A Simple Variant of the Merkle-Damgård Scheme with a Permutation

Shoichi Hirose[1], Je Hong Park[2], and Aaram Yun[2]

[1] Graduate School of Engineering, The University of Fukui
`hirose@fuee.fukui-u.ac.jp`
[2] ETRI Network & Communication Security Division
`jhpark@etri.re.kr, aaramyun@gmail.com`

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. And we study the security of simple MAC constructions out of this scheme. Finally, we also 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.

1 Introduction

Background. Merkle-Damgård [19, 12] 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 a hash function that can process strings of arbitrary length.[3] While it is a clean design with proven collision resistance, it suffers the extension property; one can compute $H(M_1 \| M_2)$ from $H(M_1)$ even without the knowledge 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, 2]. It has a simple structure, and also it has rigorous security proofs. But, given a hash function $H(\cdot)$, one of the best ways to make a MAC out of H is the prefix construction [22]:

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

Indeed, the efficiency of the above construction would be almost twice than that of the HMAC, for short messages, and we know that if $H(\cdot)$ is a random oracle, rather than a concrete hash algorithm, then the construction gives a secure MAC. Unfortunately, due to the extension property, the prefix construction is not secure when the underlying hash function is an MD hash function; given a

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

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 had to 'fix' the extension property of the underlying MD hash function, at the upper MAC construction level.

But then 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 new methodology for assessing generic, structural properties of hash function constructions [11]. They applied the notion of indifferentiability, which was first introduced by Maurer et al. [16], 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 a FIL random oracle, then comparing the resulting hash function with a true random oracle. If no efficient distinguisher can tell the two objects 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 MD scheme, which was the main obstacle for adopting the simple constructions like the prefix construction instead of 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 a 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 $M = M_1M_2\cdots M_k$,

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

In this paper, we describe the MDP composition scheme, and 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 also a PRF when keyed via the IV if the compression function is a PRF, secure against 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 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 with a dedicated key 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, as a PRF it also gives a secure MAC mechanism which is twice as fast as HMAC for short messages.

We also study the random oracle indifferentiability of MDP when the underlying compression function has some structure; we consider MDP with two specific type of compression functions. The 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 smaller 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 Works. A hash function composition scheme very similar to MDP was suggested before; in a public comment to a FIPS 180-2 draft, Kelsey [14] 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 π is equal to $\pi(x) = x \oplus C$, where C is a fixed, non-zero 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 constructions which satisfy indifferentiability from a random oracle [11], thereby proving that such schemes exist. Also, Bellare and Ristenpart proposed the EMD construction [6]. Probably it is the first paper that succeeded in finding a serious practical alternative to the MD scheme which meets the raised security goals (like, indifferentiability to a random oracle, among others). Similar to MDP, EMD is also a variant of the MD scheme. Also, EMD achieves essentially the same goals as MDP, 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 in the case of EMD; assuming that the compression function is secure as PRF under a very weak related-key attack, we prove that the keyed MDP is secure as a PRF. Therefore, at least for PRFness, 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, while achieving the security of only one key due to the divide-and-conquer attack. One may consider a one-key version of EMD by employing some key derivation function similar to the case of HMAC, but then one would need additional assumption on the compression function, namely PRF-security under some related-key attacks, 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 secure MAC, by prefix construction $H(K\|M)$. This seems to be difficult in the case of EMD.

Chang et al. [9] further discussed the indifferentiability from the random oracle for the MD scheme with prefix-free encoding. They considered compression functions consisting of a block cipher [21] and DBL compression functions of the same form we considered. Nandi [20] 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 a PRF-security 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 [5]. 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 [4].

We are also interested in seeing whether security of MDP as MAC can be proved under weaker assumptions, similar to the security of HMAC under a weaker-than-PRF assumption on the compression function [2]. After An and Bellare [1] initiated such investigations, Maurer and Sjödin [17] provided several transforms as well as a general security proof technique. As stated in [6], 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, that it is an unforgeable FIL MAC.

Recently, Bellare and Ristenpart [7] further considered several hash function constructions in the dedicated-key setting, and provided a multi-property-preservation oriented treatment of them.

Organization of the paper. In Section 2, we provide basic definitions of PRFs, RKA-secure PRFs, indifferentiability, and unforgeability. We also fix notational conventions in this section. In Section 3, we formally define the MDP construction. In Section 4, we analyze the security of MDP. Section 4 consists of three parts; first, we prove that MDP is indifferentiable from a random oracle, and then prove that MDP gives a secure PRF under necessary assumptions. And we prove that MDP yields a secure MAC under a weaker-than-PRF assumption. In Section 5, we focus on the indifferentiability of MDP based on two specific types of compression function: one is a DBL compression function and the other is the Davies-Meyer compression function composed of a block cipher. Detailed proofs for several lemmas and theorems in Section 4 are described in the full version of this paper [13].

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'll use $F_K(x)$ as 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

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

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 [5] 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 a 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 a very weak adversary 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'll 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

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

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 π-related-key attack is the only kind of related-key attacks that we consider in this paper, sometimes we'll abuse the terminology and call F simply as a RKA-secure PRF.

Indifferentiability. We use the indifferentiability framework [16, 11] to assess the security of the MDP. Consider a cryptosystem $C = C(\mathcal{F})$ with oracle access to an ideal primitive $\mathcal{F}$. Also consider an ideal primitive $\mathcal{H}$ and a simulator $S = S(\mathcal{H})$ which has oracle access to $\mathcal{H}$. C is supposed to be a 'construction' involving $\mathcal{F}$. For example, $\mathcal{F}$ could be a FIL random oracle, and C 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 differentiability advantage of A against C with respect to S as:

$$\mathrm{Adv}^{\mathrm{diff}}_{C,S}(A) = \mathsf{Pr}\left[A(C(\mathcal{F}), \mathcal{F}) \Rightarrow 1\right] - \mathsf{Pr}\left[A(\mathcal{H}, S(\mathcal{H})) \Rightarrow 1\right] \ .$$

Informally, we say that $C(\mathcal{F})$ is indifferentiable from $\mathcal{H}$ if there exists a simulator $S(\mathcal{H})$ so that no efficient adversary A can have any significant differentiability 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. The MAC-advantage of a forger A over F is

$$\mathrm{Adv}^{\mathrm{mac}}_{F}(A) = \mathsf{Pr}\big[A(F_K, \mathsf{Vf}_{F_K}) \text{ forges} \,|\, K \xleftarrow{\$} \mathcal{K}\big] \ .$$

A forger A queries to the oracle $F_K(\cdot)$ for adaptively chosen messages and learns the corresponding tag values. It then returns a forgery (M, τ). The forger A is considered successful if it makes a verification query (M, τ) to the oracle $\mathsf{Vf}_{F_K}(\cdot,\cdot)$, and confirms that $F_K(M) = \tau$ but M was not queried to $F_K(\cdot)$. We refer to a forger A of this kind as a (t, q, l, ϵ)-forger if $\mathrm{Adv}^{\mathrm{mac}}_{F}(A) \geq \epsilon$, where t, q and l are upper bounds on the running time, the number of messages, and the maximal length (in bits) of each oracle query including the forgery message M, respectively. Informally, a MAC is considered secure against existential forgery under an adaptive chosen-message attack, if there is no (t, q, l, ϵ)-forger, even for very high values of t, q and l, and very small values of ϵ.

Notation. Let b be the size of the message blocks, and c the size of the chaining variables. As usually is in popular hash functions, we assume that $c \leq b$. Then the compression function $F(s, x)$ has the following form:

$$F : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c.$$

Let $\mathcal{C} = \{0,1\}^c$ and $\mathcal{B} = \{0,1\}^b$ to abbreviate the above as $F : \mathcal{C} \times \mathcal{B} \to \mathcal{C}$.

We denote by $M_1 \| M_2$ the concatenation of bitstrings M_1 and M_2. We will often abbreviate $M_1 \| M_2 \| \cdots \| M_k$ simply as $M_1 M_2 \cdots M_k$. 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's define $\mathcal{B}^* = \cup_{i=0}^{\infty} \mathcal{B}^i$, $\mathcal{B}^+ = \cup_{i=1}^{\infty} \mathcal{B}^i$, and $\mathcal{B}^{\leq k} = \cup_{i=1}^{k} \mathcal{B}^i$.

We will process messages block by block. The notation $M_1 M_2 \cdots M_k \leftarrow \mathsf{parse}(M)$ will mean that $M = M_1 \| M_2 \| \cdots \| M_k$ and $|M_i| = b$ for all $i = 1, \ldots, k-1$, and $|M_k| \leq b$. We denote by $s \xleftarrow{\$} S$ the operation of selecting a random element from S (the uniform probability distribution over S is assumed).

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 proof.

3 The MDP Construction

Given $F : \mathcal{C} \times \mathcal{B} \rightarrow \mathcal{C}$, we define $F^* : \mathcal{C} \times \mathcal{B}^* \rightarrow \mathcal{C}$ as follows:

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

for $M = M_1 M_2 \cdots M_k$ ($M_i \in \mathcal{B}$ for all i). This is the plain Merkle-Damgård iteration of F. Now we define $F_\pi^\circ : \mathcal{C} \times \mathcal{B}^+ \rightarrow \mathcal{C}$ as follows:

$$F_\pi^\circ(s, M_1 M_2 \cdots M_k) \stackrel{\text{def}}{=} F(\pi(F^*(s, M_1 \cdots M_{k-1})), M_k).$$

where π is a permutation applied right before the last iteration. π is a fixed permutation given as a parameter of the definition. We require both π and π^{-1} to be efficiently computable. Often we omit π from the notation F_π° and simply write F°.

The domain of F° is $\mathcal{B}^+ = \cup_{i=1}^{\infty} \mathcal{B}^i = \cup_{i=1}^{\infty} \{0,1\}^{bi}$. In order to let MDP process messages of arbitrary lengths (up to 2^l, for some large number l satisfying $0 < l \leq b$), we have to use a padding function $\mathsf{pad} : \cup_{i=0}^{2^l} \{0,1\}^i \rightarrow \mathcal{B}^+$ with the following property: the last block of $\mathsf{pad}(M)$ encodes the l-bit representation of the length $|M|$ of M. For example, the SHA-1's padding rule could be used.

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

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

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 the MDP is exactly the same as the Strengthened MD (SMD).

More precisely, let's 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

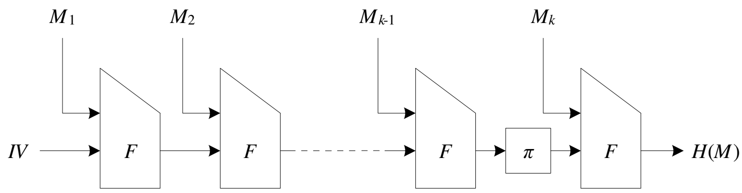

Fig. 1. The structure of MDP

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 non-negative solution to the equation $\ell+1+k \equiv b-l \pmod{b}$. Then append the l-bit representation of the number ℓ. In case of SHA-1, we have $b = 512$, and $l = 64$. Then for MDP (and SMD), the following holds:

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

For comparison, this is slightly better than the efficiency of EMD; for EMD the following holds:

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

Concretely, if we take the parameters of SHA-1, that is, $b = 512$, $c = 160$, and $l = 64$, then for messages of length between 288 and 447, EMD needs one more invocation than MDP. On the average, EMD needs c/b more invocations of the compression function than MDP. Again with the parameters of SHA-1, $c/b \approx 0.31$.

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.

4.1 Collision Resistance

First, MDP is collision-resistant. 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 random oracle $\mathcal{H}$, when a FIL random oracle $\mathcal{F}$ is used as the compression function. Therefore we need a simulator $S_{\mathcal{F}}$ so that no efficient adversary can distinguish (or rather, differentiate) the pair $(\mathsf{MDP}[\mathcal{F}, \pi], \mathcal{F})$ from the pair $(\mathcal{H}, S_{\mathcal{F}})$. We will use the simulator illustrated in Figure 2.

<u>Initialize:</u>
  $\mathcal{V} \leftarrow \mathcal{S} \leftarrow \{IV\}$

<u>Interface $\mathcal{F}(s, x)$:</u>
$\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
if $F(s, x) = \perp$ **then**
  if $s \in \mathcal{S}$ **then**
    $t \xleftarrow{\$} \mathcal{C} \setminus \big(\mathcal{V} \cup \pi(\mathcal{S}) \cup \pi^{-1}(\mathcal{V}) \cup P_\pi\big)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$
    $F(s, x) \leftarrow t$
  else if $\pi^{-1}(s) \in \mathcal{S}$ **then**
    $F(s, x) \leftarrow \mathcal{H}(M \| x)$, where $F^*(IV, M) = \pi^{-1}(s)$
  else
    $F(s, x) \xleftarrow{\$} \mathcal{C}$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{F(s, x)\}$
return $F(s, x)$

Fig. 2. Pseudocode for the simulator $S_{\mathcal{F}}$

$S_{\mathcal{F}}$ maintains a structure $F(s, x)$ where it stores previously selected value of the query $S_{\mathcal{F}}(s, x)$. Initially $F(s, x) = \perp$ for all s and x, where $\perp$ means undefined. $S_{\mathcal{F}}$ also maintains two sets $\mathcal{V}$ and $\mathcal{S}$. Both are initially set to the singleton set $\{IV\}$. As more queries are inquired, new elements are added to the sets. Note that elements never leave the sets.

When queried $S_{\mathcal{F}}(s, x)$, if $F(s, x) = \perp$, $S_{\mathcal{F}}$ will choose a value t randomly depending on the algorithm in Figure 2, and define $F(s, x) \leftarrow t$. If we consider the labeled directed graph G whose edges are $s \xrightarrow{x} F(s, x)$ for all $F(s, x) \neq \perp$, then we can see that $\mathcal{V}$ denotes the set of all vertices of G. On the other hand, $\mathcal{S}$ is then the set of all vertices that can be reached by following a path from the vertex IV. In order to prove the indifferentiability of MDP, we need a few lemmas about the simulator $S_{\mathcal{F}}$:

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

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

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

Lemmas 1 and 2 essentially say that the subgraph $\mathcal{S}$ of $\mathcal{V}$ is in fact a rooted tree with IV as the root. Note that, because these three lemmas are only about the subgraph $\mathcal{S}$, as long as the lines 102 to 105 are intact, the lines 106 to 109 do not change the validity of the lemmas. Also, due to Lemma 1 and 2, the line 107 in the pseudocode in Figure 2 works correctly. We will omit the proofs of the three lemmas since they are straightforward.

The basic intuition involved in the pseudocode of $S_{\mathcal{F}}$ 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 the MDP. But 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 the MDP. But similarly 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 doesn't have access to the adversarial choice of P.

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 $S_{\mathcal{F}}$ is designed so that 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 $\mathcal{S}$ at line 103. Note that by birthday attack, the attacker can eventually find the message pair M, M' satisfying $F^*(IV, M)$ equals $F^*(IV, M')$ or $\pi(F^*(IV, M'))$. Therefore, MDP can be indifferentiable from a random oracle only up to the birthday bound.[4]

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

Theorem 1. *Let A be an adversary distinguishing the pairs $(\mathsf{MDP}[\mathcal{F}, \pi], \mathcal{F})$ and $(\mathcal{H}, S_{\mathcal{F}})$, where the simulator $S_{\mathcal{F}}$ 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 $IV \notin P_\pi$. Then,*

$$\mathrm{Adv}^{\mathsf{diff}}_{\mathsf{MDP}[\mathcal{F},\pi],S_{\mathcal{F}}}(A) \leq \frac{5(lq_V + q_F)(3lq_V + q_F + 1)}{2^{c+1}} + \frac{lq_V q_F}{2^c} + \frac{|P_\pi|(2lq_V + q_F)}{2^c} \ ,$$

[4] MDP, being random-oracle indifferentiable, prevents the extension property. But once a colliding message pair due to an internal MD collision is found, for example by birthday attack, or by insecurity of the compression function, any common suffix can be added to the message pair. This serious effect of extension attacks is not resolved by MDP (nor by other similarly proposed composition schemes).

where q_F is the number of queries to the FIL oracle, and q_V the number of queries to the VIL oracle. l is the maximum number of message blocks for each VIL query. c is the size of the chaining variables. Moreover, $S_{\mathcal{F}}$ makes at most q_F queries and runs in time $O({q_F}^2)$.

4.3 MDP Yields Secure PRFs

In this section, we show that when the compression function F is a PRF secure against 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[5].

- 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.

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 [6]. 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.

Related-Key Multi-Oracles In order to prove the security of the two MAC schemes, first we need to introduce the notion of multi-oracle distinguishers. This was first given in [4] in order to prove that, if the MD scheme is keyed via IV, then the resulting iterated construction is 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 multi-oracle 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. But, 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 $2m$-tuple of independent random functions.

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

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

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

Lemma 4 (Related-Key Multi-Oracle Lemma). *Suppose that A is a distinguisher with access to $2m$ oracles $g_1, g'_1, \ldots, g_m, g'_m$ as above, and 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

$$t + O(q \cdot \mathrm{Time}(F) + qb \log q + qc).$$

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 multi-oracles:

Lemma 5 (Reduction to the Related-Key Multi-Oracle). *Let A be a PRF-adversary against* KMDP*. Suppose that A has time-complexity at most t, and makes at most q queries, and each query has the length at most l. Then we can construct a related-key multi-oracle distinguisher $B(g_1, g'_1, \ldots, g_q, g'_q)$ with access to $2q$ oracles so that the following holds:*

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

B makes at most q queries, and the running time of B is bounded by

$$t + O(q((l-1)(b \log q + \mathrm{Time}(F)) + c)).$$

Combining Lemma 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, and makes at most q queries, and each query has the length at most l. Then we can construct an adversary $B(g, g')$ against the π-RKA-secure PRF F such that*

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

B makes at most q queries, and the running time of B is bounded by

$$t + O(lq(b \log q + \mathrm{Time}(F) + c)).$$

Security of Prefix-MDP We prove the security of the Prefix-MDP scheme by lifting the security proof for the Keyed-MDP. Remember 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 $B_{\bar{F}}(g)$ against the dual PRF $\bar{F}$ such that*

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

Furthermore, $B_{\bar{F}}$ has time complexity at most t, and makes only 1 oracle query.

Theorem 3 (PRF-Security of Prefix-MDP). *Let A be a PRF-adversary against* PMDP*. Suppose that A has time-complexity at most t, and makes at most q queries, and each query has the length at most l. Then we can construct an adversary $B_F(g, g')$ against the π-RKA-secure PRF F, and a PRF-adversary $B_{\bar{F}}(g)$ against the dual PRF $\bar{F}$ so that*

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

Furthermore, $B_{\bar{F}}$ has time complexity at most t, and makes only 1 oracle 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 [10] exhibited a related-key distinguishing attack on the keyed MD5 compression function using pseudo-collisions of MD5 [8]. This attack shows that the keyed MD5 compression function is not a good π-RKA-secure PRF, when $\pi(x) = x \oplus \Delta$.

Remark 3. Kim et al. [15], and also Contini and Yin [10], 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 section. 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 $\mathcal{C}$ with no fixed point. Let A be a (t, q, l, ϵ)-forger of* $\mathsf{MDP}[F, \pi]$*. Then we can construct a (t', q', l', ϵ')-forger B attacking the FIL MAC F, where $q' = qN(l) + N(l) - 1$, $l' = b + c$, and $\epsilon' = 2\epsilon/(3q'^2 + 3q' + 2)$. Also, the running time t' is essentially that of A with some small overhead that is obvious from the construction of B [17].*

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 smaller 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 $P_{\pi,p} = \{u \mid u \in \mathcal{C}, \text{ and } \pi(u) = u \text{ or } p(u)\}$. Let A be an adversary distinguishing the pairs $(\mathsf{MDP}[\mathcal{F},\pi], \mathcal{F})$ and $(\mathcal{H}, S_{\mathcal{F}})$, where the simulator $S_{\mathcal{F}}$ is defined in Fig. 3. Suppose that $IV \notin P_{\pi,p}$. Then,*

$$\mathbf{Adv}^{\mathrm{diff}}_{\mathsf{MDP}[\mathcal{F},\pi],S_{\mathcal{F}}}(A) \leq \frac{7(lq_V + q_F)(3lq_V + q_F + 1) + lq_V q_F + |P_{\pi,p}|(2lq_V + q_F)}{2^c} ,$$

where q_F is the number of queries to the FIL oracle, and q_V the number of queries to the VIL oracle. l is the maximum number of message blocks for each VIL query. c is the size of the chaining variables. $S_{\mathcal{F}}$ makes at most q_F queries and runs in time $O({q_F}^2)$.

In Theorem 5, a simulator is prepared for F instead of f. 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. Thus, as is shown in Fig. 3, $S_{\mathcal{F}}$ can randomly select an output of F for each query.

5.2 MDP with the Davies-Meyer Compression Function

In this section, we consider the case that F is the Davies-Meyer compression function [18] composed of a block cipher. We show that $\mathsf{MDP}[\mathcal{F},\pi]$ is indifferentiable from a VIL random oracle if the underlying block cipher is ideal.

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, $E(K,\cdot) = E_K(\cdot)$ is a permutation for every $K \in \mathcal{B}$, and $D(K,\cdot) = D_K(\cdot) = {E_K}^{-1}(\cdot)$. E is called an ideal cipher if E_K is a truly random permutation for every $K \in \mathcal{B}$.

Initialize:

$\mathcal{V} \leftarrow \mathcal{S} \leftarrow \{IV\}$

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

$\mathcal{V} \leftarrow \mathcal{V} \cup \{s, p(s)\}$
if $F(s, x) = \perp$ **then**
  if $s \in \mathcal{S}$ **then**
    $t \xleftarrow{\$} \mathcal{C} \setminus \big(\mathcal{V} \cup \pi(\mathcal{S}) \cup \pi^{-1}(\mathcal{V}) \cup p(\mathcal{V}) \cup p(\pi(\mathcal{S})) \cup \pi^{-1}(p(\mathcal{V})) \cup P_{\pi,p}\big)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$
    $F(s, x) \leftarrow t$
    $F(p(s), x) \leftarrow \mathsf{swap}(t)$
  else if $p(s) \in \mathcal{S}$ **then**
    $t \xleftarrow{\$} \mathcal{C} \setminus \big(\mathcal{V} \cup \pi(\mathcal{S}) \cup \pi^{-1}(\mathcal{V}) \cup p(\mathcal{V}) \cup p(\pi(\mathcal{S})) \cup \pi^{-1}(p(\mathcal{V})) \cup P_{\pi,p}\big)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$
    $F(p(s), x) \leftarrow t$
    $F(s, x) \leftarrow \mathsf{swap}(t)$
  else if $\pi^{-1}(s) \in \mathcal{S}$ **then**
    $F(s, x) \leftarrow \mathcal{H}(M \| x)$, where $F^*(IV, M) = \pi^{-1}(s)$
    $F(p(s), x) \leftarrow \mathsf{swap}(F(s, x))$
  else if $\pi^{-1}(p(s)) \in \mathcal{S}$ **then**
    $F(p(s), x) \leftarrow \mathcal{H}(M \| x)$, where $F^*(IV, M) = \pi^{-1}(p(s))$
    $F(s, x) \leftarrow \mathsf{swap}(F(p(s), x))$
  else
    $F(s, x) \xleftarrow{\$} \mathcal{C}$
    $F(p(s), x) \leftarrow \mathsf{swap}(F(s, x))$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{F(s, x), F(p(s), x)\}$
return $F(s, x)$

Fig. 3. Pseudocode for the simulator $S_{\mathcal{F}}$. $\mathsf{swap}(t_1 \| t_2) = t_2 \| t_1$ for every $t_1, t_2 \in \{0,1\}^{c/2}$.

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 A be an adversary that asks at most q_V queries to the VIL oracle, q_{F_0} queries to the FIL encryption oracle and q_{F_1} queries to the FIL decryption oracle. Let l be the maximum number of message blocks for each VIL query. Suppose that $lq_V + q_{F_0} + q_{F_1} \le 2^{c-1}$. Then,*

$$\mathbf{Adv}^{\mathrm{diff}}_{\mathsf{MDP}[\mathcal{F},\pi], S_{\mathcal{E}}, S_{\mathcal{D}}}(A) \le \frac{13(lq_V + q_{F_0} + q_{F_1})(2lq_V + q_{F_0} + q_{F_1}) + |P_\pi|(3lq_V + q_{F_0})}{2^{c+1}} ,$$

where the simulators $S_{\mathcal{E}}$ and $S_{\mathcal{D}}$ are given in Fig. 4. $S_{\mathcal{E}}$ is a simulator for the encryption oracle, and $S_{\mathcal{D}}$ for the decryption oracle. $S_{\mathcal{E}}$ makes at most q_{F_0} queries and runs in time $O(q_{F_0}(q_{F_0} + q_{F_1}))$. $S_{\mathcal{D}}$ makes at most $q_{F_0} \cdot q_{F_1}$ queries and runs in time $O(q_{F_1}(q_{F_0} + q_{F_1}))$.

<u>Initialize:</u>

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

<u>Interface $\mathcal{E}(x, s)$:</u>

$\mathcal{V} \leftarrow \mathcal{V} \cup \{s\}$
if $E_x(s) = \perp$ **then**
  if $s \in \mathcal{S}$ **then**
    $E_x(s) \xleftarrow{\$} \mathcal{Q}(x) \setminus \mathbf{S}_{\text{bad}}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{E_x(s) \oplus s\}$
  else if $\pi^{-1}(s) \in \mathcal{S}$ **then**
    $u \leftarrow \mathcal{H}(M\|x) \oplus s$, where $F^*(IV, M) = \pi^{-1}(s)$
    if $u \notin \mathcal{Q}(x)$ **then**
      return `fail`
    else
      $E_x(s) \leftarrow u$
  else
    $E_x(s) \xleftarrow{\$} \mathcal{Q}(x)$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{E_x(s) \oplus s\}$
  $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{s\}$
  $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{E_x(s)\}$
return $E_x(s)$

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

if $D_x(u) = \perp$ **then**
  for every $s \in \mathcal{S}$ **do**
    if $u \oplus H(M\|x) = \pi(s)$ **then**
      $\mathbf{N} \leftarrow \mathbf{N} \cup \{s\}$, where $F^*(IV, M) = s$
  if $|\mathbf{N}| \geq 2$ **then**
    return `fail`
  else if $|\mathbf{N}| = 1$ **then**
    if $\pi(s) \notin \mathcal{P}(x)$ **then**
      return `fail`
    else
      $D_x(u) \leftarrow \pi(s)$
  else
    $D_x(u) \xleftarrow{\$} \mathcal{P}(x) \setminus (\mathcal{S} \cup \pi(\mathcal{S}))$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{D_x(u), D_x(u) \oplus u\}$
  $\mathcal{P}(x) \leftarrow \mathcal{P}(x) \setminus \{D_x(u)\}$
  $\mathcal{Q}(x) \leftarrow \mathcal{Q}(x) \setminus \{u\}$
return $D_x(u)$

Fig. 4. Pseudocode for the simulator $S_{\mathcal{E}}$ and $S_{\mathcal{D}}$. $\mathbf{S}_{\text{bad}} = \{u \oplus s \mid u \in \mathcal{V} \cup \pi(\mathcal{S}) \cup \pi^{-1}(\mathcal{V}) \cup P_\pi\}$.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The first author was supported in part by International Communications Foundation (ICF).

References

1. J.H. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. *Advances in Cryptology - CRYPTO'99*, LNCS 1666, pp. 252–269, 1999.
2. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. *Advances in Cryptology - CRYPTO 2006*, LNCS 4117, pp. 602–619, 2006.
3. M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology - CRYPTO'96*, LNCS 1109, pp. 1–15, 1996.
4. M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security, *Proc. of FOCS'96*, pp. 514–523, 1996.

5. M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. *Advances in Cryptology - EUROCRYPT 2003*, LNCS 2656, pp. 491–506, 2003.
6. M. Bellare and T. Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. *Advances in Cryptology - ASIACRYPT 2006*, LNCS 4284, pp. 299–314, 2006.
7. M. Bellare and T. Ristenpart. Hash functions in the dedicated-key setting: Design choices and MPP transforms. *Automata, Languages and Programming - ICALP 2007*, LNCS 4596, pp. 399–410, 2007.
8. B. den Boer and A. Mosselaers. Collisions for the compression function of MD5. *Advances in Cryptology - EUROCRYPT'93*, LNCS 765, pp. 293–304, 1994.
9. D. Chang, S. Lee, M. Nandi and M. Yung. Indifferentiable security analysis of popular hash function with prefix-free padding. *Advances in Cryptology - ASIACRYPT 2006*, LNCS 4284, pp. 283–298, 2006.
10. S. Contini and Y.L. Yin. Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. *Advances in Cryptology - ASIACRYPT 2006*, LNCS 4284, pp. 37–53, 2006.
11. J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. *Advances in Cryptology - CRYPTO 2005*, LNCS 3621, pp. 430–448, 2005.
12. I. Damgård. A design principle for hash functions. *Advances in Cryptology - CRYPTO'89*, LNCS 435, pp. 416–427, 1989.
13. S. Hirose, J.H. Park and A. Yun. A simple variant of the Merkle-Damgård scheme with a permutation. Full version of this paper.
14. J. Kelsey, in *Public Comments on the Draft Federal Information Processing Standard (FIPS) Draft FIPS 180-2, Secure Hash Standard (SHS)*, `http://csrc.nist.gov/CryptoToolkit/shs/dfips-180-2-comments1.pdf`, 2001.
15. J. Kim, A. Biryukov, B. Preneel and S. Lee. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1. *SCN 2006*. Also available at `http://eprint.iacr.org/2006/187`.
16. U. M. Maurer, R. Renner and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. *Theory of Cryptography - TCC 2004*, LNCS 2951, pp. 21–39, 2004.
17. U. Maurer and J. Sjödin. Single-key AIL-MACs from any FIL-MAC. *Automata, Languages and Programming - ICALP 2005*, LNCS 3580, pp. 472–484, 2005.
18. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
19. R. Merkle. One way hash functions and DES. *Advances in Cryptology, - CRYPTO'89*, LNCS 435, pp. 428–446, 1989.
20. M. Nandi. Towards optimal double-length hash functions. *Progress in Cryptology - INDOCRYPT 2005*, LNCS 3797, pp. 77–89, 2005.
21. B. Preneel, R. Govaerts and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. *Advances in Cryptology - CRYPTO'93*, LNCS 773, pp. 368–378, 1994.
22. G. Tsudik. Message authentication with one-way hash functions. *ACM Computer Communications Review*, vol. 22(5): 29–38 (1992).