Compression Functions Suitable for the Multi-Property-Preserving Transform

Kuwakado, Hidenori

Morii, Masakatu

(Citation)
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,E91-A(10):2851-2859

(Issue Date)
2008

(Resource Type)
journal article

(Version)
Version of Record

PAPER *Special Section on Information Theory and Its Applications*

Compression Functions Suitable for the Multi-Property-Preserving Transform*

Hidenori KUWAKADO[†a)] ***and*** **Masakatu MORII**[†], ***Members***

SUMMARY 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 commonly 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.
key words: *compression function, hash function, multi-property preserving*

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 in instantiating 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 [4] have formally discussed the *indifferentiability* of hash functions. The notion of indifferentiability was first introduced by Maurer, Renner, and Holenstein [5], and is a stronger notion than just indistinguishability. Coron et al. have shown that the Merkle-Damgård construction [6], [7] 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 [8] have given the formal proof of indifferentiability to the constructions of Coron et al. The indifferentiability from the random oracle is regarded as a formal definition of behaving like a random oracle. If a hash function is proved to be indifferentiable from a random oracle, then we can justify that the hash function is used in instantiating the random oracle. Namely, any scheme proved secure in the random oracle model (e.g., RSA-OAEP and RSA-PSS) is still secure even if such an instantiation is done. Thus, the indifferentiability of the hash function is useful in instantiating the random oracle. In [4] and [8], the collision resistance of the indifferentiable constructions was not explicitly studied.

Bellare and Ristenpart [9] 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 the *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. For example, let us consider the random-oracle property, which is one of important properties. When a compression function that is indifferentiable from a random oracle is given, a hash function that is indifferentiable from a random oracle can be constructed from the compression function by using the MPP transform. The indifferentiability of the compression function itself is useful in constructing the random-oracle-like hash function. This paper concentrates on the construction of multi-property compression functions. The domain extension such as the MPP transform and the Merkle-Damgård construction is outside of scope of this paper.

It should be noted that 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 [10], but it is not indifferentiable from a random oracle [4], [8]. Indeed, all the single-block, rate-1 compression functions (called PGV compression functions) are not indifferentiable from the random oracle [11]. 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 the *LM compression function*) [12] is a multi-property compression function in the ideal cipher model. We first quantify the indifferentiability between the LM compression function and the random oracle using the game-playing framework, which was developed by Bellare and Rogaway [13]. We will prove

Manuscript received January 22, 2008.
Manuscript revised April 9, 2008.
[†]The authors are with the Graduate School of Engineering, Kobe University, Kobe-shi, 657-8501 Japan.
*The preliminary version of this paper appeared in [1].
a) E-mail: kuwakado@kobe-u.ac.jp
DOI: 10.1093/ietfec/e91–a.10.2851

that the LM compression function is indifferentiable from the random oracle in the ideal cipher model. Notice that the PGV compression functions including the Davies-Meyer compression function are not indifferentiable from it in the ideal cipher model. 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 will prove that the LM compression function as well as the PGV function is collision resistant.

We also discuss the simplified version of the LM compression function, called the *CP compression function* where "CP" is an abbreviation of "Constant Plaintext." The CP compression function might be more efficient than the LM compression function when these compression functions are implemented with a real block cipher. For example, since the CP compression function requires only to encrypt the fixed plaintext, some steps in the real block cipher might be precomputable. Although we do not think that the CP compression function is novel, the CP compression function has 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 compression function for building multi-property hash functions.

(1) 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 [4], 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 [12] 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 [12], but the different type of compression functions are out of the 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 [10] 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, which this paper will discuss as an important property.

The indifferentiability of the PGV compression functions was studied in [4], [8] and [11]. The article [4] showed that the Davies-Meyer compression function, which was the most popular PGV compression function, was not indifferentiable from a random oracle in the ideal cipher model. The articles [8], [11] described that all the PGV compression functions were not indifferentiable from a random oracle in the ideal cipher model. These facts are motivation for this paper. In [11], 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 are based on the ideal-cipher model. Black [16] 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, it should be noted that the hash function shown by Black has an unusual structure. Since the compression functions that this paper will discuss do not have such a structure, a question such as Black's hash function does not arise as far as domain extensions such as [9] are used. On the other hand, a pseudorandom-permutation model that is weaker than the ideal cipher model is insufficient for building a collision-resistant hash function [16]. In fact, it is easy to prove that the LM compression function is not collision resistant under the pseudorandom-permutation model. As works in other models, Shrimpton and Stam [17] have studied the construction of a collision-resistant compression function based on small random functions. When this compression function is implemented, it is necessary to instantiate all the random functions. In contrast, when the LM compression function is implemented, it is sufficient to instantiate only one ideal cipher. Therefore, it is worthy to analyze the LM compression function and the CP compression function in the ideal cipher model because of the natural structure and the number of instantiation.

(2) Organization

In Sect. 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 Sect. 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. [10] In Sect. 4, we quantify the indifferentiability and the collision resistance of the CP compression function in a similar way to Sect. 3. In Sect. 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 $a \stackrel{\$}{\leftarrow} \mathcal{A}$ 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*[†] 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 response 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. Note that $R[x]$ is the input-output table of the function $R(x)$. Initializing $R[x]$ to $\perp$ means that all the values of $R(x)$ are undefined. As a query x is made, the symbol $\perp$ in $R[x]$ is replaced with an n-bit random string, which is used as the value of $R(x)$ after the query.

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 $E'(k, \cdot)$ [10]. 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.

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

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. For example, we will discuss the total number of queries to E' and E'^{-1} in later sections. 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

```
Random oracle R(x)
    if R[x] =⊥ then
        R[x] <-$ {0,1}^n
    return R[x]
```

Fig. 1 Random oracle R.

```
Ideal cipher E(α, k, w)
    if α = 1 then   // encryption
        x <- w
        if E[k][x] =⊥ then
            E[k][x] <-$ complement(Y(E[k]))
        return E[k][x]
    if α = -1 then   // decryption
        y <- w
        Find x s.t. E[k][x] = y.
        if no such an x then
            x <-$ complement(X(E[k]))
            E[k][x] <- y
        return x
```

Fig. 2 Ideal cipher E.

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. Thus, the ideal cipher E is considered as a manager of the table $E[k][x]$ for an encryption query and a decryption query[†].

2.2 Definition of Compression Functions

In this paper, we first analyze security of the compression function that was proposed by Lai and Massey [12] (called the *LM compression function*). Although they proposed it, they did not discuss its security. Our purpose is to show that the LM compression function has multi-properties (exactly, indifferentiability and collision resistance). If the LM compression function has the multi-properties, then the MPP transform [9] allows us to construct a hash function with the multi-properties.

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

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

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

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

[†]In [9], 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.

where c is an n-bit public constant string, say 0^n. Since the CP compression function requires only to encrypt the constant plaintext, the CP compression function might be more efficient than the LM compression function when E is instantiated by a real block cipher. Although we do not think that the CP compression function is novel, the security of the CP compression function has 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. [10] 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.

```
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.    // a: ℓ − n bit, z: n bits
202       if E_0[k][x] =⊥ then
203          if a = x then    // Lines 203–206: just y ←$ {0, 1}^n
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 ←$ complement(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 ←$ complement(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 $G_0 = (H_0, E_0)$.

3. The LM Compression Function

Let E be the ideal cipher of Fig. 2, i.e., the 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 subscript 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

There are two proof methodologies for quantifying the indifferentiability. One is a methodology by Bellare and Rogaway (a game-playing proof) [13], the other is a methodology by Chang, Lee, Nandi, and Yung [8]. 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.

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

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

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 [13]. 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 $\{0,1\}^n$. We can verify that H_0 and E_0 in the game G_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,

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

Compared with E in Fig. 2, E_0 in Fig. 3 seems to involve redundancy. For example, statements from line 202 to line 210 in Fig. 3 can be written as statements from line 202

Random oracle $R_1(z, x)$
100 if $R_1[x \| z] = \perp$ then
101 $R_1[x \| z] \xleftarrow{\$} \{0, 1\}^n$
102 return $R_1[x \| z]$

Simulator $S_1(\alpha, k, w)$
200 if $\alpha = 1$ then
201 $x \leftarrow w$; Parse k into $a \| z$.
202 if $S_1[k][x] = \perp$ then
203 if $a = x$ then
204 $y \leftarrow R_1(z, x)$
205 else
206 $y \xleftarrow{\$} \{0, 1\}^n$
207 if $y \in \mathcal{Y}(S_1[k])$ then
208 $bad \leftarrow$ `true`
209 if $a \neq x$ then
210 $y \xleftarrow{\$} \overline{\mathcal{Y}(S_1[k])}$
211 $S_1[k][x] \leftarrow y$
212 return $S_1[k][x]$
213 if $\alpha = -1$ then
214 $y \leftarrow w$; Parse k into $a \| z$.
215 Find x s.t. $S_1[k][x] = y$.
216 if no such an x then
217 $x \xleftarrow{\$} \overline{\mathcal{X}(S_1[k])}$
218 if $a = x$ then
219 $bad \leftarrow$ `true`
220 $S_1[k][x] \leftarrow R_1(z, x)$
221 $x \xleftarrow{\$} \overline{\mathcal{X}(S_1[k])}$
222 $S_1[k][x] \leftarrow y$
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 $S_2(\alpha, k, w)$
200 if $\alpha = 1$ then
201 $x \leftarrow w$; Parse k into $a \| z$.
202 if $S_2[k][x] = \perp$ then
203 if $a = x$ then
204 $y \xleftarrow{\$} \{0, 1\}^n$
205 else
206 $y \xleftarrow{\$} \{0, 1\}^n$
207 if $y \in \mathcal{Y}(S_2[k])$ then
208 $bad \leftarrow$ `true`
209 if $a \neq x$ then
210 $y \xleftarrow{\$} \overline{\mathcal{Y}(S_2[k])}$
211 $S_2[k][x] \leftarrow y$
212 return $S_2[k][x]$
213 if $\alpha = -1$ then
214 $y \leftarrow w$; Parse k into $a \| z$.
215 Find x s.t. $S_2[k][x] = y$.
216 if no such an x then
217 $x \xleftarrow{\$} \overline{\mathcal{X}(S_2[k])}$
218 if $a = x$ then
219 $bad \leftarrow$ `true`
220 $S_2[k][x] \leftarrow \{0, 1\}^n$
221 $x \xleftarrow{\$} \overline{\mathcal{X}(S_2[k])}$
222 $S_2[k][x] \leftarrow y$
223 return x

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

to line 203 in Fig. 2. Notice that the redundant description in Fig. 3 will be useful in later discussion. The flag *bad* in line 208 and line 218 does not have any effect on the output of E_0. However, the flag will play an important role for calculating the pro-advantage of A. In addition, the "if" statement in line 203 is pointless because the statements from line 203 to line 206 can be replaced with just $y \xleftarrow{\$} \{0, 1\}^n$. This redundancy will be useful in comparing with 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 as S. Since R and S in Eq. (4) correspond to R_1 and S_1, respectively, we have

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

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

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

For some adversary A, $\mathbf{Adv}^{\text{pro}}_{H,S}(A)$ might be not zero because S_1 cannot fully emulate the ideal cipher E (i.e., line 209 and line 220).

We compare the game G_1 and a game G_2 of Fig. 5. As explained below, these games are equivalent. The game G_1 uses two tables $R_1[x \| z]$ and $S_1[k][x]$ that are used for the random oracle $R_1(z, x)$ and the simulator $S_1(\alpha, k, w)$, respectively. However, since these tables are dependent (e.g., line 204 in Fig. 4), it is possible to combine them to make one table. Indeed, a table $S_2[k][x]$ in the game G_2 works as these tables.

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

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

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

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

We compare the game G_0 and 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 $bad \leftarrow$ `true`. Namely, E_0 and S_2 are identical until the flag *bad* sets. This is called *identical-until-bad*. Before the flag *bad* becomes `true`, no algorithm can distinguish between E_0 and S_2, that is, for any algorithm A, the output of A^{G_0} is equal to that of A^{G_2}. Hence, if

†"fresh" means that the query is not made to the oracle until then.

the output of A^{G_0} is different from that of A^{G_2}, then the flag *bad* is `true`. Notice that even if *bad* becomes `true`, the output of A^{G_0} is not always different from that of A^{G_2} because A cannot see the value of *bad*. Using the fundamental lemma ([13] or Appendix), we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) &= \Pr\left[A^{G_0}\Rightarrow 1\right] - \Pr\left[A^{G_2}\Rightarrow 1\right] \\ &\le \Pr\left[G_0 \text{ sets } bad\right],\end{aligned} \tag{8}$$

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

$$\begin{aligned}&\Pr[G_0 \text{ sets } bad] \\ &\quad \le \Pr[G_0 \text{ sets } bad_e] + \Pr[G_0 \text{ sets } bad_d],\end{aligned} \tag{9}$$

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-th 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-th query is not greater than $(i-1)/2^n$, and the probability that bad_d is set at the i-th 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

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

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

where we assumed that $q \le 2^{n-1}+1$. Substituting the above inequalities into Eq. (9), we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) &\le \Pr[G_0 \text{ sets } bad] \\ &\le \frac{q(q+3)}{2^{n+1}}.\end{aligned} \tag{10}$$

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 [10].

$$\begin{aligned}&\mathbf{Adv}^{\mathrm{col}}_{H}(B) \\ &= \Pr\left[B^{H^E,E}\Rightarrow((z,x),(z',x')) : \mathsf{ocol}\right],\end{aligned} \tag{11}$$

where ocol means that one of the following events occurs.

- $(z,x) \ne (z',x') \wedge H(z,x) = H(z',x')$
- 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

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

Let C_i be the event that there exists $j \in \{1,2,\ldots,i-1\}$ such that $(z_i,x_i) \ne (z_j,x_j) \wedge H_0(z_i,x_i) = H_0(z_j,x_j)$ 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 $\Pr[\mathsf{C}_i] \le i/(2^n-(i-1))$. Notice that this inequality is vacuous if $i \ge 2^{n-1}+1$. Hence,assuming that $q \le 2^{n-1}$, we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}}_{H}(B) &\le \Pr\left[\mathsf{C}_1 \vee \mathsf{C}_2 \vee \ldots \vee \mathsf{C}_q\right] \\ &\le \sum_{i=1}^{q} \Pr[\mathsf{C}_i] \\ &\le \frac{1}{2^n-(q-1)}\sum_{i=1}^{q} i \\ &\le \frac{1}{2^n-(2^{n-1}-1)}\sum_{i=1}^{q} i \\ &\le \frac{1}{2^n-2^{n-1}}\sum_{i=1}^{q} i \\ &= \frac{q(q+1)}{2^n}.\end{aligned} \tag{12}$$

4. The CP Compression Function

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

$$H(z,x) = E(1, x \,\|\, z, c), \tag{13}$$

where c is a public constant string, say 0^n. In this section, we omit the subscript 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. What is important in achieving these properties is that the selection of a permutation depends on both of x and z, that is, a permutation for x and z is selected as $E(1, x \,\|\, z, \cdot)$. For example, lines 203, 209, 218 in Fig. 5 check whether the selected permutations are used. In contrast, it is not so important what E encrypt. Hence, the analysis of the CP compression function will be similar to that of the LM compression function.

4.1 Indifferentiability

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

$$\mathbf{Adv}^{\text{pro}}_{H,S}(A) = \Pr\left[A^{H^E,E}\Rightarrow 1\right] - \Pr\left[A^{R,S^R}\Rightarrow 1\right], \qquad (14)$$

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. (13) 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 in discussing games. We next consider a game G_4 as shown in Fig. 7. In Fig. 7, 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. (14) is rewritten as

$$\mathbf{Adv}^{\text{pro}}_{H,S}(A) = \Pr\left[A^{G_3}\Rightarrow 1\right] - \Pr\left[A^{G_4}\Rightarrow 1\right]. \qquad (15)$$

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. (15) is rewritten as

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

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 $\Pr[G_3 \text{ sets } \mathit{bad}]$ in a similar way to Sect. 3.1.

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

The above bound is the same as Eq. (10). Comparing Eq. (3) and Eq. (13), we observe that encrypting variable x in the LM compression function does not improve the upper bound of the advantage.

```
The CP compression function H3(z, x)
    return E3(1, x || z, c)

Ideal cipher E3(α, k, w)
    if α = 1 then
        x←w
        if E3[k][x] =⊥ then
            if x = c then
                y ←$ {0,1}^n
            else
                y ←$ {0,1}^n
                if y ∈ Y(E3[k]) then
                    bad←true   // bad_e
                    y ←$ ¬Y(E3[k])
            E3[k][x]←y
        return E3[k][x]
    if α = −1 then
        y←w
        Find x s.t. E3[k][x] = y.
        if no such an x then
            x ←$ ¬X(E3[k])
            if x = c then
                bad←true   // bad_d
            E3[k][x]←y
        return x
```

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

```
Random oracle R4(z, x)
100 if R4[x || z] =⊥ then
101     R4[x || z] ←$ {0,1}^n
102 return R4[x || z]

Simulator S4(α, k, w)
200 if α = 1 then
201     x←w
202     if S4[k][x] =⊥ then
203         if x = c then
204             y←R4(z, x)
205         else
206             y ←$ {0,1}^n
207             if y ∈ Y(S4[k]) then
208                 bad←true
209                 if x ≠ c then
210                     y ←$ ¬Y(S4[k])
211         S4[k][x]←y
212     return S4[k][x]
213 if α = −1 then
214     y←w
215     Find x s.t. S4[k][x] = y.
216     if no such an x then
217         x ←$ ¬X(S4[k])
218         if x = c then
219             bad←true
220             S4[k][x]←R4(z, x)
221             x ←$ ¬X(S4[k])
222         S4[k][x]←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

```
Function R5(z, x)
    return S5(1, x || z, c)

Simulator S5(α, k, w)
    if α = 1 then
        x←w
        if S5[k][x] =⊥ then
            if x = c then
                y ←$ {0, 1}^n
            else
                y ←$ {0, 1}^n
            if y ∈ Y(S5[k]) then
                bad←true
                if x ≠ c then
                    y ←$ ‾Y(S5[k])‾
            S5[k][x]←y
        return S5[k][x]
    if α = −1 then
        y←w
        Find x s.t. S5[k][x] = y.
        if no such an x then
            x ←$ ‾X(S5[k])‾
            if x = c then
                bad←true
                S5[k][x]←{0, 1}^n
                x ←$ ‾X(S5[k])‾
            S5[k][x]←y
        return x
```

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

compression function H, we define the col-advantage of adversary B as

$$\mathbf{Adv}_H^{\mathrm{col}}(B) = \Pr\left[B^{H^E,E} \Rightarrow ((z,x),(z',x')) : \mathsf{ocol}\right]. \tag{16}$$

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

$$\begin{aligned}\mathbf{Adv}_H^{\mathrm{col}}(B) &= \Pr\left[B^{G_3} \Rightarrow ((z,x),(z',x')) : \mathsf{ocol}\right] \\ &\leq \frac{q(q+1)}{2^n},\end{aligned} \tag{17}$$

where we assumed that $q \leq 2^{n-1}$. The above bound is the same as Eq. (12).

5. Concluding Remarks

The problem of building a multi-property hash function was reduced to that of building a multi-property compression function due to [9]. Hence, it is significant to build a multi-property compression function.

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 used for popular hash functions such as the SHA family [18], 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.

Acknowledgments

The authors would like to thank anonymous reviewers for useful comments. This research was partially supported by the National Institute of Information and Communications Technology, Japan.

References

[1] H. Kuwakado and M. Morii, "Compression functions suitable for the multi-property-preserving transform," Cryptology ePrint Archive, Report 2007/302, 2007. http://eprint.iacr.org/

[2] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," Advances 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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] H. Kuwakado and M. Morii, "Indifferntialbility of single-block-length and rate-1 compression functions," IEICE Trans. Fundamentals, vol.E90-A, no.10, pp.2301–2308, Oct. 2007.

[12] 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.
[13] 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/
[14] H. Kuwakado and M. Morii, "Restrictively sanitizable signature scheme," Proc. 2006 Symposium on Cryptography and Information Security, 2006.
[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.
[16] 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/
[17] T. Shrimpton and M. Stam, "Building a collision-resistant compression function from non-compressing primitives." Cryptology ePrint Archive, Report 2007/409, 2007. http://eprint.iacr.org/
[18] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publication 180-2, Aug. 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

Appendix: Proof of Eq. (8)

The adversary A may make random coin tosses, and the game G_0 (or G_2) makes random coin tosses (e.g., line 204 in Fig. 3). Consider a sequence c of random coin tosses. The sequence c represents results of all the random coin tosses needed by both of A and G_0 (or G_2). There is a finite set C that includes all such possible sequences. Each sequence $c \in C$ will result in a particular behavior of A as A plays the game G_0, or a particular behavior of A as A plays the game G_2.

Suppose that the output b of A is 0 or 1. Consider a sequence $c \in C$ that causes A to output b if A plays G_0. Let $C^b_{G_0}$ be the set of all such sequences. The set $C^b_{G_0}$ is divided into two disjoint subsets as

$$C^b_{G_0} = C^{b,\mathtt{true}}_{G_0} \cup C^{b,\mathtt{false}}_{G_0}, \quad C^{b,\mathtt{true}}_{G_0} \cap C^{b,\mathtt{false}}_{G_0} = \emptyset, \quad \text{(A· 1)}$$

where $C^{b,\mathtt{true}}_{G_0}$ ($\subseteq C^b_{G_0}$) is the set of sequences such that the flag *bad* becomes $\mathtt{true}$ and $C^{b,\mathtt{false}}_{G_0}$ ($\subseteq C^b_{G_0}$) is the set of sequences such that *bad* does not become $\mathtt{true}$. For the game G_2, sets $C^b_{G_2}$, $C^{b,\mathtt{true}}_{G_2}$, and $C^{b,\mathtt{false}}_{G_2}$ are defined in a similar way. Since G_0 and G_2 are identical-until-bad, $c \in C^{b,\mathtt{false}}_{G_0}$ if and only if $c \in C^{b,\mathtt{false}}_{G_2}$, that is,

$$C^{b,\mathtt{false}}_{G_0} = C^{b,\mathtt{false}}_{G_2}. \quad \text{(A· 2)}$$

In addition, consider a sequence $c \in C$ such that *bad* becomes $\mathtt{true}$ if A plays G_0. Let $C^{\mathtt{true}}_{G_0}$ be the set of all such sequences. Note that

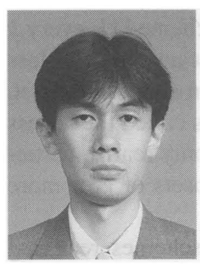
$$C^{\mathtt{true}}_{G_0} = C^{0,\mathtt{true}}_{G_0} \cup C^{1,\mathtt{true}}_{G_0}. \quad \text{(A· 3)}$$

Now, we have

$$\Pr\left[A^{G_0} \Rightarrow 1\right] = \frac{\left|C^1_{G_0}\right|}{|C|}, \quad \Pr\left[A^{G_2} \Rightarrow 1\right] = \frac{\left|C^1_{G_2}\right|}{|C|},$$

where $\left|C^1_{G_0}\right|$, $\left|C^1_{G_2}\right|$, and $|C|$ denote the numbers of elements in $C^1_{G_0}$, $C^1_{G_2}$, and C, respectively. Hence, the pro-advantage of A is calculated as follows.

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{pro}}_{H,S}(A) \\
&= \Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_2} \Rightarrow 1\right] \\
&= \frac{\left|C^1_{G_0}\right|}{|C|} - \frac{\left|C^1_{G_2}\right|}{|C|} \\
&= \frac{\left|C^{1,\mathtt{true}}_{G_0}\right| + \left|C^{1,\mathtt{false}}_{G_0}\right|}{|C|} - \frac{\left|C^{1,\mathtt{true}}_{G_2}\right| + \left|C^{1,\mathtt{false}}_{G_2}\right|}{|C|} && \text{(by Eq. (A· 1))} \\
&= \frac{\left|C^{1,\mathtt{true}}_{G_0}\right| - \left|C^{1,\mathtt{true}}_{G_2}\right|}{|C|} && \text{(by Eq. (A· 2))} \\
&\leq \frac{\left|C^{1,\mathtt{true}}_{G_0}\right|}{|C|} \\
&\leq \frac{\left|C^{\mathtt{true}}_{G_0}\right|}{|C|} && \text{(by Eq. (A· 3))} \\
&= \Pr\left[G_0 \text{ sets } \mathit{bad}\right].
\end{aligned}$$



Hidenori Kuwakado received the B.E., M.E. and D.E. degrees from Kobe University in 1990, 1992, and 1999 respectively. He worked for Nippon Telegraph and Telephone Corporation from 1992 to 1996. From 1996 to 2002 he was a Research Associate in the Faculty of Engineering, Kobe University. From 2002 to 2007, he was an Associate Professor in the Faculty of Engineering, Kobe University. Since 2007, he has been an Associate Professor in Graduate School of Engineering, Kobe University. His research interests are in cryptography and information security.

Masakatu Morii received the B.E. degree in electrical engineering and the M.E. degree in electronics engineering from Saga University, Saga, Japan, and the D.E. degree in communication engineering from Osaka University, Osaka, Japan, in 1983, 1985, and 1989, respectively. From 1989 to 1990 he was an Instructor in the Department of Electronics and Information Science, Kyoto Institute of Technology, Japan. From 1990 to 1995 he was an Associate Professor at the Department of Computer Science, Faculty of Engineering at Ehime University, Japan. From 1995 to 2005 he was a Professor at the Department of Intelligent Systems and Information Science, Faculty of Engineering at the University of Tokushima, Japan. From 2005 to 2007, he was a Professor at the Department of Electrical and Electronics Engineering, Faculty of Engineering at the Kobe University, Japan. Since 2007, he has been a Professor at the Department of Electrical and Electronic Engineering, Graduate School of Engineering at the Kobe University, Japan. His research interests are in error correcting codes, cryptography, discrete mathematics and computer networks and information security. Dr. Morii is a member of the IEEE, the Japan Society for Industrial and Applied Mathematics, the Information Processing Society of Japan and the Society of Information Theory and its Applications.