Compression Functions Suitable for the Multi-Property-Preserving Transform

Hidenori Kuwakado and Masakatu Morii

Graduate School of Engineering, Kobe University
1-1 Rokkodai-cho Nada-ku Kobe 657-8501, Japan

Abstract. Since Bellare and Ristenpart showed a multi-property preserving domain extension transform, the problem of the construction for multi-property hash functions has been reduced to that of the construction for multi-property compression functions. However, the Davies-Meyer compression function that is widely used for standard hash functions is not a multi-property compression function. That is, in the ideal cipher model, the Davies-Meyer compression function is collision resistant, but it is not indifferentiable from a random oracle. In this paper, we show that the compression function proposed by Lai and Massey is a multi-property compression function. In addition, we show that the simplified version of the Lai-Massey compression function is also a multi-property compression function. The use of these compression functions enables us to construct multi-property hash functions by the multi-property preserving domain extension transform.

1 Introduction

Cryptographic hash functions play a fundamental role in modern cryptographic protocols. Hash functions are used for data integrity in conjunction with digital signatures and message authentication codes. These applications require that hash functions satisfy the following properties: preimage resistance, second-preimage resistance, and collision resistance. Another application of hash functions is an alternative to a random oracle. For example, hash functions are used to instantiate random oracles in public-key schemes such as RSA-OAEP [2] and RSA-PSS [3]. This application requires that hash functions are indistinguishable from random oracles.

Coron, Dodis, Malinaud, and Puniya [8] have formally discussed the *indifferentiability* of hash functions. The notion of indifferentiability was first introduced by Maurer, Renner, and Holenstein [12], and is a stronger notion than just indistinguishability. Coron et al. have shown that the Merkle-Damgård construction [9][13] is not indifferentiable from the random oracle, and have proposed hash-function constructions that are indifferentiable from the random oracle. Chang, Lee, Nandi, and Yung [7] have given the formal proof of indifferentiability to the constructions of Coron et al. In [7] and [8], the collision resistance of the indifferentiable constructions were not explicitly studied.

Bellare and Ristenpart [1] have shown that the indifferentiability from the random oracle does not guarantee the collision resistance, and have proposed a multi-property preserving domain extension transform (called an *MPP transform*) where "multi-property" means indifferentiability and collision resistance. The MPP transform enables a constructed hash function to inherit these properties of an underlying compression function. Due to their works, the problem of the construction for multi-property hash functions was reduced to that of the construction for multi-property compression functions.

However, the Davies-Meyer compression function, which is used for popular hash functions, is not a multi-property compression function in the ideal cipher model. Namely, the Davies-Meyer compression function is collision resistant [6], but it is not indifferentiable from a random oracle [7][8][10]. Therefore, it is important to construct a multi-property compression function.

In this paper, we show that the compression function proposed by Lai and Massey (called an *LM compression function*) [11] is a multi-property compression function. We first quantify the indifferentiability between the LM compression function and the random oracle. There are two proof methodologies for quantifying the indifferentiability. One is a methodology by Bellare and Rogaway (a *game-playing proof*) [4], the other is a methodology by Chang, Lee, Nandi, and Yung [7]. To see the difference between the two methodologies, let us consider the indifferentiability of two oracles. In the methodology by Chang et al., an event must be carefully defined so that the adversary views of two

oracles are identically distributed when the event does not occur. However, how to define the event is not necessarily obvious. On the other hand, the game-playing proof provides how to define an event for distinguishing the two oracles, which is called *identical-until-bad*. Since the notion of identical-until-bad is easy to use, we quantitatively evaluate the indifferentiability using the game-playing framework. We next quantify the collision resistance of the LM compression function because Lai and Massey did not give the formal proof of collision resistance.

We also propose the simplified version of the LM compression function, called a *CP compression function* where "CP" is an abbreviation of "Constant Plaintext." Although we do not think that the CP compression function is novel, the CP compression function have not been studied in terms of indifferentiability and collision resistance. We show that the CP compression function as well as the LM compression function is a multi-property compression function. Therefore, these compression functions are promising primitives for building multi-property hash functions.

Related Works Since the Merkle-Damgård construction is a collision-resistant preserving domain extension transform, the construction of collision-resistant compression functions have attracted interest. Since the advent of Coron et al.'s paper [8], the indifferentiability has been focused. We here summarize related works from the viewpoint of the construction for rate-1 and single-length compression functions.

Lai and Massey [11] proposed a compression function, which is studied in this paper because they did not provide any security observation. The LM compression function is based on the block cipher such that the key length is longer than the block length. Since the LM compression function requires one invocation of the block cipher and the output length is equal to the block length of the block cipher, the LM compression function is a rate-1 and single-length compression function. Parenthetically, they also proposed the different type of compression functions in [11], but the different type of compression functions are out of scope of this paper.

Preneel, Govaerts, and Vandewalle [15] analyzed the security of 64 compression functions (*PGV compression functions*) in context of attacks, but did not provide any formal proof. The PGV compression functions include popular compression functions such as the Davies-Meyer compression function, the Matyas-Meyer-Oseas compression function, and the Miyaguchi-Preneel compression function. Notice that the PGV compression functions do not include the LM compression function.

Black, Rogaway, and Shrimpton [6] provided a formal and quantitative treatment of all the PGV compression functions. Their proof is based on the ideal cipher model. They studied the collision resistance and the inversion resistance of the PGV compression functions, but did not study indifferentiability from a random oracle.

In [8], the Davies-Meyer compression function is not indifferentiable from a random oracle in the ideal cipher model. In [7][10], the PGV compression functions are not indifferentiable from a random oracle in the ideal cipher model. In [10], a compression function such that many block ciphers are used selectively was proposed, and it was stated that the proposed compression function was implemented by the LM compression function. However, the difference between the proposed compression function and the LM compression function was not discussed.

The above related works as well as this paper are based on the ideal-cipher model. Black [5] pointed out suspicion as to the wisdom of blindly using the ideal-cipher model in proofs of security. Black showed that, given a collision-resistant hash function in the ideal cipher model, there exists a block cipher that makes the hash function collision-easy. However, as described in [5], a pseudo-random permutation that is a weaker assumption than the ideal cipher model is insufficient for building a collision-resistant hash function. In fact, it is easy to prove that the LM compression function is not collision resistant under the pseudo-random-permutation assumption. Therefore, we employ the ideal cipher model in this paper.

Organization In Section 2, we describe notation, primitives, and definitions of the LM compression function and the CP compression function. Our discussion is based on the ideal cipher model. In Section 3, we first quantitatively argue the indifferentiability between the LM compression function and a random oracle. We next discuss the collision resistance of the LM compression function in a similar way to that of Black et al.[6] In Section 4, we quantify the indifferentiability and the collision

resistance of the CP compression function in a similar way to Section 3. In Section 5, we summarize remarks.

2 Preliminaries

2.1 Notation and Primitives

We will write $a \leftarrow b$ to mean that a is to be set to the result of evaluating expression b, and write $a \xleftarrow{\$} \mathcal{A}$ to mean that a is uniformly chosen at random from a finite set $\mathcal{A}$. For algorithms A and B, A^B means that A uses B as an oracle. We denote by $\Pr[A \Rightarrow a]$ the probability that an algorithm A outputs a. In addition, we denote by $\Pr[a : b]$ the probability that a predicate b is true after a was performed. We denote by $\Pr[b \mid a]$ the probability that b is true when a occurred. We let $\|$ denote the concatenation operator on strings.

Let R be a function from a finite set $\mathcal{X}$ to a finite set $\mathcal{Y}$. The function R is said to be a random oracle[1] if R satisfies the following equation for $x \notin \{x_1, x_2, \ldots, x_q\}$ and $y \in \mathcal{Y}$.

$$\Pr[R(x) = y \mid x \neq x_i \wedge R(x_i) = y_i \text{ for } i = 1, 2, \ldots, q] = \frac{1}{|\mathcal{Y}|}$$

where $|\mathcal{Y}|$ is the number of elements in $\mathcal{Y}$. Notice that R returns the same string for the same query.

When $\mathcal{Y} = \{0,1\}^n$, the random oracle R can be emulated by the algorithm of Fig. 1. In Fig. 1, the table $R[x]$ is initialized to the special symbol $\perp$, and is used for storing responses to previous queries. As queries are made, each $R[x]$ is filled with an n-bit random string.

A block cipher is a function E' from $\{0,1\}^\ell \times \{0,1\}^n$ to $\{0,1\}^n$ where, for each $k \in \{0,1\}^\ell$, $E'(k,\cdot)$ is a permutation on $\{0,1\}^n$. When E' is a block cipher, E'^{-1} denotes its inverse, i.e., $E'^{-1}(k,y)$ gives the string x such that $E'(k,x) = y$. Let $\mathsf{Bloc}(\ell, n)$ be the set of all block ciphers from $\{0,1\}^\ell \times \{0,1\}^n$ to $\{0,1\}^n$. Choosing a random element of $\mathsf{Bloc}(\ell, n)$ means that for each $k \in \{0,1\}^\ell$ one chooses a random permutation $E'(k,\cdot)$ [6]. An *ideal cipher* is defined as a random element of $\mathsf{Bloc}(\ell, n)$. Accordingly, the ideal cipher E' satisfies the following equation for each k.

$$\Pr[E'(k,x) = y \mid E'(k,x_i) = y_i \text{ for } i = 1, 2, \ldots, q] = \frac{1}{|Y| - q},$$

where each x_i is distinct, $x \notin \{x_1, x_2, \ldots, x_q\}$, and $y \notin \{y_1, y_2, \ldots, y_q\}$. Since the ideal cipher model allows an adversary to have access to both of E' and E'^{-1}, combining them simplifies description of discussion. We will use $E(1,\cdot,\cdot)$ and $E(-1,\cdot,\cdot)$ instead of E' and E'^{-1} here.

The ideal cipher E of $\mathsf{Bloc}(\ell, n)$ can be emulated by the algorithm of Fig. 2. In Fig. 2, E takes three inputs; $\alpha \in \{1, -1\}$ specifies encryption or decryption, k is an ℓ-bit key, and if $\alpha = 1$ w is an n-bit plaintext, otherwise w is an n-bit ciphertext. The double dash, //, begins a comment that extends to the end of the line. The table $E[k][x]$ is initialized with the special symbol $\perp$, and stores a ciphertext y obtained by encrypting the plaintext x with the key k. The symbol $\mathcal{Y}(E[k])$ denotes a current set of all ciphertexts y defined with the key k, and $\overline{\mathcal{Y}(E[k])}$ denotes the complement of $\mathcal{Y}(E[k])$ relative to $\{0,1\}^n$. Similarly, $\mathcal{X}(E[k])$ denotes a current set of all plaintexts x defined with the key k, and $\overline{\mathcal{X}(E[k])}$ denotes its complement set. As queries are made, each $E[k][x]$ is filled with an n-bit random string.

2.2 Definition of Compression Functions

In this paper, we first analyze security of the compression function that was proposed by Lai and Massey [11] (called an *LM compression function*). Although they proposed it, they did not discuss its security. Our purpose is to show that the LM compression function has good properties (exactly, indifferentiability and collision resistance). If the LM compression function has the good properties, then the MPP transform [1] allows us to construct a hash function with the good properties.

[1] In [1], this is the definition of a random function, and a random oracle is defined as a public random function. In this paper we treat only a public random function.

Random oracle $R(x)$
100 if $R[x] = \perp$ then
101 $R[x] \xleftarrow{\$} \{0,1\}^n$
102 return $R[x]$

Fig. 1. Random oracle R.

Ideal cipher $E(\alpha, k, w)$
200 if $\alpha = 1$ then // encryption
201 $x \leftarrow w$
202 if $E[k][x] = \perp$ then
203 $E[k][x] \xleftarrow{\$} \overline{\mathcal{Y}(E[k])}$
204 return $E[k][x]$
205 if $\alpha = -1$ then // decryption
206 $y \leftarrow w$
207 Find x s.t. $E[k][x] = y$.
208 if no such an x then
209 $x \xleftarrow{\$} \overline{\mathcal{X}(E[k])}$
210 $E[k][x] \leftarrow y$
211 return x

Fig. 2. Ideal cipher E.

Let E be an ideal cipher in $\mathsf{Bloc}(\ell, n)$ where $\ell > n$. For $z \in \{0,1\}^{\ell-n}$ and $x \in \{0,1\}^n$, the LM compression function is defined as

$$H_{LM}(z, x) = E(1, x \parallel z, x). \tag{1}$$

In addition, we call the following function a *CP compression function* where CP stands for a Constant Plaintext.

$$H_{CP}(z, x) = E(1, x \parallel z, c), \tag{2}$$

where c is an n-bit public constant string, say 0^n. Although we do not think that the CP compression function is novel, the security of the CP compression function have not been studied formally. Our purpose is to show that the CP compression function has good properties.

Compression functions are usually classified as *rate* and *length*. A compression function H is called a rate-$1/r$ compression function if r invocations of block cipher $E(1, \cdot, \cdot)$ is necessary to compute H. A compression function H is called a single-length compression function if the output length of H is equal to the block length of E. Accordingly, the LM compression function and the CP compression function are rate-1 and single-length. Although Black et al. [6] cyclopaedically analyzed collision resistance of rate-1 and single-length compression functions, the LM compression function and the CP compression function were not included in them because Black et al. analyzed compression functions based on the block cipher such that the key length was equal to the block length.

3 The LM Compression Function

Let E be the ideal cipher of Fig. 2, i.e., a function from $\{1,-1\} \times \{0,1\}^\ell \times \{0,1\}^n$ to $\{0,1\}^n$ where an element of $\{1,-1\}$ stands for encryption or decryption, ℓ is key length, and n is block length. The LM compression function is defined as

$$H(z, x) = E(1, x \parallel z, x), \tag{3}$$

which is a function from $\{0,1\}^{\ell-n} \times \{0,1\}^n$ to $\{0,1\}^n$. In this section, we omit the subscription LM of Eq. (1) for simplification. In hash-function contractions such as the MPP transform, z is a message block to be compressed and x is output of the preceding compression function.

3.1 Indifferentiability

To evaluate the indifferentiability from a random oracle, we introduce the advantage of an adversary against the LM compression function, which is called a *pro-advantage*. The pro-advantage indicates how much the LM compression function behaves like a random oracle. The pro-advantage of an adversary A is defined as

$$\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) = \Pr\left[A^{H^E,E} \Rightarrow 1\right] - \Pr\left[A^{R,S^R} \Rightarrow 1\right], \tag{4}$$

```
The LM compression function H_0(z, x)
100  return E_0(1, x || z, x)

Ideal cipher E_0(α, k, w)
200  if α = 1 then
201    x←w; Parse k into a || z.
202    if E_0[k][x] =⊥ then
203      if a = x then
204        y <-$ {0,1}^n
205      else
206        y <-$ {0,1}^n
207      if y ∈ Y(E_0[k]) then
208        bad←true    // bad_e
209        y <-$ Y(E_0[k])
210      E_0[k][x]←y
211    return E_0[k][x]
212  if α = −1 then
213    y←w; Parse k into a || z.
214    Find x s.t. E_0[k][x] = y.
215    if no such an x then
216      x <-$ X(E_0[k])
217      if a = x then
218        bad←true    // bad_d
219      E_0[k][x]←y
220    return x
```

Fig. 3. Game $G_0 = (H_0, E_0)$.

where H is the LM compression function, E is the ideal cipher, R is the random oracle, and S is a simulator. The random oracle R exposes the same interface as H, i.e., R is a function from $\{0,1\}^{\ell-n} \times \{0,1\}^n$ to $\{0,1\}^n$. It is easy to implement R using a random oracle from $\{0,1\}^\ell$ to $\{0,1\}^n$. The simulator S exposes the same interface as E, and emulates E as possible. If the value of $\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A)$ is negligibly small, then it means that the adversary A cannot distinguish between the LM compression function H and the random oracle R.

We quantify the indifferentiability of the LM compression function using the game-playing framework [4]. We assume that A is an infinitely powerful adversary and A makes no pointless query such as the same query to oracles.

We start with a game G_0 as shown in Fig. 3. In Fig. 3, H_0 is a function from $\{0,1\}^{\ell-n} \times \{0,1\}^n$ to $\{0,1\}^n$, and E_0 is a function from $\{1,-1\} \times \{0,1\}^\ell \times \{0,1\}^n$ to $\{0,1\}^n$. In line 201 and line 213, k is parsed into an $(\ell - n)$-bit string a and an n-bit string z. The flag *bad* in line 208 and line 218, which will be used for later discussion, does not any effect on the output of E_0. It is easy to verify that H_0 and E_0 exactly emulate the LM compression function of Eq. (3) and the ideal cipher of Fig. 2, respectively. Hence, we have, for any adversary A,

$$\Pr\left[A^{H^E,E} \Rightarrow 1\right] = \Pr\left[A^{G_0} \Rightarrow 1\right].$$

Note that the description of E_0 is of redundancy (e.g., line 203–206), which is helpful to compare other games. We next consider a game G_1 as shown in Fig. 4. In Fig. 4, R_1 exposes the same interface as H_0, but R_1 is algorithmically equivalent to the random oracle R of Fig. 1. The function S_1 is a simulator that we here study, and is designed to emulate the ideal cipher as possible. It follows that

$$\Pr\left[A^{R,S^R} \Rightarrow 1\right] = \Pr\left[A^{G_1} \Rightarrow 1\right]$$

for any adversary A. Therefore, the pro-advantage of Eq. (4) is rewritten as

$$\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) = \Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right]. \tag{5}$$

We compare the game G_1 and a game G_2 of Fig. 5. The function R_2 exposes the same interface as R_1, but the algorithm of R_2 differs from that of R_1. However, if a query is fresh, then R_2 returns an n-bit random string due to line 204, line 209, and line 220 in Fig. 5. Hence, R_2 as well as R_1 is the random oracle. Comparing S_1 and S_2, we see that the difference is line 204 and line 220, i.e., $y \leftarrow R_1(z,x)$ in S_1 and $y \leftarrow \{0,1\}^n$ in S_2. Both of them return a random string if a query is fresh. It follows that S_1 and S_2 are functionally equivalent. Hence, we have

$$\Pr\left[A^{G_1} \Rightarrow 1\right] = \Pr\left[A^{G_2} \Rightarrow 1\right] \tag{6}$$

```
Random oracle R_1(z, x)
100 if R_1[x ‖ z] =⊥ then
101     R_1[x ‖ z] ←$ {0,1}^n
102 return R_1[x ‖ z]

Simulator S_1(α, k, w)
200 if α = 1 then
201     x←w; Parse k into a ‖ z.
202     if S_1[k][x] =⊥ then
203         if a = x then
204             y←R_1(z, x)
205         else
206             y ←$ {0,1}^n
207         if y ∈ 𝒴(S_1[k]) then
208             bad←true
209             if a ≠ x then
210                 y ←$ 𝒴(S_1[k])‾
211         S_1[k][x]←y
212     return S_1[k][x]
213 if α = −1 then
214     y←w; Parse k into a ‖ z.
215     Find x s.t. S_1[k][x] = y.
216     if no such an x then
217         x ←$ 𝒳(S_1[k])‾
218         if a = x then
219             bad←true
220             S_1[k][x]←R_1(z, x)
221         x ←$ 𝒳(S_1[k])‾
222         S_1[k][x]←y
223     return x
```

Fig. 4. Game $G_1 = (R_1, S_1)$.

```
Function R_2(z, x)
100 return S_2(1, x ‖ z, x)

Simulator S_2(α, k, w)
200 if α = 1 then
201     x←w; Parse k into a ‖ z.
202     if S_2[k][x] =⊥ then
203         if a = x then
204             y ←$ {0,1}^n
205         else
206             y ←$ {0,1}^n
207         if y ∈ 𝒴(S_2[k]) then
208             bad←true
209             if a ≠ x then
210                 y ←$ 𝒴(S_2[k])‾
211             S_2[k][x]←y
212     return S_2[k][x]
213 if α = −1 then
214     y←w; Parse k into a ‖ z.
215     Find x s.t. S_2[k][x] = y.
216     if no such an x then
217         x ←$ 𝒳(S_2[k])‾
218         if a = x then
219             bad←true
220             S_2[k][x]←{0,1}^n
221         x ←$ 𝒳(S_2[k])‾
222         S_2[k][x]←y
223     return x
```

Fig. 5. Game $G_2 = (R_2, S_2)$.

for any adversary A. Substituting Eq. (6) into Eq. (5) yields the following equation.

$$\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) = \Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_2} \Rightarrow 1\right]. \tag{7}$$

We compare the game G_0 with the game G_2. Since H_0 and R_2 are algorithmically the same, we focus on the difference between E_0 and S_2. We easily see that the difference appears after the statement *bad*←`true`. Namely, E_0 and S_2 are identical until the flag *bad* sets. Using the fundamental lemma [4], we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) &= \Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_2} \Rightarrow 1\right] \\ &\leq \Pr\left[G_0 \text{ sets } bad\right],\end{aligned}$$

where $\Pr[G_0 \text{ sets } bad]$ denotes the probability that the flag *bad* in Fig. 3 is set to `true` in the execution of A with the game G_0. We calculate the probability $\Pr[G_0 \text{ sets } bad]$. Since the query to H_0 turns out to be the query to E_0, we consider only the query to E_0. Since the flag *bad* appears in line 208 and line 218, we have

$$\Pr\left[G_0 \text{ sets } bad\right] \leq \Pr\left[G_0 \text{ sets } bad_e\right] + \Pr\left[G_0 \text{ sets } bad_d\right], \tag{8}$$

where $\Pr[G_0 \text{ sets } bad_e]$ and $\Pr[G_0 \text{ sets } bad_d]$ are probabilities that bad is set to `true` in line 208 and line 218, respectively. These probabilities are calculated as follows. At the i-the query to E_0, the number of defined elements in the table $E_0[k][x]$ is at most $i-1$, namely, $|\mathcal{X}(E_0[k])|$ and $|\mathcal{Y}(E_0[k])|$ are at most $i-1$. The probability that bad_e is set at the i-the query is not greater than $(i-1)/2^n$, and the probability that bad_d is set at the i-the query is not greater than $1/(2^n-(i-1))$. Assuming that q queries are made in the execution of A with the game G_0, we have

$$\begin{aligned}
\Pr[G_0 \text{ sets } bad_e] &\le \frac{1}{2^n} + \frac{2}{2^n} + \ldots + \frac{q-1}{2^n} \\
&= \frac{q(q-1)}{2^{n+1}}, \\
\Pr[G_0 \text{ sets } bad_d] &\le \frac{1}{2^n} + \frac{1}{2^n-1} + \ldots + \frac{1}{2^n-(q-1)} \\
&\le \frac{q}{2^{n-1}},
\end{aligned}$$

where we assumed that $q \le 2^{n-1}+1$. Substituting the above inequalities into Eq. (8), we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) &\le \Pr[G_0 \text{ sets } bad] \\
&\le \frac{q(q+3)}{2^{n+1}}.
\end{aligned} \tag{9}$$

3.2 Collision Resistance

In this section, we analyze the collision resistance of the LM compression function. Although Lai and Massey proposed this function, they did not evaluate its collision resistance. To quantify the difficulty of finding a collision in H, we consider the following probability, called a *col-advantage* of adversary B [6].

$$\mathbf{Adv}^{\mathrm{col}}_{H}(B) = \Pr\left[B^{H^E,E} \Rightarrow ((z,x),(z',x')) : \text{Outputs collide.}\right], \tag{10}$$

where "Outputs collide" means that one of the following events occurs.

- $(z,x) \neq (z',x') \wedge H(z,x) = H(z',x')$
- For a constant y_0 given in advance, $H(z,x) = y_0$.

Since the game G_0 $(= (H_0, E_0))$ in Fig. 3 exactly emulates H and E, the col-advantage is given by

$$\mathbf{Adv}^{\mathrm{col}}_{H}(B) = \Pr\left[B^{G_0} \Rightarrow ((z,x),(z',x')) : \text{Outputs collide.}\right].$$

Let $\mathtt{C}_i$ be the event that there exists $j \in \{1,2,\ldots,i-1\}$ such that $(z_i,x_i) \neq (z_j,x_j) \wedge H_0(z_i,x_i) = H_0(z_j,x_j)$ or $H_0(z_i,x_i) = y_0$. In the game G_0, an oracle's answer is randomly selected from a set of at least $2^n-(i-1)$ because the adversary makes no pointless query. Noticing that y_0 was given in advance, we have $\Pr[\mathtt{C}_i] \le i/(2^n-(i-1))$. Hence, we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{col}}_{H}(B) &\le \Pr[\mathtt{C}_1 \vee \mathtt{C}_2 \vee \ldots \vee \mathtt{C}_q] \\
&\le \sum_{i=1}^{q} \Pr[\mathtt{C}_i] \\
&\le \frac{q(q+1)}{2^n},
\end{aligned} \tag{11}$$

where we assumed that $q \le 2^{n-1}$.

4 The CP Compression Function

Let E be an ideal cipher from $\{1,-1\}\times\{0,1\}^{\ell}\times\{0,1\}^{n}$ to $\{0,1\}^{n}$ as shown in Fig. 2. For $z\in\{0,1\}^{\ell-n}$ and $x\in\{0,1\}^{n}$, the CP compression function is defined as

$$H(z,x)=E(1,x\parallel z,c), \tag{12}$$

where c is a public constant string, say 0^n. In this section, we omit the subscription CP of Eq. (2). In hash-function contractions, z is a message block to be compressed and x is output of the preceding compression function.

In this section, we quantify the indifferentiability and the collision resistance of the CP compression function. We will observe that these properties of the CP compression function and those of the LM compression function are the same level in terms of adversary's advantage. In other words, the encryption of variable x in the LM compression function does not contribute to improving adversary's advantage.

4.1 Indifferentiability

We quantify the indifferentiability of the CP compression function in a similar way to Section 3.1. We define the pro-advantage of an adversary A as

$$\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A)=\Pr\left[A^{H^E,E}\Rightarrow 1\right]-\Pr\left[A^{R,S^R}\Rightarrow 1\right], \tag{13}$$

where H is the CP compression function, E is the ideal cipher, R is the random oracle, and S is a simulator that we study here. We assume that A is an infinitely powerful adversary and A makes no pointless query.

We start with a game G_3 as shown in Fig. 6. In Fig. 6, H_3 and E_3 exactly emulate the CP compression function of Eq. (12) and the ideal cipher of Fig. 2, respectively. Thus, for any adversary A, we have

$$\Pr\left[A^{H^E,E}\Rightarrow 1\right]=\Pr\left[A^{G_3}\Rightarrow 1\right].$$

Note that the redundant description of E_3 is helpful to compare other games. We next consider a game G_4 as shown in Fig. 7. In Fig. 4, R_4 exposes the same interface as H_3, but R_4 is algorithmically equivalent to the random oracle R of Fig. 1. The function S_4 is a simulator that we here study. It follows that

$$\Pr\left[A^{R,S^R}\Rightarrow 1\right]=\Pr\left[A^{G_4}\Rightarrow 1\right]$$

for any adversary A. Therefore, the pro-advantage of Eq. (13) is rewritten as

$$\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A)=\Pr\left[A^{G_3}\Rightarrow 1\right]-\Pr\left[A^{G_4}\Rightarrow 1\right]. \tag{14}$$

We compare the game G_4 and a game G_5 of Fig. 8. The function R_5 exposes the same interface as R_4, and R_5 always returns an n-bit random string due to line 204, line 209, and line 220 in Fig. 7. Hence, R_4 as well as R_3 is the random oracle. Comparing S_4 and S_5, we see that line 204 and line 220 are different, but both of them return a random string if the query is fresh. Hence, S_4 and S_5 are functionally equivalent. Since G_4 and G_5 are the same for the adversary, Eq. (14) is rewritten as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A)&=\Pr\left[A^{G_3}\Rightarrow 1\right]-\Pr\left[A^{G_5}\Rightarrow 1\right]\\&\le\Pr\left[G_3\text{ sets }bad\right].\end{aligned}$$

The above inequality is based on the fact that E_3 and S_5 are identical until the flag bad sets. We can calculate the probability $\Pr[G_3\text{ sets }bad]$ in a similar way to Section 3.1.

$$\begin{aligned}\Pr\left[G_3\text{ sets }bad\right]&\le\Pr\left[G_3\text{ sets }bad_e\right]+\Pr\left[G_3\text{ sets }bad_d\right]\\&\le\frac{q(q+3)}{2^{n+1}}.\end{aligned}$$

The above bound is the same as Eq. (9). Comparing Eq. (3) and Eq. (12), we observe that encrypting variable x does not improve the upper bound of the advantage.

The CP compression function $H_3(z, x)$
100 return $E_3(1, x \parallel z, c)$

Ideal cipher $E_3(\alpha, k, w)$
200 if $\alpha = 1$ then
201 $x \leftarrow w$
202 if $E_3[k][x] = \perp$ then
203 if $x = c$ then
204 $y \xleftarrow{\$} \{0,1\}^n$
205 else
206 $y \xleftarrow{\$} \{0,1\}^n$
207 if $y \in \mathcal{Y}(E_3[k])$ then
208 $bad \leftarrow \texttt{true}$ // bad_e
209 $y \xleftarrow{\$} \overline{\mathcal{Y}(E_3[k])}$
210 $E_3[k][x] \leftarrow y$
211 return $E_3[k][x]$
212 if $\alpha = -1$ then
213 $y \leftarrow w$
214 Find x s.t. $E_3[k][x] = y$.
215 if no such an x then
216 $x \xleftarrow{\$} \overline{\mathcal{X}(E_3[k])}$
217 if $x = c$ then
218 $bad \leftarrow \texttt{true}$ // bad_d
219 $E_3[k][x] \leftarrow y$
220 return x

Fig. 6. Game $G_3 = (H_3, E_3)$.

Random oracle $R_4(z, x)$
100 if $R_4[x \parallel z] = \perp$ then
101 $R_4[x \parallel z] \xleftarrow{\$} \{0,1\}^n$
102 return $R_4[x \parallel z]$

Simulator $S_4(\alpha, k, w)$
200 if $\alpha = 1$ then
201 $x \leftarrow w$
202 if $S_4[k][x] = \perp$ then
203 if $x = c$ then
204 $y \leftarrow R_4(z, x)$
205 else
206 $y \xleftarrow{\$} \{0,1\}^n$
207 if $y \in \mathcal{Y}(S_4[k])$ then
208 $bad \leftarrow \texttt{true}$
209 if $x \neq c$ then
210 $y \xleftarrow{\$} \overline{\mathcal{Y}(S_4[k])}$
211 $S_4[k][x] \leftarrow y$
212 return $S_4[k][x]$
213 if $\alpha = -1$ then
214 $y \leftarrow w$
215 Find x s.t. $S_4[k][x] = y$.
216 if no such an x then
217 $x \xleftarrow{\$} \overline{\mathcal{X}(S_4[k])}$
218 if $x = c$ then
219 $bad \leftarrow \texttt{true}$
220 $S_4[k][x] \leftarrow R_4(z, x)$
221 $x \xleftarrow{\$} \overline{\mathcal{X}(S_4[k])}$
222 $S_4[k][x] \leftarrow y$
223 return x

Fig. 7. Game $G_4 = (R_4, S_4)$.

4.2 Collision Resistance

To quantify the difficulty of finding a collision in the CP compression function H, we define the col-advantage of adversary B as

$$\mathbf{Adv}^{\mathrm{col}}_{H}(B) = \Pr\left[B^{H^E,E} \Rightarrow ((z,x),(z',x')) : \text{Outputs collide.}\right]. \tag{15}$$

In a similar way to Section 3.2, we obtain the following bound on $\mathbf{Adv}^{\mathrm{col}}_{H}(B)$.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}}_{H}(B) &= \Pr\left[B^{G_3} \Rightarrow ((z,x),(z',x')) : \text{Outputs collide.}\right] \\ &\leq \frac{q(q+1)}{2^n},\end{aligned} \tag{16}$$

where we assumed that $q \leq 2^{n-1}$. The above bound is the same as Eq. (11).

```
Function R_5(z, x)
100  return S_5(1, x || z, c)

Simulator S_5(α, k, w)
200  if α = 1 then
201     x←w
202     if S_5[k][x] =⊥ then
203        if x = c then
204           y ←$ {0,1}^n
205        else
206           y ←$ {0,1}^n
207        if y ∈ 𝒴(S_5[k]) then
208           bad←true
209           if x ≠ c then
210              y ←$ 𝒴(S_5[k])
211           S_5[k][x]←y
212     return S_5[k][x]
213  if α = −1 then
214     y←w
215     Find x s.t. S_5[k][x] = y.
216     if no such an x then
217        x ←$ 𝒳(S_5[k])
218        if x = c then
219           bad←true
220           S_5[k][x]←{0,1}^n
221           x ←$ 𝒳(S_5[k])
222        S_5[k][x]←y
223     return x
```

Fig. 8. Game $G_5 = (R_5, S_5)$.

5 Concluding Remarks

Due to [1], the problem of building a multi-property hash function was reduced to that of building a multi-property compression function. Hence, it is significant to build a multi-property compression function from primitives. In this paper, we employed the ideal cipher as the primitive.

We have first quantified the indifferentiability and the collision resistance of the LM compression function in the ideal cipher model. In order to distinguish between the LM compression function and the random oracle, or in order to find a collision in the LM compression function, an adversary needs about $\sqrt{2^n}$ queries to oracles where n is output length. Next, we have analyzed the indifferentiability and the collision resistance of the CP compression function, which is a variant of the LM compression function. We have shown that the CP compression function has the same properties as the LM compression function in terms of adversary's advantage.

Although the Davies-Meyer compression function is widely used for popular hash functions such as the SHA family [14], the Davies-Meyer compression function is not a multi-property compression function, that is, it is distinguishable from the random oracle in the ideal cipher model. In contrast, the LM compression function and the CP compression function are multi-property compression functions. Therefore, the use of these compression functions enables us to build hash functions with the same properties by the MPP transform.

References

1. M. Bellare and T. Ristenpart, "Multi-property-preserving hash domain extension and the EMD transform," Advances in Cryptology - ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4248, pp. 299–314, 2006.
2. M. Bellare and P. Rogaway, "Optimal asymmetric encryption," Advanced in Cryptology - EUROCRYPT '94, Lecture Notes in Computer Science, vol. 950, pp. 92–111, 1994.
3. M. Bellare and P. Rogaway, "The exact security of digital signatures – how to sign with RSA and Rabin," Advances in Cryptology - EUROCRYPT '96, Lecture Notes in Computer Science, vol. 1070, pp. 399–416, 1996.
4. M. Bellare and P. Rogaway, "Code-based game-playing proofs and the security of triple encryption," Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.
5. J. Black, "The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function," Fast Software Encryption, FSE 2006, Lecture Notes in Computer Science, vol. 4047, pp. 328–340, 2006. Cryptology ePrint Archive, Report 2005/210, `http://eprint.iacr.org/`.
6. J. Black, P. Rogaway, and T. Shrimpton, "Black-box analysis of the block-cipher-based hash-function constructions from PGV," Advances in Cryptology - CRYPTO 2002, Lecture Notes in Computer Science, vol. 2442, pp. 320–335, 2002.
7. D. Chang, S. Lee, M. Nandi, and M. Yung, "Indifferentiable security analysis of popular hash functions with prefix-free padding," Advances in Cryptology - ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4284, pp. 283–298, 2006.
8. J. S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgård revisited: How to construct a hash function," Advances in Cryptology - CRYPTO 2005, Lecture Notes in Computer Science, vol. 3621, pp. 430–448, 2005.
9. I. B. Damgård, "Collision free hash functions and public key signature schemas," Advances in Cryptology - EUROCRYPT '87, Lecture Notes in Computer Science, vol. 304, pp. 203–216, 1988.
10. H. Kuwakado and M. Morii, "Indifferentiability of single-block-length and rate-1 compression functions," Cryptology ePrint Archive, Report 2006/485, 2006. `http://eprint.iacr.org/`.
11. X. Lai and J. L. Massey, "Hash functions based on block ciphers," Advances in Cryptology - EUROCRYPT '92, Lecture Notes in Computer Science, vol. 658, pp. 55–70, 1993.
12. U. Maurer, R. Renner, and C. Holenstein, "Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology," First Theory of Cryptography Conference, TCC 2004, Lecture Notes in Computer Science, vol. 2951, pp. 21–39, 2004.
13. R. C. Merkle, "One way hash functions and DES," Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science, vol. 435, pp. 428–446, 1990.
14. National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publication 180-2, August 2002. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`.
15. B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science, vol. 773, pp. 368–378, 1994.