
12: **for** $i = 1$ to $k-1$ **do**
13: $\quad s_i \leftarrow \mathsf{SE}(M_i, s_{i-1}) \oplus s_{i-1}$
14: $s_k \leftarrow \mathsf{SE}(M_k, \pi(s_{k-1})) \oplus \pi(s_{k-1})$
15: **return** s_k

Interface $\mathcal{H}(M)$ for G6:

10: $\mathsf{SH}(M)$

Interface $\mathcal{E}(x,s)$:

20: $\mathsf{SE}(x,s)$

Interface $\mathcal{D}(x,u)$:

30: $\mathsf{SD}(x,u)$

Function $\mathsf{SH}(M)$:

100: **if** $\mathtt{H}(M) = \perp$ **then**
101: $\quad \mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathtt{H}(M)$

Function $\mathsf{SE}(x,s)$:

200: **if** $s \in \mathcal{T}$ **then**
201: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
202: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$
203: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
204: $\quad$ **if** (x,s) is from $\mathcal{E}$ **then**
205: $\quad\quad$ **if** $s \in \mathcal{T}_A$ **then**
206: $\quad\quad\quad \mathcal{T}_A \leftarrow \mathcal{T}_A \cup \{\mathtt{E}_x(s) \oplus s\}$
207: $\quad\quad$ **else**
208: $\quad\quad\quad bad \leftarrow \mathtt{true}$
209: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
210: $\quad$ **if** (x,s) is from $\mathcal{E}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$ **then**
211: $\quad\quad bad \leftarrow \mathtt{true}$
212: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
213: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
214: $\quad\quad \mathtt{E}_x(s) \leftarrow \mathsf{SH}(\tilde{M} \| x) \oplus s$
215: $\quad\quad$ **if** $\mathtt{E}_x(s) \notin \mathcal{Q}(x)$ **then**
216: $\quad\quad\quad$ **return** $\mathtt{fail}$
217: **else**
218: $\quad$ **if** $\mathtt{E}_x(s) = \perp$ **then**
219: $\quad\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
220: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
221: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
222: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
223: **return** $\mathtt{E}_x(s)$

Function $\mathsf{SD}(x,u)$:

300: **if** $\mathtt{D}_x(u) \in \mathcal{T}$ **then**
301: $\quad bad \leftarrow \mathtt{true}$
302: **else**
303: $\quad S \leftarrow \emptyset$
304: $\quad$ **for all** $s \in \mathcal{T}$ **do**
305: $\quad\quad \tilde{M} \leftarrow \mathsf{getnode}(s)$
306: $\quad\quad$ **if** $u = \mathsf{SH}(\tilde{M} \| x) \oplus \pi(s)$ **then**
307: $\quad\quad\quad S \leftarrow S \cup \{s\}$
308: $\quad$ **if** $|S| \geq 2$ **then**
309: $\quad\quad$ **return** $\mathtt{fail}$
310: $\quad$ **if** $S = \{s^*\}$ **then**
311: $\quad\quad$ **if** $\pi(s^*) \notin \mathcal{P}(x)$ **then** ▷ For G6 only
312: $\quad\quad\quad$ **return** $\mathtt{fail}$ ▷ For G6 only
313: $\quad\quad$ **else** ▷ For G6 only
314: $\quad\quad\quad \mathtt{D}_x(u) \leftarrow \pi(s^*)$
315: $\quad$ **else**
316: $\quad\quad \mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
317: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
318: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
319: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
320: **return** $\mathtt{D}_x(u)$

Fig. 16. The games G5 and G6.

that $\alpha_1 < \alpha_2 < \cdots < \alpha_k$. Since $s_0 = IV \in \mathcal{T}$, $\mathtt{E}_{M_1}(s_0)$ is selected at 202, and $s_1 (= \mathtt{E}_{M_1}(s_0) \oplus s_0)$ is added to $\mathcal{T}$ at 203. Thus, $\alpha_1 < \alpha_2$ since $s_1 \notin \mathcal{V}_{\alpha_1}$. Similarly, $\alpha_2 < \alpha_3 < \cdots < \alpha_{k-1}$ and $s_i \in \mathcal{T}$ for $2 \le i \le k-1$. $\alpha_{k-1} < \alpha_k$ since $s_{k-1} \notin \pi^{-1}(\mathcal{V}_{\alpha_{k-1}})$. Thus, at the α_kth invocation with $(M_k, \pi(s_{k-1}))$, SE invokes $\mathsf{SH}(M)$. Notice that $(M_1, s_0), (M_2, s_1), \ldots, (M_{k-1}, s_{k-2}), (M_k, \pi(s_{k-1}))$ cannot be set by SD because of the line 316.

There still be difference between G5 and G6. $\mathcal{H}$ calls SE in G5, while it does not in G6. In G6, SE is called only by $\mathcal{E}$, and A makes no repeated queries. Therefore, *bad* never gets `true` at 208, 211, and 301.

In G5, on the other hand, *bad* may get `true`. If *bad* gets `true`

- at 208, then $s \in \mathcal{T}$ and $s \notin \mathcal{T}_A$,
- at 211, then $\pi^{-1}(s) \in \mathcal{T}$ and $\pi^{-1}(s) \notin \mathcal{T}_A$,
- at 301, then $\mathtt{D}_x(u)$ has already been set by $\mathcal{H}$ and $u \in \mathcal{T}$.

In any case, if *bad* gets `true`, then it implies that A successfully asks the value of s' or $\pi(s')$ for some $s' \in \mathcal{T}$ without having access to s' using $\mathcal{E}$ from IV. Since the number of elements in $\mathcal{T}$ fixed by $\mathcal{H}$ is at most ℓq_{V} and they are chosen from $\mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$,

$$\Pr\left[A^{\mathrm{G5}} \text{ sets } bad\right] \le \frac{2\,\ell q_{\mathrm{V}}(q_{\mathrm{e}} + q_{\mathrm{d}})}{2^c - 4q - 3q_{\mathrm{d}} - |P_\pi| + 1}.$$

G5 and G6 are identical until *bad* gets `true` in G5 since the lines 311 to 313 in G6 correspond to the lines 215 to 216, which has already been considered in the transformation from G0 to G1. Thus,

$$\Pr\left[A^{\mathrm{G5}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G6}} \Rightarrow 1\right] \le \Pr\left[A^{\mathrm{G5}} \text{ sets } bad\right].$$

(G6 $\rightarrow$ G7). The game G7 is presented in Fig. 17. It is obtained from G6 by removing $\mathcal{T}_A$ in the initialization, the operations related to it in SE: 204 to 208 and 210 to 211. Moreover, since $\mathtt{E}_x(s) = \perp$ for every query (x, s) to $\mathcal{E}$, the lines 201, 212, and 218 are removed from SE. The lines 300 to 302 are also removed from SD. These modifications do not affect the outputs by $\mathcal{H}$, $\mathcal{E}$ and $\mathcal{D}$. Thus,

$$\Pr\left[A^{\mathrm{G6}} \Rightarrow 1\right] = \Pr\left[A^{\mathrm{G7}} \Rightarrow 1\right].$$

G7 is a faithful implementation of a true random oracle and the simulators S_E and S_D. Thus,

$$\Pr\left[A^{\mathrm{G7}} \Rightarrow 1\right] = \Pr\left[A^{H, S_E, S_D} \Rightarrow 1\right].$$

Combining all of the above, we get

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_E,S_D}(A) \le \frac{3q^2 + (8q_{\mathrm{d}} + 2|P_\pi| + 3)q}{2\,(2^c - (q + q_{\mathrm{d}} - 1))} + \frac{q^2 + (4q_{\mathrm{d}} - 1)q}{2^{c+1}} + \frac{2\,\ell q_{\mathrm{V}}(q_{\mathrm{e}} + q_{\mathrm{d}})}{2^c - 4q - 3q_{\mathrm{d}} - |P_\pi| + 1}.$$

Initialize:
1: $\mathcal{V} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \{IV\}$
3: $\mathcal{P}(x) \leftarrow \mathcal{C}$
4: $\mathcal{Q}(x) \leftarrow \mathcal{C}$

Interface $\mathcal{H}(M)$:
10: $\mathsf{SH}(M)$

Interface $\mathcal{E}(x, s)$:
20: $\mathsf{SE}(x, s)$

Interface $\mathcal{D}(x, u)$:
30: $\mathsf{SD}(x, u)$

Function $\mathsf{SH}(M)$:
100: **if** $\mathtt{H}(M) = \perp$ **then**
101: $\quad \mathtt{H}(M) \xleftarrow{\$} \mathcal{C}$
102: **return** $\mathtt{H}(M)$

Function $\mathsf{SE}(x, s)$:
200: **if** $s \in \mathcal{T}$ **then**
201: $\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathcal{C}_{\mathrm{bad}}$
202: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathtt{E}_x(s) \oplus s\}$
203: **else if** $\pi^{-1}(s) \in \mathcal{T}$ **then**
204: $\quad \tilde{M} \leftarrow \mathsf{getnode}(\pi^{-1}(s))$
205: $\quad \mathtt{E}_x(s) \leftarrow \mathsf{SH}(\tilde{M} \| x) \oplus s$
206: $\quad$ **if** $\mathtt{E}_x(s) \notin \mathcal{Q}(x)$ **then**
207: $\quad\quad$ **return** `fail`
208: **else**
209: $\quad \mathtt{E}_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
210: $\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
211: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
212: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{\mathtt{E}_x(s)\}$
213: **return** $\mathtt{E}_x(s)$

Function $\mathsf{SD}(x, u)$:
300: $S \leftarrow \emptyset$
301: **for** every $s \in \mathcal{T}$ **do**
302: $\quad \tilde{M} \leftarrow \mathsf{getnode}(s)$
303: $\quad$ **if** $u = \mathsf{SH}(\tilde{M} \| x) \oplus \pi(s)$ **then**
304: $\quad\quad S \leftarrow S \cup \{s\}$
305: **if** $|S| \geq 2$ **then**
306: $\quad$ **return** `fail`
307: **if** $S = \{s^*\}$ **then**
308: $\quad$ **if** $\pi(s^*) \notin \mathcal{P}(x)$ **then**
309: $\quad\quad$ **return** `fail`
310: $\quad$ **else**
311: $\quad\quad \mathtt{D}_x(u) \leftarrow \pi(s^*)$
312: **else**
313: $\quad \mathtt{D}_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{T} \cup \pi(\mathcal{T}))$
314: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathtt{D}_x(u)\}$
315: $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{\mathtt{D}_x(u)\}$
316: $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
317: **return** $\mathtt{D}_x(u)$

Fig. 17. The game G7.

If $q + q_{\mathrm{d}} \leq 2^{c-1}$, then the first term on the right side is less than or equal to

$$\frac{3q^2 + (8q_{\mathrm{d}} + 2|P_\pi| + 3)q}{2^c}.$$

If $\ell q_{\mathrm{V}} \geq 1$, $q_{\mathrm{e}} \geq 1$, $q_{\mathrm{d}} \geq 1$, and $q + q_{\mathrm{d}} \geq 2^{c-1}$, then it is greater than 1. Thus,

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{MDP}[F,\pi],S_E,S_D}(A) \leq \frac{7q^2 + (20q_{\mathrm{d}} + 4|P_\pi| + 5)q}{2^{c+1}} + \frac{2\ell q_{\mathrm{V}}(q_{\mathrm{e}} + q_{\mathrm{d}})}{2^c - 4q - 3q_{\mathrm{d}} - |P_\pi| + 1}.$$

□

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The work was done while the third author was at the Electronics and Telecommunications Research Institute.

References

[1] E. Andreeva, G. Neven, B. Preneel, T. Shrimpton, Seven-property-preserving iterated hashing: ROX, in *Advances in Cryptology—ASIACRYPT 2007*. LNCS, vol. 4833 (2007), pp. 130–146
[2] J.H. An, M. Bellare, Constructing VIL-MACs from FIL-MACs: message authentication under weakened assumptions, in *Advances in Cryptology—CRYPTO'99*. LNCS, vol. 1666 (1999), pp. 252–269
[3] M. Bellare, New proofs for NMAC and HMAC: security without collision-resistance, in *Advances in Cryptology—CRYPTO 2006*. LNCS, vol. 4117 (2006), pp. 602–619
[4] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, in *Advances in Cryptology—CRYPTO'96*. LNCS, vol. 1109 (1996), pp. 1–15
[5] M. Bellare, R. Canetti, H. Krawczyk, Pseudorandom functions revisited: the cascade construction and its concrete security, in *Proc. of FOCS'96* (1996), pp. 514–523
[6] M. Bellare, T. Kohno, A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications, in *Advances in Cryptology—EUROCRYPT 2003*. LNCS, vol. 2656 (2003), pp. 491–506
[7] M. Bellare, T. Ristenpart, Multi-property-preserving hash domain extension and the EMD transform, in *Advances in Cryptology—ASIACRYPT 2006*. LNCS, vol. 4284 (2006), pp. 299–314
[8] M. Bellare, T. Ristenpart, Hash functions in the dedicated-key setting: design choices and MPP transforms, in *Automata, Languages and Programming—ICALP 2007*. LNCS, vol. 4596 (2007), pp. 399–410
[9] M. Bellare, P. Rogaway, The security of triple encryption and a framework for code-based game-playing proofs, in *Advances in Cryptology—EUROCRYPT 2006*. LNCS, vol. 4004 (2006), pp. 409–426
[10] R. Bhattacharyya, A. Mandal, M. Nandi, Indifferentiability characterization of hash functions and optimal bounds of popular domain extensions, in *Progress in Cryptology—INDOCRYPT 2009*. LNCS, vol. 5922 (2009), pp. 199–218
[11] B. den Boer, A. Mosselaers, Collisions for the compression function of MD5, in *Advances in Cryptology—EUROCRYPT'93*. LNCS, vol. 765 (1994), pp. 293–304
[12] D. Chang, S. Lee, M. Nandi, M. Yung, Indifferentiable security analysis of popular hash function with prefix-free padding, in *Advances in Cryptology—ASIACRYPT 2006*. LNCS, vol. 4284 (2006), pp. 283–298
[13] D. Chang, M. Nandi, Improved indifferentiability security analysis of chopMD hash function, in *Fast Software Encryption—FSE 2008*. LNCS, vol. 5086 (2008), pp. 429–443
[14] S. Contini, Y.L. Yin, Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions, in *Advances in Cryptology—ASIACRYPT 2006*. LNCS, vol. 4284 (2006), pp. 37–53
[15] J.-S. Coron, Y. Dodis, C. Malinaud, P. Puniya, Merkle–Damgård revisited: how to construct a hash function, in *Advances in Cryptology—CRYPTO 2005*. LNCS, vol. 3621 (2005), pp. 430–448
[16] I. Damgård, A design principle for hash functions, in *Advances in Cryptology—CRYPTO'89*. LNCS, vol. 435 (1989), pp. 416–427
[17] S. Hirose, H. Kuwakado, A scheme to base a hash function on a block cipher, in *Selected Areas in Cryptography—SAC 2008*. LNCS, vol. 5381 (2008), pp. 262–275
[18] S. Hirose, J.H. Park, A. Yun, A simple variant of the Merkle–Damgård scheme with a permutation, in *Advances in Cryptology—ASIACRYPT 2007*. LNCS, vol. 4833 (2007), pp. 113–129
[19] J. Kelsey, in *Public Comments on the Draft Federal Information Processing Standard (FIPS) Draft FIPS 180-2, Secure Hash Standard (SHS)* (2001)
[20] J. Kim, A. Biryukov, B. Preneel, S. Lee, On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1, in *Security and Cryptography for Networks—SCN 2006*. LNCS, vol. 4116 (2006), pp. 242–256
[21] J. Lee, J.P. Steinberger, Multi-property-preserving domain extension using polynomial-based modes of operation, in *Advances in Cryptology—EUROCRYPT 2010*. LNCS, vol. 6110 (2010), pp. 573–596

[22] S. Lucks, A failure-friendly design principle for hash functions, in *Advances in Cryptology—ASIACRYPT 2005*. LNCS, vol. 3788 (2005), pp. 474–494
[23] U.M. Maurer, R. Renner, C. Holenstein, Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology, in *Theory of Cryptography—TCC 2004*. LNCS, vol. 2951 (2004), pp. 21–39
[24] U. Maurer, J. Sjödin, Single-key AIL-MACs from any FIL-MAC, in *Automata, Languages and Programming—ICALP 2005*. LNCS, vol. 3580 (2005), pp. 472–484
[25] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, Boca Raton, 1996)
[26] R. Merkle, One way hash functions and DES, in *Advances in Cryptology—CRYPTO'89*. LNCS, vol. 435 (1989), pp. 428–446
[27] M. Nandi, Towards optimal double-length hash functions, in *Progress in Cryptology—INDOCRYPT 2005*. LNCS, vol. 3797 (2005), pp. 77–89
[28] B. Preneel, R. Govaerts, J. Vandewalle, Hash functions based on block ciphers: a synthetic approach, in *Advances in Cryptology—CRYPTO'93*. LNCS, vol. 773 (1994), pp. 368–378
[29] P. Rogaway, T. Shrimpton, Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance, in *Fast Software Encryption—FSE 2004*. LNCS, vol. 3017 (2004), pp. 371–388
[30] G. Tsudik, Message authentication with one-way hash functions. *ACM Comput. Commun. Rev.* **22**(5), 29–38 (1992)
[31] K. Yasuda, A double-piped mode of operation for MACs, PRFs and PROs: security beyond the birthday barrier, in *Advances in Cryptology—EUROCRYPT 2009*. LNCS, vol. 5479 (2009), pp. 242–259
[32] K. Yasuda, HMAC without the "second" key, in *Information Security—ISC 2009*. LNCS, vol. 5735 (2009), pp. 443–458